nexus

Release v2026.01

NIAC, https://www.nexusformat.org

Feb 19, 2026

CONTENTS

1 NeXus: User Manual **3**
- 1.1 NeXus Introduction . . . 3
- 1.2 NeXus Design . . . 18
- 1.3 Constructing NeXus Files and Application Definitions . . . 62
- 1.4 Strategies for storing information in NeXus data files . . . 74
- 1.5 Verification and validation of files . . . 78
- 1.6 Frequently Asked Questions . . . 78

2 Examples of writing and reading NeXus data files **83**
- 2.1 Code Examples in Various Languages . . . 83
- 2.2 Visualization tools . . . 118
- 2.3 Examples for Specific Instruments . . . 122
- 2.4 Other tools to handle NeXus data files . . . 139

3 NeXus: Reference Documentation **141**
- 3.1 Introduction to NeXus definitions . . . 141
- 3.2 NXDL: The NeXus Definition Language . . . 145
- 3.3 NeXus Class Definitions . . . 175

4 NAPI: NeXus Application Programmer Interface (frozen) **1185**
- 4.1 Status . . . 1185
- 4.2 Overview . . . 1185
- 4.3 Core API . . . 1186
- 4.4 Utility API . . . 1193
- 4.5 Building Programs . . . 1194
- 4.6 Reporting Bugs in the NeXus API . . . 1195

5 NeXus Community **1197**
- 5.1 NeXus Webpage . . . 1197
- 5.2 Contributed Definitions . . . 1197
- 5.3 Other Ways NeXus Coordinates with the Scientific Community . . . 1197

6 Installation **1201**
- 6.1 Precompiled Binary Installation . . . 1201
- 6.2 Source Installation . . . 1202
- 6.3 Releases . . . 1202

7 NeXus Utilities **1205**
- 7.1 Utilities supplied with NeXus . . . 1205
- 7.2 Validation . . . 1206
- 7.3 Other Utilities . . . 1206

7.4 Data Analysis . . . 1207
7.5 HDF Tools . . . 1208
7.6 Language APIs for NeXus and HDF5 . . . 1209

8 Brief history of NeXus **1211**

9 About these docs **1215**
9.1 Authors . . . 1215
9.2 Colophon . . . 1215
9.3 Revision History . . . 1216
9.4 Copyright and Licenses . . . 1216

Index **1217**

https://www.nexusformat.org/

CHAPTER

ONE

NEXUS: USER MANUAL

1.1 NeXus Introduction

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

1. *raw data*: NeXus defines a format that can serve as a container for all relevant data associated with a scientific instrument or beamline. This is a very important use case. This includes the case of streaming data acquisition, where time stamped data are logged.
2. *processed data*: NeXus also defines standards for processed data. This is data which has underwent some form of data reduction or data analysis. NeXus allows storing the results of such processing together with documentation about how the processed data was generated.
3. *standards*: NeXus defines standards in the form of *application definitions* for the exchange of data between applications. NeXus provides standards for both raw and processed data.

A community of scientists and computer programmers working in neutron and synchrotron facilities around the world came to the conclusion that a common data format would fulfill a valuable function in the scattering community. As instrumentation becomes more complex and data visualization becomes more challenging, individual scientists, or even institutions, find it difficult to keep up with new developments. A common data format makes it easier, both to exchange experimental results and to exchange ideas about how to analyze them. It promotes greater cooperation in software development and stimulates the design of more sophisticated visualization tools. Additional background information is given in the chapter titled *Brief history of NeXus*.

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

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

[1] *J. Appl. Cryst.* (2015). **48**, 301-305 (https://doi.org/10.1107/S1600576714027575)

1.1.1 What is NeXus?

The NeXus data format has four components:

A set of design principles
: to help people understand what is in the data files.

A set of data storage objects
: (*Base Class Definitions* and *Application Definitions*) to allow the development of portable analysis software.

A set of subroutines
: (*Utilities* and *examples*) to make it easy to read and write NeXus data files.

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

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

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

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

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

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

A Set of Design Principles

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

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

Fields
: Fields can be scalar values or multidimensional arrays of a variety of sizes (1-byte, 2-byte, 4-byte, 8-byte) and types (characters, integers, floats). Fields are represented as HDF5 *datasets*.

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

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

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

Example of a NeXus File

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

Example of a NeXus Data File

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

Important Classes

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

Note

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

NXentry

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

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

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

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

`NXinstrument` excerpt

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

Simple Example

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

Example structure of a simple data file

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

Structure of a very simple NeXus Data file

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

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

Plot of a very simple NeXus HDF5 Data file

NeXus files are easy to create. This example NeXus file was created using a short Python program and the *h5py* package:

Using Python to write a very simple NeXus HDF5 Data file

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

import h5py

angle = [18.9094, 18.9096, 18.9098, 18.91,  18.9102,
         18.9104, 18.9106, 18.9108, 18.911, 18.9112,
         18.9114, 18.9116, 18.9118, 18.912, 18.9122]
diode = [1193, 4474, 53220, 274310, 515430, 827880,
         1227100, 1434640, 1330280, 1037070, 598720,
         316460, 56677, 1000, 1000]

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

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

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

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

    counts = nxdata.create_dataset('counts', data=diode)
```

(continues on next page)

(continued from previous page)

```
    counts.attrs['units'] = 'counts'
    counts.attrs['long_name'] = 'photodiode counts'
```

A Set of Data Storage Objects

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

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

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

A very simple NeXus Definition Language (NXDL) file

```xml
<?xml version="1.0" ?>
<definition
  xmlns="http://definition.nexusformat.org/nxdl/3.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
  category="base"
  name="NXverysimple"
  type="group" extends="NXobject">

  <doc>
    A very simple NeXus NXDL file
  </doc>
  <group type="NXentry">
    <group type="NXdata">
      <field name="counts" type="NX_INT"  units="NX_UNITLESS">
        <doc>counts recorded by detector</doc>
      </field>
      <field name="two_theta" type="NX_FLOAT" units="NX_ANGLE">
        <doc>rotation angle of detector arm</doc>
      </field>
    </group>
  </group>
</definition>
```

Complete examples of reading and writing NeXus data files are provided *later*. This chapter has several examples of writing and reading NeXus data files. If you want to define the format of a particular type of NeXus file for your own use, e.g. as the standard output from a program, you are encouraged to *publish* the format using this XML format. An example of how to do this is shown in the *Creating a NXDL Specification* section.

A Set of Subroutines

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

In the past, a data format was defined by a document describing the precise location of every item in the data file, either as row and column numbers in an ASCII file, or as record and byte numbers in a binary file. It is the job of the subroutine library to retrieve the data. This subroutine library is commonly called an application-programmer interface or API.

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

Simple example of reading data using the NeXus API

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

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

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

Scientific Community

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

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

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

Representation of data examples

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

example of NeXus data file structure

```
entry:NXentry
  instrument:NXinstrument
    detector:NXdetector
      data:[]
        @long_name = "strip detector 1-D array"
      bins:[0, 1, 2, ... 1023]
        @long_name = "bin index numbers"
  sample:NXsample
```

(continues on next page)

(continued from previous page)

```
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).

[1] https://github.com/nexusformat/code/releases/

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
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). NAPI[2] returns a file identifier of type `NXhandle`.
2. **line 7:**
   Next, we create the *NXentry* group to contain the scan using `NXmakegroup()` and then open it for access using `NXopengroup()`.[3]
3. **line 10:**
   The plottable data is contained within an *NXdata* group, which must also be created and opened.
4. **line 12:**
   To create a field, call `NXmakedata()`, specifying the data name, type (`NX_FLOAT32`), rank (in this case, 1), and length of the array (`n`). Then, it can be opened for writing.[4]

[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. **line 14:**
   Write the data using `NXputdata()`.
6. **line 15:**
   With the field still open, we can also add some field attributes, such as the data units,[5][6] which are specified as a character string (`type="NX_CHAR"`[7]) that is 7 bytes long.
7. **line 16:**
   Then we close the field before opening another. In fact, the API will do this automatically if you attempt to open another field, but it is better style to close it yourself.
8. **line 17:**
   The remaining fields in this group are added in a similar fashion. Note that the indentation whenever a new field or group are opened is just intended to make the structure of the NeXus file more transparent.
9. **line 20:**
   Finally, close the groups (`NXdata` and `NXentry`) before closing the file itself.

How do I read a NeXus file?

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

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

Reading a simple NeXus file using NAPI

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

How do I browse a NeXus file?

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

[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*

Using `nxbrowse`

```
%> nxbrowse lrcs3701.nxs

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

program analysis

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

The source code of `nxbrowse`[8] provides an example of how to write a NeXus reader. The test programs included in the *NeXus API* may also be useful to study.

1.2 NeXus Design

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

The structure of NeXus files is extremely flexible, allowing the storage both of simple data sets, such as a single data array and its axes, and also of highly complex data, such as the simulation results or an entire multi-component instrument. This flexibility is a necessity as NeXus strives to capture data from a wild variety of applications in X-ray, muSR and neutron scattering. The flexibility is achieved through a hierarchical structure, with related *fields* collected together into *groups*, making NeXus files easy to navigate, even without any documentation. NeXus files are self-describing, and should be easy to understand, at least by those familiar with the experimental technique.

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.

[8] https://github.com/nexusformat/code/blob/master/applications/NXbrowse/NXbrowse.c

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

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

Note

Notation used to describe a NeXus data file

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

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

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

Groups

NeXus files consist of data groups, which contain fields and/or other groups to form a hierarchical structure. This hierarchy is designed to make it easy to navigate a NeXus file by storing related fields together. Data groups are identified both by a name, which must be unique within a particular group, and a class. There can be multiple groups with the same class but they must have different names (based on the HDF rules).

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

Fields

Fields (also called data fields, data items or data sets) contain the essential information stored in a NeXus file. They can be scalar values or multidimensional arrays of a variety of sizes (1-byte, 2-byte, 4-byte, 8-byte) and types (integers, floats, characters). The fields may store both experimental results (counts, detector angles, etc.), and other information associated with the experiment (start and end times, user names, etc.). Fields are identified by their names, which must be unique within the group in which they are stored. Some fields have engineering units to be specified. In some cases, such as */NXdata/DATA*, a field is expected to be an array of several dimensions.

Examples of fields

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

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

`wavelength` (*NX_FLOAT*)
wavelength of radiation, `units="NX_FLOAT"`.

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

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

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

See the sections *Data Types allowed in NXDL specifications* and *Unit Categories allowed in NXDL specifications* for complete lists of the data types and engineering units types, respectively.

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

Attributes

Attributes are extra (meta-)information that are associated with particular groups or fields. They are used to annotate data, e.g. with physical units or calibration offsets, and may be scalar numbers or character strings. In addition, NeXus uses attributes to identify plottable data and their axes, etc. In a *tree structure*, an attribute is usually shown with a `@` prefix, such as `@units`. A description of some of the many possible attributes can be found in the next table:

Examples of attributes

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

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

`long_name` (*NX_CHAR*)
Defines title of signal data or axis label of dimension scale

`calibration_status` (*NX_CHAR*)
Defines status of data value - set to `Nominal` or `Measured`

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

`interpretation` (*NX_CHAR*)
Describes how to display the data. `rgba`, `hsla` and `cmyk` are (*n x m x 4*) arrays, where the 4 channels are the colour channels appropriately. If the image data does not contain an alpha channel, then the array should simply be (*n x m x 3*). Allowed values include:

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

File attributes

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

Links

Python h5py code to make NeXus links

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

- **{hdf5_object}._id.link()**
- **h5py.ExternalLink()**

Links are pointers to existing data somewhere else. The concept is very much like symbolic links in a unix filesystem. The NeXus definition sometimes requires to have access to the same data in different groups in the same file. For example: detector data is stored in the *NXinstrument*/*NXdetector* group but may be needed in *NXdata* for automatic plotting. Rather then replicating the data, NeXus uses links in such situations. See the *figure* for a more descriptive representation of the concept of linking.

NeXus links are HDF5 hard links with an additional `target` attribute. The `target` attribute is added[1] for NeXus to

[1] When using the NAPI, the `target` attribute is added automatically. When the NAPI is not used to write NeXus/HDF5 files, this attribute must be added. Here are the steps to follow:

1. Get the HDF5 reference ID of the source item (*field*, *group*, or *link*) to be linked.
2. If the ID does not have a `target` attribute defined: #. Get the absolute HDF5 address[3] of the ID. #. Create a `target` attribute for the ID. #. Set the `target` attribute's value to the absolute HDF5 address of the ID.
3. Create an HDF5 hard link[4] to the ID at the desired (new) HDF5 address.

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

Note

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

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

Fig. 1: Linking in a NeXus file

distinguish the HDF5 path to the *original*[2] dataset. The value of the `target` attribute is the HDF5 path[Page 21, 3] to the *original* dataset.

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

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

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

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

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

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

[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?**

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

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

External File Links

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

Example of linking to data in an external HDF5 file

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

Note

The NAPI code[5] makes no `target` attribute assignment for links to external files. It is best to avoid using the `target` attribute with external file links. The NIAC is working at resolving the technical limitations

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

[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

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 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.)

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:

[7] see: https://github.com/nexusformat/definitions/issues/397

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

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

NeXus Application Definitions

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

Note

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

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

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

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

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

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

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

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

Actions of standard NeXus fields

Transformation Actions

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

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

implicit order of NeXus spherical coordinate system

```
azimuthal_angle:polar_angle:distance
```

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

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

Shape Descriptions

`NXoff_geometry`

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

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

```
OFF
#  cube.off
#  A cube

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

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

The list of vertices in an OFF file maps directly to the `vertices` dataset in the *NXoff_geometry* class. The vertex indices of the face list in the OFF file occupy the `winding_order` dataset of the NeXus class, however the list is flattened to 1D in order to avoid a ragged-edged dataset, which are not easy to work with using HDF libraries. A `faces` dataset

contains the position of the first entry in `winding_order` for each face. The `NXoff_geometry` equivalent of the OFF cube example is shown below.

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

NXcylindrical_geometry

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

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

Fig. 3: Cylinder definition with three vertices

Detector Shape Descriptions

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

Legacy Geometry Descriptions

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

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

McStas and `NXgeometry` System

As stated above, NeXus uses the *McStas coordinate system* (http://mcstas.org) as its laboratory coordinate system. The instrument is given a global, absolute coordinate system where the z axis points in the direction of the incident beam, the x axis is perpendicular to the beam in the horizontal plane pointing left as seen from the source, and the y axis points upwards. See below for a drawing of the McStas coordinate system. The origin of this coordinate system is the sample position or, if this is ambiguous, the center of the sample holder with all angles and translations set to zero. The McStas coordinate system is illustrated in the next figure:

Fig. 4: The McStas Coordinate System

The NeXus NXgeometry class directly uses the McStas coordinate system. `NXgeometry` classes can appear in any component in order to specify its position. The suggested name to use is geometry. In `NXgeometry` the `NXtranslation/values` field defines the absolute position of the component in the McStas coordinate system. The `NXorientation/value` field describes the orientation of the component as a vector of in the McStas coordinate system.

Simple (Spherical Polar) Coordinate System

In this system, the instrument is considered as a set of components through which the incident beam passes. The variable *distance* is assigned to each component and represents the effective beam flight path length between this component and the sample. A sign convention is used where negative numbers represent components pre-sample and positive numbers components post-sample. At each component there is local spherical coordinate system with the angles *polar_angle* and *azimuthal_angle*. The size of the sphere is the distance to the previous component.

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

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

1.2.3 Rules and Underlying File Formats

Rules for Structuring Information in NeXus Files

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

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

Content of a Raw Data `NXentry` Group

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

NeXus Raw Data Hierarchy

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

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

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

Content of a processed data `NXentry` group

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

NeXus Processed Data Hierarchy

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

(continues on next page)

(continued from previous page)

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

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

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

`NXsubentry` or Multi-Method Data

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

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

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

NeXus Multi Method Hierarchy

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

(continues on next page)

(continued from previous page)

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

Rules for Special Cases

Scans

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

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

Simple scan

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

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

NeXus Simple Scan Example

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

Simple scan with area detector

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

NeXus Simple Scan Example with Area Detector

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

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

Complex *hkl* scan

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

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

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

NeXus Complex *hkl* Scan

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

(continues on next page)

(continued from previous page)

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

Multi-parameter scan: XAS

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

The example is for X-ray Absorption Spectroscopy (XAS) data where the monochromator energy is step-scanned and counts are read back from detectors before (`I0`) and after (`I`) the sample. These energy scans are repeated at a sequence of sample temperatures to map out, for example, a phase transition. While it is customary in XAS to plot *log(I0/I)*, we show them separately here in two different `NXdata` groups to demonstrate that such things are possible. Note that the length of the 1-D energy array is `NE` while the length of the 1-D temperature array is `NT`

NeXus Multi-parameter scan: XAS

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

(continues on next page)

(continued from previous page)

```
                data:NX_NUMBER[NE,NT]
                energy --> /entry/monochromator/energy
                temperature --> /entry/sample/temperature
        sample:NXsample
            temperature:NX_NUMBER[NT]
        monochromator:NXmonochromator
            energy:NX_NUMBER[NE]
        I_data:NXdata
            @signal = "data"
            @axes = ["energy", "temperature"]
            @energy_indices = 0
            @temperature_indices = 0
            data --> /entry/instrument/I/data
            energy --> /entry/monochromator/energy
            temperature --> /entry/sample/temperature
        I0_data:NXdata
            @signal = data
            @axes = ["energy", "temperature"]
            @energy_indices = 0
            @temperature_indices = 0
            data --> /entry/instrument/I00/data
            energy --> /entry/monochromator/energy
            temperature --> /entry/sample/temperature
```

Rastering

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

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

Warning

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

Streaming Data Acquisition And Logging

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

Just use the standard NeXus raw data file structure, but replace the corresponding data object with an *NXlog* or *NXevent_data* structure of the same name.

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

NXcollection

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

`NXcollection` Example

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

Rules for Storing Data Items in NeXus Files

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

Naming Conventions

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

- The names of NeXus *group* and *field* items must only contain a restricted set of characters.

  This set is described by a regular expression syntax regular expression *regular expression syntax*, as described below.

- For the class names[1] of NeXus *group* items, the prefix *NX* is reserved as shown in the *table* below. Thus all NeXus class names start with NX. The chapter titled *NeXus: Reference Documentation* lists the available NeXus class names as either *base classes*, *application definitions*, or *contributed definitions*.

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

NXDL group and field names

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

- *Recommended* names[3]
  - lower case words separated by underscores and, if needed, with a trailing number
  - NOTE: this is used by the NeXus base classes
- *Allowed* names
  - any combination of upper and lower case letter, numbers, underscores and periods, except that periods cannot be at the start or end of the string
  - NOTE: this matches the *validItemName* regular expression *below*
- *Invalid* names
  - NOTE: does not match the *validItemName* regular expression *below*

Regular expression pattern for NXDL group and field names

The NIAC recognises that the majority of the world uses characters outside of the basic latin (a.k.a. US-ASCII, 7-bit ASCII)[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*):

```
^[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.

```
number_of_lenses
```

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

> **Note**
>
> The NeXus base classes provide a comprehensive dictionary of terms that can be used for each class. The expected spelling and definition of each term is specified in the base classes. It is not required to provide all the terms specified in a base class. Terms with other names are permitted but might not

[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

> 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:

```
temperature
temperature_set
```

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

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

```
unit_cell_abc
unit_cell_alphabetagamma
unit_cell_volume
```

Reserved prefixes

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

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

Reserved suffixes

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

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

Variants

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

A little example is in order to illustrate the scheme:

```
beam_center_x
        @variant=beam_center_x_refined
beam_center_x_refined
        @variant=beam_center_x_initial_guess
beam_center_x_initial_guess
```

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

[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`

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:

```
data
data_errors
beam_center_x
beam_center_x_errors
```

Where data errors would contain the errors on data, and beam_center_x_errors the error on the beam center for x.

NeXus Array Storage Order

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

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

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

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

Non C Storage Order

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

Offset and Stride for 1 D data:

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

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

   * reverse stride:
```

(continues on next page)

(continued from previous page)

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

Offset and Stride for 2D Data

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

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

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

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

Offset and Stride for 3D Data

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

  * row major (C) stride:
    stride[3] = { 20, 5, 1 }
    offset[3] = { 0, 0, 0 }
    for i:
       for j:
          for k:
             result[i][j][k]:
                0 1 2 3 4
                5 6 7 8 9
                10 11 12 13 14
                15 16 17 18 19

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

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

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

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

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

NeXus Data Types

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

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

integers
: `NX_INT8`, `NX_INT16`, `NX_INT32`, or `NX_INT64`

floating-point numbers
: `NX_FLOAT32` or `NX_FLOAT64`

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

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

binary data
: Binary data is to be written as `UINT8`.

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

NeXus dates and times

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

[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.

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

Find the plottable data

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

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

version 2
Associating plottable data by name using the axes attribute

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

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.

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

   Open the `NXdata` group selected above. If it has a `signal` attribute, the attribute's value is the name of the field to be plotted. (The value of the `signal` attribute *names* an existing child of this group. The child group must

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

itself be a NeXus field.) If no `signal` attribute is present on the `NXdata` group, then proceed to try an *older NeXus method* to find the default plottable data.

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

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

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

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

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

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

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

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

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

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

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

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

# assumes group.attrs["NX_class"] == "NXdata"
signal_field_name = group.attrs["signal"]
data = group[signal_field_name]

plot(data)
```

Version 2

Tip

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

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

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

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

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

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

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

   This is the default plottable data.

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

   1. If this field has an attribute `axes`:

      1. The `axes` attribute value contains a colon (or comma) delimited list (in the C-order of the data array) with the names of the dimension scales associated with the plottable data. Such as: `axes="polar_angle:time_of_flight"`

2. Parse `axes` and open the fields to describe your dimension scales

2. If this field has no attribute `axes`:

   1. Search for fields with attributes `axis=1`, `axis=2`, etc.

   2. These are the fields describing your axis. There may be several fields for any axis, i.e. there may be multiple fields with the attribute `axis=1`. Among them the field with the attribute `primary=1` is the preferred one. All others are alternative dimension scales.

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

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

Version 1

Tip

Try this method for older NeXus data files.

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

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

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

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

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

3. Loop through NeXus fields in this group searching for the item with attribute `signal="1"` indicating this field has the plottable data.
4. Search for the one-dimensional NeXus fields with attribute `primary=1`. These are the dimension scales to label the axes of each dimension of the data.
5. Link each dimension scale to the respective data dimension by the `axis` attribute (`axis=1`, `axis=2`, ... up to the rank of the data).

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

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 to identify the default data to be plotted is quite simple. Given any NeXus data file, any `NXentry`, or any `NXdata`, follow the chain as it is described from that point. Specifically:

- The root of the NeXus file may have a `default` attribute that names the default *NXentry* group. This attribute may be omitted if there is only one NXentry group. If a second NXentry group is later added, the `default` attribute must be added then.
- Every *NXentry* group may have a `default` attribute that names the default *NXdata* group. This attribute may be omitted if there is only one NXdata group or if no NXdata is present. If a second NXdata group is later added, the `default` attribute must be added then.
- Every *NXdata* group will have a `signal` attribute that names the field name to be plotted by default. This attribute is required.

`axes`

String array[8] that defines the independent data fields used in the default plot for all of the dimensions of the *signal* field. One entry is provided for every dimension in the *signal* field.

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

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

See examples provided on the NeXus webpage ([9]).

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

`AXISNAME_indices`

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

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

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

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

[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

[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

```
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".

Examples

Several examples are provided to illustrate this method. More examples are available in the NeXus webpage ([Page 55, 9]).

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
    pressure: float[20]
    temperature: float[20]
    time: float[1000]
```

Associating plottable data by name using the `axes` attribute

Warning

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

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

denoting axes by name

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

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

Warning

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

The original method defines an attribute of each dimension scale field called *axis*. It is an integer whose value is the number of the dimension, in order of fastest varying dimension. That is, if the array being stored is data with elements `data[j][i]` in C and `data(i,j)` in Fortran, where `i` is the time-of-flight index and `j` is the polar angle index, the `NXdata` group would contain:

denoting axes by integer number

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

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

There are limited circumstances in which more than one dimension scale for the same data dimension can be included in the same `NXdata` group. The most common is when the dimension scales are the three components of an *(hkl)* scan. In order to handle this case, we have defined another attribute of type integer called `primary` whose value determines the order in which the scale is expected to be chosen for plotting, i.e.

- 1st choice: `primary=1`
- 2nd choice: `primary=2`
- etc.

If there is more than one scale with the same value of the `axis` attribute, one of them must have set `primary=1`. Defining the `primary` attribute for the other scales is optional.

Note

The `primary` attribute can only be
used with the first method of defining

dimension scales
discussed above. In addition to the `signal` data, this group could contain a data set of the same rank and dimensions called `errors` containing the standard deviations of the data.

Physical File format

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

Choice of HDF as Underlying File Format

At its beginnings, the founders of NeXus identified the Hierarchical Data Format (HDF) as a capable and efficient multi-platform data storage format. HDF was designed for large data sets and already had a substantial user community. HDF was developed and maintained initially by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC) and later spun off into its own group called The HDF Group (THG: http://www.hdfgroup.org/). Rather then developing its own unique physical file format, the NeXus group choose to build NeXus on top of HDF.

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

Mapping NeXus into HDF

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

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

Note

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

- *How do I write a NeXus file?*
- *How do I read a NeXus file?*
- ***HDF5 in C with NAPI***
  - *HDF5 in Python with NAPI*
- *Writing a simple NeXus file using native HDF5 commands in C*
- *Reading a simple NeXus file using native HDF5 commands in C*
- *Write a NeXus HDF5 File*
- *Read a NeXus HDF5 File*

Perhaps the easiest way to view the implementation of NeXus in HDF5 is to look at the data structure. For this, we use the `h5dump` command-line utility provided with the HDF5 support libraries. Short examples are provided for the basic NeXus data components:

- *group*: created in C NAPI by:

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

- *field*: created in C NAPI by:

```
NXmakedata (fileID, "two_theta", NX_FLOAT32, 1, &n);
  NXopendata (fileID, "two_theta");
NXputdata (fileID, tth);
```

- *attribute*: created in C NAPI by:

```
NXputattr (fileID, "units", "degrees", 7, NX_CHAR);
```

- *link* created in C NAPI by:

```
NXmakelink (fileid, &itemid);
# -or-
NXmakenamedlink (fileid, "linked_name", &itemid);
```

`h5dump` of a NeXus `NXentry` group

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

`h5dump` of a NeXus field (HDF5 dataset)

```
 DATASET "two_theta" {
    DATATYPE  H5T_IEEE_F64LE
    DATASPACE  SIMPLE { ( 31 ) / ( 31 ) }
    DATA {
    (0): 17.9261, 17.9259, 17.9258, 17.9256, 17.9254, 17.9252,
    (6): 17.9251, 17.9249, 17.9247, 17.9246, 17.9244, 17.9243,
    (12): 17.9241, 17.9239, 17.9237, 17.9236, 17.9234, 17.9232,
    (18): 17.9231, 17.9229, 17.9228, 17.9226, 17.9224, 17.9222,
    (24): 17.9221, 17.9219, 17.9217, 17.9216, 17.9214, 17.9213,
    (30): 17.9211
    }
    ATTRIBUTE "units" {
```

(continues on next page)

(continued from previous page)

```
      DATATYPE  H5T_STRING {
            STRSIZE 7;
            STRPAD H5T_STR_NULLPAD;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
      DATASPACE  SCALAR
      DATA {
      (0): "degrees"
      }
   }
   # ... other attributes
}
```

h5dump of a NeXus attribute

```
ATTRIBUTE "axes" {
   DATATYPE  H5T_STRING {
         STRSIZE 9;
         STRPAD H5T_STR_NULLPAD;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
   DATASPACE  SCALAR
   DATA {
   (0): "two_theta"
   }
}
```

h5dump of a NeXus link

```
# NeXus links have two parts in HDF5 files.

# The dataset is created in some group.
# A "target" attribute is added to indicate the HDF5 path to this dataset.

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

```

(continues on next page)

(continued from previous page)

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

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

1.3 Constructing NeXus Files and Application Definitions

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

Note

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

1.3.1 The WOnderful New Instrument (WONI)

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

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

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

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

Decide which parameters need to be stored

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

Mapping parameters to NeXus

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

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

Basic structure of a NeXus file with a monochromator added

```
entry:NXentry
  NXdata
  NXinstrument
     monochromator:Nxcrystal
        wavelength
        d_spacing
        rotation_angle
        reflection
        type
  NXmonitor
  NXsample
```

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

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

Decide on `NXdata`

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

Fill in auxiliary Information

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

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

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

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

1.3.3 Creating a NXDL Specification

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

One easy way to describe how to store data in the NeXus class structure and to create a NXDL specification is to work through an example. Along the way, we will describe some key decisions that influence our particular choices of metadata selection and data organization. So, on with the example …

Application Definition Steps

With all this introductory stuff out of the way, let us look at the process required to define an application definition:

1. *Think!* hard about what has to go into the data file.
2. *Map* the required fields into the NeXus hierarchy
3. *Describe* this map in a NXDL file
4. *Standardize* your definition through communication with the NIAC

Step 1: *Think!* hard about data

This is actually the hard bit. There are two things to consider:

1. What has to go into the data file?
2. What is the normal plot for this type of data?

For the first part, one of the NeXus guiding principles gives us - Guidance! "A NeXus file must contain all the data necessary for standard data analysis."

Not more and not less for an application definition. Of course the definition of *standard* data for analysis or a *standard* plot depends on the science and the type of data being described. Consult senior scientists in the field about this is if you are unsure. Perhaps you must call an international meeting with domain experts to haggle that out. When considering this, people tend to put in everything which might come up. This is not the way to go.

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

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

Now for the case of WONI, the standard data analysis is either Rietveld refinement or profile analysis. For both purposes, the kind of radiation used to probe the sample (for WONI, neutrons), the wavelength of the radiation, the monitor (which tells us how long we counted) used to normalize the data, the counts and the two theta angle of each detector element are all required. Usually, it is desirable to know what is being analyzed, so some metadata would be nice: a title, the sample name and the sample temperature. The data typically being plotted is two theta against counts, as shown in *Example Powder Diffraction Plot from (fictional) WONI at HYNES* above. Summarizing, the basic information required from WONI is given next.

- *title* of measurement
- sample *name*
- sample *temperature*
- counts from the incident beam *monitor*
- type of radiation *probe*
- *wavelength* (λ) of radiation incident on sample
- angle (2θ or *two theta*) of detector elements
- *counts* for each detector element

If you start to worry that this is too little information, hold on, the section on Using an Application Definition (*Using an Application Definition*) will reveal the secret how to go from an application definition to a practical file.

Step 2: *Map* Data into the NeXus Hierarchy

This step is actually easier then the first one. We need to map the data items which were collected in Step 1 into the NeXus hierarchy. A NeXus file hierarchy starts with an `NXentry` group. At this stage it is advisable to pull up the base class definition for `NXentry` and study it. The first thing you might notice is that `NXentry` contains a field named `title`. Reading the documentation, you quickly realize that this is a good place to store our title. So the first mapping has been found.

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

Note

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

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

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

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

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

For the other data items, there seem to be no solutions in `NXentry`. But each of these data items describe the instrument in more detail. NeXus stores instrument descriptions in the `/NXentry/NXinstrument` branch of the hierarchy. Thus, we continue by looking at the definition of the `NXinstrument` base class. In there we find further groups for all possible instrument components. Looking at the schematic of WONI (*The (fictional) WONI example powder diffractometer*), we realize that there is a source, a monochromator and a detector. Suitable groups can be found for these components in `NXinstrument` and further inspection of the appropriate base classes reveals the following further mappings:

```
probe = /NXentry/NXinstrument/NXsource/probe
wavelength = /NXentry/NXinstrument/NXcrystal/wavelength
two theta of detector elements = /NXentry/NXinstrument/NXdetector/polar angle
counts for each detector element = /NXentry/NXinstrument/NXdetector/data
```

Thus we mapped all our data items into the NeXus hierarchy! What still needs to be done is to decide upon the content of the `NXdata` group in `NXentry`. This group describes the data necessary to make a quick plot of the data. For WONI this is `counts` versus `two theta`. Thus we add this mapping:

```
two theta of detector elements = /NXentry/NXdata/polar angle
counts for each detector element = /NXentry/NXdata/data
```

The full mapping of WONI data into NeXus is documented in the next table:

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

Looking at this table, one might get concerned that the two theta and counts data is stored in two places and thus duplicated. Stop worrying, this problem is solved at the NeXus API level. Typically `NXdata` will only hold links to the corresponding data items in `/NXentry/NXinstrument/NXdetector`.

In this step problems might occur. The first is that the base class definitions contain a bewildering number of parameters. This is on purpose: the base classes serve as dictionaries which define names for most things which possibly can occur. You do not have to give all that information. Keep it simple and only require data that is needed for typical data analysis for this type of application.

Another problem which can occur is that you require to store information for which there is no name in one of the existing base classes or you have a new instrument component for which there is no base class altogether. New fields and base classes can be introduced if necessary.

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

Step 3: *Describe* this map in a NXDL file

This is even easier. Some XML editing is necessary. Fire up your XML editor of choice and open a file. If your XML editor supports XML schema while editing XML, it is worth to load `nxdl.xsd`. Now your XML editor can help you to create a proper NXDL file. As always, the start is an empty template file. This looks like the XML code below.

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

NXDL template file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
# NeXus - Neutron and X-ray Common Data Format
#
# Copyright (C) 2008-2026 NeXus International Advisory Committee (NIAC)
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 3 of the License, or (at your option) any later version.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
# Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public
# License along with this library; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# For further information, see https://www.nexusformat.org/
-->
<definition name="NX__template__" extends="NXobject" type="group"
    category="application"
    xmlns="http://definition.nexusformat.org/nxdl/3.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
    >
    <doc>template for a NXDL application definition</doc>
</definition>
```

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

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

```
<group type="NXdata">

</group>
```

The type is the actual NeXus base class this group belongs to. Optionally a `name` attribute may be given (default is `data`).

Next, one needs to include data items, too. The XML for such a data item looks similar to this:

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

The meaning of the `name` attribute is intuitive, the `type` can be looked up in the relevant base class definition. A `field` definition can optionally contain a `doc` element which contains a description of the data item. The `dimensions` entry specifies the dimensions of the data set. The `size` attribute in the dimensions tag sets the rank of the data, in this example: `rank="1"`. In the `dimensions` group there must be *rank* `dim` fields. Each `dim` tag holds two attributes: `index` determines to which dimension this tag belongs, the `1` means the first dimension. The `value` attribute then describes the size of the dimension. These can be plain integers, variables, such as in the example `ndet` or even expressions like `tof+1`.

Thus a NXDL file can be constructed. The full NXDL file for the WONI example is given in *Full listing of the WONI Application Definition*. Clever readers may have noticed the strong similarity between our working example `NXwoni` and `NXmonopd` since they are essentially identical. Give yourselves a cookie if you spotted this.

Step 4: *Standardize* with the NIAC

Basically you are done. Your first application definition for NeXus is constructed. In order to make your work a standard for that particular application type, some more steps are required:

- Send your application definition to the NIAC for review
- Correct your definition per the comments of the NIAC
- Cure and use the definition to verify it is ready
- After a final review, it becomes the standard

The NIAC must review an application definition before it is accepted as a standard. The 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. No specific time limit is required, but generally the standard should be well vetted by your community and the NIAC.

Full listing of the WONI Application Definition

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="nxdlformat.xsl" ?>
<!--
# NeXus - Neutron and X-ray Common Data Format
#
# Copyright (C) 2008-2026 NeXus International Advisory Committee (NIAC)
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License as published by the Free Software Foundation; either
# version 3 of the License, or (at your option) any later version.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
# Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public
# License along with this library; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# For further information, see http://www.nexusformat.org
-->
<definition name="NXmonopd" extends="NXobject" type="group"
    category="application"
    xmlns="http://definition.nexusformat.org/nxdl/3.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
    >
    <symbols>
        <doc>
            The symbol(s) listed here will be used below to coordinate datasets with the same shape.
        </doc>
        <symbol name="i">
            <doc>i is the number of wavelengths</doc>
        </symbol>
        <symbol name="nDet">
            <doc>Number of detectors</doc>
        </symbol>
    </symbols>
    <doc>
        Monochromatic Neutron and X-Ray Powder diffractometer

        Instrument
        definition for a powder diffractometer at a monochromatic neutron
        or X-ray beam. This is both suited for a powder diffractometer
        with a single detector or a powder diffractometer with a position
        sensitive detector.
    </doc>
    <group type="NXentry">
```

(continues on next page)

(continued from previous page)

```
        <field name="title"/>
        <field name="start_time" type="NX_DATE_TIME"/>
        <field name="definition">
            <doc> Official NeXus NXDL schema to which this file conforms </doc>
            <enumeration>
                <item value="NXmonopd"/>
            </enumeration>
        </field>
        <group type="NXinstrument">
            <group type="NXsource">
                <field name="type"/>
                <field name="name"/>
                <field name="probe">
                    <enumeration>
                        <item value="neutron"/>
                        <item value="x-ray"/>
                        <item value="electron"/>
                    </enumeration>
                </field>
            </group>
            <group type="NXcrystal">
                <field name="wavelength" type="NX_FLOAT" units="NX_WAVELENGTH">
                    <doc>Optimum diffracted wavelength</doc>
                    <dimensions rank="1">
                        <dim index="1" value="i"/>
                    </dimensions>
                </field>
            </group>
            <group type="NXdetector">
                <field name="polar_angle" type="NX_FLOAT" axis="1">
                  <dimensions rank="1">
                    <dim index="1" value="nDet" />
                  </dimensions>
                </field>
                <field name="data" type="NX_INT" signal="1">
                  <doc>
                    detector signal (usually counts) are already
                    corrected for detector efficiency
                  </doc>
                  <dimensions rank="1">
                    <dim index="1" value="nDet" />
                  </dimensions>
                </field>
            </group>
        </group>
        <group type="NXsample">
            <field name="name">
                <doc>Descriptive name of sample</doc>
            </field>
            <field name="rotation_angle" type="NX_FLOAT" units="NX_ANGLE">
                <doc>
                    Optional rotation angle for the case when the powder diagram
```

(continues on next page)

(continued from previous page)

```
                    has been obtained through an omega-2theta scan like from a
                    traditional single detector powder diffractometer
                </doc>
            </field>
        </group>
        <group type="NXmonitor">
            <field name="mode">
                <doc>
                    Count to a preset value based on either clock time (timer)
                    or received monitor counts (monitor).
                </doc>
                <enumeration>
                    <item value="monitor"/>
                    <item value="timer"/>
                </enumeration>
            </field>
            <field name="preset" type="NX_FLOAT">
                <doc>preset value for time or monitor</doc>
            </field>
            <field name="integral" type="NX_FLOAT" units="NX_ANY">
                <doc>Total integral monitor counts</doc>
            </field>
        </group>
        <group type="NXdata">
            <link name="polar_angle" target="/NXentry/NXinstrument/NXdetector/polar_angle
↪">
                <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
            </link>
            <link name="data" target="/NXentry/NXinstrument/NXdetector/data">
                <doc>Link to data in /NXentry/NXinstrument/NXdetector</doc>
            </link>
        </group>
    </group>
</definition>
```

Using an Application Definition

The application definition is like an interface for your data file. In practice files will contain far more information. For this, the extendable capability of NeXus comes in handy. More data can be added, and upstream software relying on the interface defined by the application definition can still retrieve the necessary information without any changes to their code.

NeXus application definitions only standardize classes. You are free to decide upon names of groups, subject to them matching regular expression for NeXus name attributes (see the *regular expression pattern for NXDL group and field names* in the *Naming Conventions* section). Note the length limit of 63 characters imposed by HDF5. Please use sensible, descriptive names and separate multi worded names with underscores.

Something most people wish to add is more metadata, for example in order to index files into a database of some sort. Go ahead, do so, if applicable, scan the NeXus base classes for standardized names. For metadata, consider to use the `NXarchive` definition. In this context, it is worth to mention that a practical NeXus file might adhere to more then one application definition. For example, WONI data files may adhere to both the `NXmonopd` and `NXarchive` definitions. The first for data analysis, the second for indexing into the database.

Often, instrument scientists want to store the complete state of their instrument in data files in order to be able to find out what went wrong if the data is unsatisfactory. Go ahead, do so, please use names from the NeXus base classes.

Site policy might require you to store the names of all your bosses up to the current head of state in data files. Go ahead, add as many `NXuser` classes as required to store that information. Knock yourselves silly over this.

Your Scientific Accounting Department (SAD) may ask of you the preposterous; to store billing information into data files. Go ahead, do so if your judgment allows. Just do not expect the NIAC to provide base classes for this and do not use the prefix NX for your classes.

In most cases, NeXus files will just have one `NXentry` class group. But it may be required to store multiple related data sets of the results of data analysis into the same data file. In this case create more entries. Each entry should be interpretable standalone, i.e. contain all the information of a complete `NXentry` class. Please keep in mind that groups or data items which stay constant across entries can always be linked to save space. Application definitions describe only what is included within an `NXentry` and so have no power to enforce any particular usage of `NXentry` groups. However, documentation within and accompanying an application definition can provide guidance and recommendations on situations where the use of multiple `NXentry` groups would be appropriate.

1.3.4 Processed Data

Data reduction and analysis programs are encouraged to store their results in NeXus data files, which may be the same file that contains the raw data. It is recommended to document the actions taken to generate the processed data in a *NXprocess* group, which has fields to store the name of the program used, its version number, and the date when it was run. If there are multiple processes recorded in the file, the group should also contain a sequence index to specify the order in which they were run. NXprocess groups can also contain one or more *NXparameters* groups to store the parameters used by the program as well as one or more *NXdata* groups that contain the results of the process. This has the advantage of encapsulating all the information required to preserve the provenance of the processed data in a single group. However, it is also acceptable to store the resulting data in a NXdata group at the same level as the NXprocess group in the NeXus hierarchy.

1.4 Strategies for storing information in NeXus data files

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

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

It is intended for this chapter to grow, with the addition of different use cases as they are presented for suggestions.

1.4.1 Strategies: The simplest case(s)

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

Step scan with two or more data columns

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

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

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

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

1.4.2 Strategies: The wavelength

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

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

Partial data file structure for *canonical location to store wavelength*

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

More recently, this location makes more sense to many:

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

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

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

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

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

1.4.3 Strategies: Time-stamped data

How should I store time-stamped data?

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

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

Both *NXlog* and *NXevent_data* have additional support for storing time-stamped data in the form of cues; cues can be used to place markers in the data that allow one to quickly look up coarse time ranges of interest. This coarse range of data can then be manually trimmed to be more selective, if required. The application writing the NeXus file is responsible for writing cues and when they are written. For example, the cue could be written every 10 seconds, every pulse, every 100 datapoints and so on.

Let's consider the case where NXlog is being used to store sample temperature data that has been sampled once every three seconds. The application that wrote the data has added cues every 20 seconds. Pictorially, this may look something like this:

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

In the NeXus Features repository, the feature ECB064453EDB096D shows example code that uses cues to select time-stamped data.

1.4.4 Strategies: The next case

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

1.5 Verification and validation of files

The intent of verification and validation of files is to ensure, in an unbiased way, that a given file conforms to the relevant specifications. Validation does not check that the data content of the file is sensible; this requires scientific interpretation based on the technique.

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

1.5.1 nxvalidate

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

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

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

1.5.2 punx

Python Utilities for NeXus HDF5 files

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

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

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

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

1.6 Frequently Asked Questions

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

1. Is it Nexus, NeXus or NeXuS?

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

2. How many facilities use NeXus?

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

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

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

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

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

5. Why are the NeXus classes so complicated? I'll never store all that information

   The NeXus classes are essentially glossaries of terms. If you need to store a piece of information, consult the class definitions to see if it has been defined. If so, use it. It is not compulsory to include every item that has been defined in the base class if it is not relevant to your experiment. On the other hand, a NeXus application definition lists a smaller set of compulsory items that should allow other researchers or software to analyze your data. You should really follow the application definition that corresponds to your experiment to take full advantage of NeXus.

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

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

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

7. **I want to contribute an application definition.**
   How do I go about it?

   Read the NXDL Tutorial in *Creating a NXDL Specification* and have a try. You can ask for help on the *mailing lists*. Once you have a definition that is working well for at least your case, you can submit it to the NIAC for acceptance as a standard. The procedures for acceptance are defined in the NIAC constitution.[1]

8. What is the purpose of `NXdata`?

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

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

9. How do I identify the plottable data?

   See the section: *Find the plottable data*.

10. Why aren't `NXsample` and `NXmonitor` groups stored in the `NXinstrument` group?

    A NeXus file can contain a number of `NXentry` groups, which may represent different scans in an experiment, or sample and calibration runs, etc. In many cases, though by no means all, the instrument has the same configuration so that it would be possible to save space by storing the `NXinstrument` group once and using multiple links in the remaining `NXentry` groups. It is assumed that the sample and monitor information would be more likely to change from run to run, and so should be stored at the top level.

11. Can I use a NXDL specification to parse a NeXus data file?

    This should be possible as there is nothing in the NeXus specifications to prevent this but it is not implemented in NAPI. You would need to implement it for yourself.

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

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

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

    > **Note**
    >
    > NeXus uses HDF5 hard links
    >
    > In HDF, a hard link points to a data object. A soft link points to a directory entry. Since NeXus uses hard links, there is no need to distinguish between two (or more) directory entries that point to the same data.

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

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

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

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

    See the *Strategies: The wavelength* section.

16. Where do I store metadata about my experiment?

    See the *Where to Store Metadata* section.

17. What file extension should I use when writing a NeXus data file?

    Any extension is permitted. Common extensions are *.h5*, *.hdf*, *.hdf5*, and *.nxs* while others are possible. See the many examples in the NeXus exampledata repository. (https://github.com/nexusformat/exampledata)

18. Can instances of classes inside definitions require new fields that were previously optional?

    Yes. That is one of the motivations to have application definitions. By default, all content in an application definition is required.

    For example, the `radiation` field in *NXcanSAS* requires 1 (and only 1) instance.

19. Can instances of classes inside definitions make optional new fields that were previously not mentioned?

    Yes. To make it optional, set attribute `minOccurs="0"`.

    For example, see the `Idev` field in *NXcanSAS*.

20. Can instances of classes inside definitions require new fields that were previously not mentioned?

    Yes.

    For example, see the `qx` field in *NXiqproc*.

21. Can we view the process of defining classes within an application definition as defining a subclass of the original class? That is, all instances of the class within the definition are valid instances of the original class, but not vice-versa?

    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 `NXdirecttof` extends the `NXtofraw` application definition:

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

    Most NXDL files extend the *NXobject* base class (the base object of NeXus).

CHAPTER

TWO

EXAMPLES OF WRITING AND READING NEXUS DATA FILES

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

2.1 Code Examples in Various Languages

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

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

2.1.1 HDF5 in C with `libhdf5`

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

Writing a simple NeXus file using native HDF5 commands in C

Note

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

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

(continues on next page)

(continued from previous page)

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

static void write_string_attr(hid_t hid, const char* name, const char* value)
{
  /* HDF-5 handles */
  hid_t atts, atttype, attid;

  atts = H5Screate(H5S_SCALAR);
  atttype = H5Tcopy(H5T_C_S1);
  H5Tset_size(atttype, strlen(value));
  attid = H5Acreate(hid,name, atttype, atts, H5P_DEFAULT, H5P_DEFAULT);
  H5Awrite(attid, atttype, value);
  H5Sclose(atts);
  H5Tclose(atttype);
  H5Aclose(attid);
}

static void write_int_attr(hid_t hid, const char* name, int value)
{
  /* HDF-5 handles */
  hid_t atts, atttype, attid;

  atts = H5Screate(H5S_SCALAR);
  atttype = H5Tcopy(H5T_NATIVE_INT);
  H5Tset_size(atttype,1);
  attid = H5Acreate(hid,name, atttype, atts, H5P_DEFAULT, H5P_DEFAULT);
  H5Awrite(attid, atttype, &value);
  H5Sclose(atts);
  H5Tclose(atttype);
  H5Aclose(attid);
}

#define LENGTH 400
int main(int argc, char *argv[])
```

(continues on next page)

(continued from previous page)

```
{
  float two_theta[LENGTH];
  int counts[LENGTH], i, rank;

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


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



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


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

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

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

(continues on next page)

(continued from previous page)

```
  write_int_attr(gid, "two_theta_indices", 0);

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

  H5Dclose(dataid);

  /*
   * store the two_theta dataset
   */
  dataspace = H5Screate_simple(rank,dim,maxdim);
  datatype = H5Tcopy(H5T_NATIVE_FLOAT);
  dataprop = H5Pcreate(H5P_DATASET_CREATE);
  dataid = H5Dcreate(gid,"two_theta",datatype,dataspace,H5P_DEFAULT,dataprop,H5P_
↪DEFAULT);
  H5Dwrite(dataid, datatype, H5S_ALL, H5S_ALL, H5P_DEFAULT, two_theta);
  H5Sclose(dataspace);
  H5Tclose(datatype);
  H5Pclose(dataprop);

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

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

  H5Dclose(dataid);

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

  /*
```

(continues on next page)

(continued from previous page)

```
   * close the file
   */
  H5Fclose(fid);
}
```

Reading a simple NeXus file using native HDF5 commands in C

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

int main(int argc, char *argv[])
{
  float *two_theta = NULL;
  int *counts = NULL,  rank, i;
  hid_t fid, dataid, fapl;
  hsize_t *dim = NULL;
  hid_t dataspace, memdataspace;

  /*
   * Open file, thereby enforcing proper file close
   * semantics
   */
  fapl = H5Pcreate(H5P_FILE_ACCESS);
  H5Pset_fclose_degree(fapl,H5F_CLOSE_STRONG);
  fid = H5Fopen("NXfile.h5", H5F_ACC_RDONLY,fapl);
  H5Pclose(fapl);

  /*
   * open and read the counts dataset
   */
  dataid = H5Dopen(fid,"/scan/data/counts",H5P_DEFAULT);
  dataspace = H5Dget_space(dataid);
  rank = H5Sget_simple_extent_ndims(dataspace);
  dim = malloc(rank*sizeof(hsize_t));
  H5Sget_simple_extent_dims(dataspace, dim, NULL);
  counts = malloc(dim[0]*sizeof(int));
  memdataspace = H5Tcopy(H5T_NATIVE_INT32);
  H5Dread(dataid,memdataspace,H5S_ALL, H5S_ALL,H5P_DEFAULT, counts);
  H5Dclose(dataid);
  H5Sclose(dataspace);
```

(continues on next page)

(continued from previous page)

```
  H5Tclose(memdataspace);

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



  H5Fclose(fid);

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

}
```

2.1.2 HDF5 in Python

One way to gain a quick familiarity with NeXus is to start working with some data. For at least the first few examples in this section, we have a simple two-column set of 1-D data, collected as part of a series of alignment scans by the Advanced Photon Source USAXS instrument during the time it was stationed at beam line 32ID. We will show how to read and write this data in Python using both the `nexusformat`[1] and `h5py`[2] packages. The `nexusformat` package provides a simplified syntax for reading and writing NeXus-compliant files by automatically handling some of the features required by the NeXus standard, such as the attributes that define group classes and plottable data. However, it also uses the `h5py` package to read/write the HDF5 files on disk. We provide tabbed examples showing how to produce equivalent files either using `nexusformat` or directly in `h5py`.

The actual data to be written was extracted (elsewhere) from a `spec`[3] data file and read as a text block from a file by the Python source code. Our examples will start with the simplest case and add only mild complexity with each new case since these examples are meant for those who are unfamiliar with NeXus.

[1] *nexusformat*: https://nexpy.github.io/nexpy/
[2] *h5py*: https://www.h5py.org/
[3] *SPEC*: http://certif.com/spec.html

Code examples

Getting started

Write a NeXus HDF5 File

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

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

Note

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

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

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

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

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

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

nexusformat

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

from pathlib import Path
from re import X
import numpy

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

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

# load data from two column format
```

(continues on next page)

(continued from previous page)

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

# create the HDF5 NeXus file
with nxopen(fileName, "w") as f:

    # create the NXentry group
    f["entry"] = NXentry()
    f["entry/title"] = "1-D scan of I00 v. mr"

    # create the NXdata group
    x = NXfield(mr_arr, name="mr", units="degrees", long_name="USAXS mr (degrees)")
    y = NXfield(i00_arr, name="I00", units="counts",
                long_name="USAXS I00 (counts)")
    f["entry/mr_scan"] = NXdata(y, x)

print("wrote file:", fileName)
```

h5py

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

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

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

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

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

(continues on next page)

(continued from previous page)

```

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

    # create the NXentry group
    nxdata = nxentry.create_group("mr_scan")
    nxdata.attrs["NX_class"] = "NXdata"
    nxdata.attrs["signal"] = "I00"  # Y axis of default plot
    nxdata.attrs["axes"] = "mr"  # X axis of default plot
    nxdata.attrs["mr_indices"] = [
        0,
    ]  # use "mr" as the first dimension of I00

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

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

print("wrote file:", fileName)
```

Read a NeXus HDF5 File

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

Note

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

simple_example_basic_read.py: Read a NeXus HDF5 file using Python

nexusformat

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

from nexusformat.nexus import nxopen

fileName = "simple_example_basic.nexus.hdf5"
with nxopen(fileName) as f:
    print(f.tree)
```

h5py

```python
#!/usr/bin/env python
"""Reads NeXus HDF5 files using h5py and prints the contents"""

import h5py  # HDF5 support

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

Output from `simple_example_basic_read.py` is shown next.

Output from `simple_example_basic_read.py`

```
file_name: simple_example_basic.nexus.hdf5
file_time: 2010-10-18T17:17:04-0500
creator: simple_example_basic_write.py
HDF5_Version: 1.8.5
NeXus_version: 4.3.0
h5py_version: 1.2.1
instrument: APS USAXS at 32ID-B
#   mr  I00
0   17.9261 1037
1   17.9259 1318
2   17.9258 1704
3   17.9256 2857
4   17.9254 4516
5   17.9252 9998
6   17.9251 23819
7   17.9249 31662
8   17.9247 40458
9   17.9246 49087
10  17.9244 56514
```

(continues on next page)

(continued from previous page)

```
11  17.9243 63499
12  17.9241 66802
13  17.9239 66863
14  17.9237 66599
15  17.9236 66206
16  17.9234 65747
17  17.9232 65250
18  17.9231 64129
19  17.9229 63044
20  17.9228 60796
21  17.9226 56795
22  17.9224 51550
23  17.9222 43710
24  17.9221 29315
25  17.9219 19782
26  17.9217 12992
27  17.9216 6622
28  17.9214 4198
29  17.9213 2248
30  17.9211 1321
```

downloads

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

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

Write a NeXus HDF5 file

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

In the *above figure*, the data file (`simple_example_write1_h5py.hdf5`) contains a hierarchy of items, starting with an `NXentry` named `entry`. (The full HDF5 path reference, `/entry` in this case, is shown to the right of each component

Fig. 1: Simple Example

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

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

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

from pathlib import Path

import numpy

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

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

with nxopen("simple_example_write1.hdf5", "w") as f:  # create the NeXus file
    f["Scan"] = NXentry()
```

(continues on next page)

(continued from previous page)

```
    tth = NXfield(tthData, name="two_theta", units="degrees")
    counts = NXfield(countsData, name="counts", units="counts")
    f["Scan/data"] = NXdata(counts, tth)
```

h5py

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

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

from pathlib import Path
import h5py
import numpy

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

with h5py.File("simple_example_write1.hdf5", "w") as f:  # create the HDF5 NeXus file
    # since this is a simple example, no attributes are used at this point

    nxentry = f.create_group("Scan")
    nxentry.attrs["NX_class"] = "NXentry"

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

    tth = nxdata.create_dataset("two_theta", data=tthData)
    tth.attrs["units"] = "degrees"

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

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

Note

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

Since the output of `h5dump` is verbose (see the *Downloads* section below), the *punx tree* tool[1] was used to print out the structure of HDF5 data files. This tool provides a simplified view of the NeXus file. Here is the output:

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

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

downloads

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

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

Write a NeXus HDF5 file with plottable data

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

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

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

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

nexusformat

```
#!/usr/bin/env python
"""
Writes a simple NeXus HDF5 file using h5py with links
according to the example from Figure 2.1 in the Design chapter
"""

from pathlib import Path

import numpy

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

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

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

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

    f["entry/data"] = NXdata(NXlink("/entry/instrument/detector/counts"),
                             NXlink("/entry/instrument/detector/two_theta"))
    f["entry/data"].set_default()
```

h5py

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

(continues on next page)

(continued from previous page)

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

from pathlib import Path
import h5py
import numpy

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

with h5py.File("simple_example_write2.hdf5", "w") as f:  # create the HDF5␣
↪NeXus file
    f.attrs["default"] = "entry"

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

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

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

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

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

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

    source_addr = "/entry/instrument/detector/counts"  # existing data
    target_addr = "counts"  # new location
    ds_counts.attrs["target"] = source_addr  # a NeXus API convention for links
    nxdata[target_addr] = f[source_addr]  # hard link
```

(continues on next page)

(continued from previous page)

```
    # nxdata._id.link(source_addr, target_addr, h5py.h5g.LINK_HARD)
```

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

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

A bit of explanation is necessary at this point. The data is not stored in either HDF5 group directly. Instead, HDF5 creates a `DATA` storage element in the file and posts a reference to that `DATA` storage element as needed. An HDF5 *hard link* requests another reference to that same `DATA` storage element. The `h5dump` tool describes in full that `DATA` storage element the first time (alphabetically) it is called. In our case, that is within the `NXdata` group. The next time it is called, within the `NXdetector` group, `h5dump` reports that a hard link has been made and shows the HDF5 path to the description.

NeXus recognizes this behavior of the HDF5 library and adds an additional structure when building hard links, the `target` attribute, to preserve the original location of the data. Not that it actually matters. the *punx tree* tool knows about the additional NeXus `target` attribute and shows the data to appear in its original location, in the `NXdetector` group.

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

downloads

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

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

Write a NeXuS HDF5 File with links to external data

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

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

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

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

file: external_angles.hdf5

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

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

file: external_counts.hdf5

The data in the file `external_angles.hdf5` might be referenced from another HDF5 file (such as `external_counts.hdf5`) by an HDF5 external link.[1] Here is an example of the structure:

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

[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

file: external_master.hdf5

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

> **Note**
>
> It is necessary for all these files to be located together in the same directory for the HDF5 external file links to work properly.`

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

```
data:NXdata
  @signal = counts
  @axes = "two_theta"
  @two_theta_indices = 0
```

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

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

source code: external_example_write.py

Here is the complete code of a Python program, using `h5py` to write a NeXus-compliant HDF5 file with links to data in other HDF5 files.

external_example_write.py: Write using HDF5 external links

nexusformat

```python
#!/usr/bin/env python
"""
Writes a NeXus HDF5 file using h5py with links to data in other HDF5 files.

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

from pathlib import Path

import h5py
```

(continues on next page)

(continued from previous page)

```python
import numpy

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

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

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

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

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

# put the detector counts in an external HDF5 data file
# with *incomplete* NeXus structure (no NXdata group)
with nxopen(FILE_HDF5_COUNTS, "w") as f:
    f["entry"] = NXentry()
    f["entry/instrument"] = NXinstrument()
    f["entry/instrument/detector"] = NXdetector()
    f["entry/instrument/detector/counts"] = NXfield(countsData, units="counts")
    f["entry/instrument/detector/two_theta"] = NXlink("/angles",
                                                      FILE_HDF5_ANGLES)

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

h5py

```python
#!/usr/bin/env python
"""
Writes a NeXus HDF5 file using h5py with links to data in other HDF5 files.

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

from pathlib import Path
```

(continues on next page)

(continued from previous page)

```python
import h5py
import numpy

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

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

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

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

# put the detector counts in an external HDF5 data file
# with *incomplete* NeXus structure (no NXdata group)
with h5py.File(FILE_HDF5_COUNTS, "w") as f:
    nxentry = f.create_group("entry")
    nxentry.attrs["NX_class"] = "NXentry"
    nxinstrument = nxentry.create_group("instrument")
    nxinstrument.attrs["NX_class"] = "NXinstrument"
    nxdetector = nxinstrument.create_group("detector")
    nxdetector.attrs["NX_class"] = "NXdetector"
    ds = nxdetector.create_dataset("counts", data=countsData)
    ds.attrs["units"] = "counts"
    # link the "two_theta" data stored in separate file
    local_addr = nxdetector.name + "/two_theta"
    f[local_addr] = h5py.ExternalLink(FILE_HDF5_ANGLES, "/angles")

# create a master NeXus HDF5 file
with h5py.File(FILE_HDF5_MASTER, "w") as f:
    f.attrs["default"] = "entry"
    nxentry = f.create_group("entry")
    nxentry.attrs["NX_class"] = "NXentry"
    nxentry.attrs["default"] = "data"
    nxdata = nxentry.create_group("data")
    nxdata.attrs["NX_class"] = "NXdata"

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

    # link in the axes data
    local_addr = "/entry/data/two_theta"
```

(continues on next page)

(continued from previous page)

```
    f[local_addr] = h5py.ExternalLink(FILE_HDF5_ANGLES, "/angles")
    nxdata.attrs["axes"] = "two_theta"
    nxdata.attrs["two_theta_indices"] = [
        0,
    ]

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

downloads

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

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

Find plottable data in a NeXus HDF5 file

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

reader_attributes_trail.py: Read a NeXus HDF5 file using Python

nexusformat

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

filename = str(
    Path(__file__).absolute().parent.parent
    / "simple_example_basic"
    / "simple_example_basic.nexus.hdf5"
)
with nxopen(filename) as f:
    # find the default NXdata group
    nx_data = f.get_default()
```

(continues on next page)

(continued from previous page)

```
    signal = nx_data.nxsignal
    axes = nx_data.nxaxes[0]

nx_data.plot() # plot the data using Matplotlib

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

h5py

```
from pathlib import Path
import h5py

filename = str(
    Path(__file__).absolute().parent.parent
    / "simple_example_basic"
    / "simple_example_basic.nexus.hdf5"
)
with h5py.File(filename, "r") as nx:
    # find the default NXentry group
    nx_entry = nx[nx.attrs["default"]]
    # find the default NXdata group
    nx_data = nx_entry[nx_entry.attrs["default"]]
    # find the signal field
    signal = nx_data[nx_data.attrs["signal"]]
    # find the axes field(s)
    attr_axes = nx_data.attrs["axes"]
    if isinstance(attr_axes, (set, tuple, list)):
        #  but check that attr_axes only describes 1-D data
        if len(attr_axes) == 1:
            attr_axes = attr_axes[0]
        else:
            raise ValueError(f"expected 1-D data but @axes={attr_axes}")
    axes = nx_data[attr_axes]

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

Output from `reader_attributes_trail.py` is shown next.

Output from `reader_attributes_trail.py`

```
file: simple_example_basic.nexus.hdf5
signal: /entry/mr_scan/I00
axes: /entry/mr_scan/mr
/entry/mr_scan/mr /entry/mr_scan/I00
17.92608 1037
17.92591 1318
17.92575 1704
17.92558 2857
17.92541 4516
17.92525 9998
17.92508 23819
17.92491 31662
17.92475 40458
17.92458 49087
17.92441 56514
17.92425 63499
17.92408 66802
17.92391 66863
17.92375 66599
17.92358 66206
17.92341 65747
17.92325 65250
17.92308 64129
17.92291 63044
17.92275 60796
17.92258 56795
17.92241 51550
17.92225 43710
17.92208 29315
17.92191 19782
17.92175 12992
17.92158 6622
17.92141 4198
17.92125 2248
17.92108 1321
```

downloads

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

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

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

Example data used

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

Fig. 3: data structure of the simple example

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

Fig. 4: plot of the simple example data

Simple example values

```
17.92608    1037
17.92591    1318
17.92575    1704
17.92558    2857
17.92541    4516
17.92525    9998
17.92508    23819
17.92491    31662
17.92475    40458
17.92458    49087
17.92441    56514
17.92425    63499
17.92408    66802
17.92391    66863
17.92375    66599
17.92358    66206
17.92341    65747
17.92325    65250
17.92308    64129
17.92291    63044
17.92275    60796
17.92258    56795
17.92241    51550
17.92225    43710
17.92208    29315
17.92191    19782
17.92175    12992
17.92158    6622
17.92141    4198
17.92125    2248
17.92108    1321
```

2.1.3 HDF5 in MATLAB

author
Paul Kienzle, NIST

Note

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

input.dat

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

```
17.92608    1037
17.92591    1318
17.92575    1704
17.92558    2857
17.92541    4516
17.92525    9998
17.92508    23819
17.92491    31662
17.92475    40458
17.92458    49087
17.92441    56514
17.92425    63499
17.92408    66802
17.92391    66863
17.92375    66599
17.92358    66206
17.92341    65747
17.92325    65250
17.92308    64129
17.92291    63044
17.92275    60796
17.92258    56795
17.92241    51550
17.92225    43710
17.92208    29315
17.92191    19782
17.92175    12992
17.92158    6622
17.92141    4198
17.92125    2248
17.92108    1321
```

writing data

basic_writer.m: Write a NeXus HDF5 file using Matlab

```
% Writes a NeXus HDF5 file using matlab

disp 'Write a NeXus HDF5 file'
filename = 'prj_test.nexus.hdf5';
timestamp = '2010-10-18T17:17:04-0500';

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

% clear out old file, if it exists
```

(continues on next page)

(continued from previous page)

```

delete(filename);

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

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

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

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

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


h5disp(filename);
```

reading data

basic_reader.m: Read a NeXus HDF5 file using Matlab

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

filename = 'prj_test.nexus.hdf5';
root = h5info(filename,'/');
attrs = root.Attributes;
for i = 1:length(attrs)
    fprintf('%s: %s\n', attrs(i).Name, attrs(i).Value);
```

(continues on next page)

(continued from previous page)

```
end
mr = h5read(filename,'/entry/mr_scan/mr');
i00 = h5read(filename, '/entry/mr_scan/I00');
fprintf('#\t%s\t%s\n','mr','I00');
for i = 1:length(mr)
    fprintf('%d\t%g\t%d\n', i, mr(i), i00(i));
end
```

writing data file with links

writer_2_1.m: Write a NeXus HDF5 file with links

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

filename = 'writer_2_1.hdf5';

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

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

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

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

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

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

% add NeXus metadata
h5writeatt(filename,'/','file_name',filename);
```

(continues on next page)

(continued from previous page)

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

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

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

```
function h5link(filename, from, to)
%H5LINK Create link to an HDF5 dataset.
%   H5LINK(FILENAME,SOURCE,TARGET) creates an HDF5 link from the
%   dataset at location SOURCE to a dataset at location TARGET.  All
%   intermediate groups in the path to target are created.
%
%   Example:  create a link from /hello/world to /goodbye/world
%      h5create('myfile.h5','/hello/world',[100 200]);
%      h5link('myfile.h5','/hello/world','/goodbye/world');
%      hgdisp('myfile.h5');
%
%   See also: h5create, h5read, h5write, h5info, h5disp

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

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

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

% open groups and create link
from_id = H5G.open(fid, from_path);
```

(continues on next page)

(continued from previous page)

```
to_id = H5G.open(fid, to_path);
H5L.create_hard(from_id, from_data, to_id, to_data, 'H5P_DEFAULT','H5P_DEFAULT');

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

Downloads

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

2.1.4 HDF5 in C with NAPI

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

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

NAPI C Example: write simple NeXus file

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

```c
#include "napi.h"

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

(continues on next page)

(continued from previous page)

```
 * Read in data using local routines to populate phi and counts
 *
 * for example you may create a getdata() function and call
 *
 *      getdata (n_t, t, n_p, phi, counts);
 */
/* Open output file and output global attributes */
    NXopen ("NXfile.nxs", NXACC_CREATE5, &file_id);
      NXputattr (file_id, "user_name", "Joe Bloggs", 10, NX_CHAR);
/* Open top-level NXentry group */
      NXmakegroup (file_id, "Entry1", "NXentry");
      NXopengroup (file_id, "Entry1", "NXentry");
/* Open NXdata group within NXentry group */
        NXmakegroup (file_id, "Data1", "NXdata");
        NXopengroup (file_id, "Data1", "NXdata");
/* Output time channels */
          NXmakedata (file_id, "time_of_flight", NX_FLOAT32, 1, &n_t);
          NXopendata (file_id, "time_of_flight");
            NXputdata (file_id, t);
            NXputattr (file_id, "units", "microseconds", 12, NX_CHAR);
          NXclosedata (file_id);
/* Output detector angles */
          NXmakedata (file_id, "polar_angle", NX_FLOAT32, 1, &n_p);
          NXopendata (file_id, "polar_angle");
            NXputdata (file_id, phi);
            NXputattr (file_id, "units", "degrees", 7, NX_CHAR);
          NXclosedata (file_id);
/* Output data */
          dims[0] = n_t;
          dims[1] = n_p;
          NXmakedata (file_id, "counts", NX_INT32, 2, dims);
          NXopendata (file_id, "counts");
            NXputdata (file_id, counts);
            i = 1;
            NXputattr (file_id, "signal", &i, 1, NX_INT32);
            NXputattr (file_id, "axes",  "polar_angle:time_of_flight", 26, NX_CHAR);
          NXclosedata (file_id);
/* Close NXentry and NXdata groups and close file */
        NXclosegroup (file_id);
      NXclosegroup (file_id);
    NXclose (&file_id);
    return;
}
```

2.1.5 HDF5 in Fortran with NAPI

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

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

NAPI F77 Example: write simple NeXus file

Note

The F77 interface is no longer being developed.

```
      program WRITEDATA

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

!Read in data using local routines
      call getdata (n_t, t, n_p, phi, counts)
!Open output file
      status = NXopen ('NXFILE.NXS', NXACC_CREATE, file_id)
        status = NXputcharattr
     +         (file_id, 'user', 'Joe Bloggs', 10, NX_CHAR)
!Open top-level NXentry group
        status = NXmakegroup (file_id, 'Entry1', 'NXentry')
        status = NXopengroup (file_id, 'Entry1', 'NXentry')
!Open NXdata group within NXentry group
          status = NXmakegroup (file_id, 'Data1', 'NXdata')
          status = NXopengroup (file_id, 'Data1', 'NXdata')
!Output time channels
            status = NXmakedata
     +         (file_id, 'time_of_flight', NX_FLOAT32, 1, n_t)
            status = NXopendata (file_id, 'time_of_flight')
              status = NXputdata (file_id, t)
              status = NXputcharattr
     +         (file_id, 'units', 'microseconds', 12, NX_CHAR)
            status = NXclosedata (file_id)
!Output detector angles
            status = NXmakedata (file_id, 'polar_angle', NX_FLOAT32, 1, n_p)
            status = NXopendata (file_id, 'polar_angle')
              status = NXputdata (file_id, phi)
              status = NXputcharattr (file_id, 'units', 'degrees', 7, NX_CHAR)
            status = NXclosedata (file_id)
!Output data
            dims(1) = n_t
            dims(2) = n_p
            status = NXmakedata (file_id, 'counts', NX_INT32, 2, dims)
```

(continues on next page)

(continued from previous page)

```
            status = NXopendata (file_id, 'counts')
              status = NXputdata (file_id, counts)
              status = NXputattr (file_id, 'signal', 1, 1, NX_INT32)
              status = NXputattr
     +          (file_id, 'axes', 'polar_angle:time_of_flight', 26, NX_CHAR)
            status = NXclosedata (file_id)
!Close NXdata and NXentry groups and close file
          status = NXclosegroup (file_id)
        status = NXclosegroup (file_id)
      status = NXclose (file_id)

      stop
      end
```

NAPI F90 Example: write simple NeXus file

Note

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

```
program WRITEDATA

   use NXUmodule

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

!Use local routines to allocate pointers and fill in data
   call getlocaldata (t, phi, counts)
!Open output file
   if (NXopen ("NXfile.nxs", NXACC_CREATE, file_id) /= NX_OK) stop
   if (NXUwriteglobals (file_id, user="Joe Bloggs") /= NX_OK) stop
!Set compression parameters
   if (NXUsetcompress (file_id, NX_COMP_LZW, 1000) /= NX_OK) stop
!Open top-level NXentry group
   if (NXUwritegroup (file_id, "Entry1", "NXentry") /= NX_OK) stop
   !Open NXdata group within NXentry group
      if (NXUwritegroup (file_id, "Data1", "NXdata") /= NX_OK) stop
   !Output time channels
         if (NXUwritedata (file_id, "time_of_flight", t, "microseconds") /= NX_OK) stop
   !Output detector angles
         if (NXUwritedata (file_id, "polar_angle", phi, "degrees") /= NX_OK) stop
   !Output data
         if (NXUwritedata (file_id, "counts", counts, "counts") /= NX_OK) stop
            if (NXputattr (file_id, "signal", 1) /= NX_OK) stop
            if (NXputattr (file_id, "axes", "polar_angle:time_of_flight") /= NX_OK) stop
```

(continues on next page)

(continued from previous page)

```
   !Close NXdata group
      if (NXclosegroup (file_id) /= NX_OK) stop
!Close NXentry group
   if (NXclosegroup (file_id) /= NX_OK) stop
!Close NeXus file
   if (NXclose (file_id) /= NX_OK) stop

end program WRITEDATA
```

2.1.6 HDF5 in Python with NAPI

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

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

NAPI Python Example: write simple NeXus file

```
#!/usr/bin/python

import sys
import nxs
import numpy

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

nf = nxs.open("simple3D.h5", "w5")

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

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

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

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

```

(continues on next page)

(continued from previous page)

```
nf.close()

exit
```

2.2 Visualization tools

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

2.2.1 Plot a NeXus HDF5 file with *NeXpy*

A NeXus HDF5 file with plottable data (see *Find plottable data in a NeXus HDF5 file*) can be plotted by `NeXpy`[1].

Fig. 5: plot the simple example using NeXpy

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.

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

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].

Fig. 6: plot the simple example using silx

2.2.3 View a NeXus HDF5 file with *punx tree*

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

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

(continues on next page)

[1] **silx view** : http://www.silx.org/doc/silx/latest/applications/view.html
[2] **silx** : http://www.silx.org/doc/silx/latest/
[1] **punx tree** : https://punx.readthedocs.io/en/latest/source_code/h5tree.html#how-to-use-h5tree
[2] **punx** : https://punx.readthedocs.io/

(continued from previous page)

```
            ]
            [
              [12, 13, 14, 15]
              [16, 17, 18, 19]
              [20, 21, 22, 23]
            ]
          ]
```

2.2.4 View a NeXus HDF5 file with *h5dump*

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

```
HDF5 "simple3D.h5" {
GROUP "/" {
   ATTRIBUTE "NeXus_version" {
      DATATYPE  H5T_STRING {
            STRSIZE 5;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
      DATASPACE  SCALAR
      DATA {
      (0): "4.1.0"
      }
   }
   ATTRIBUTE "file_name" {
      DATATYPE  H5T_STRING {
            STRSIZE 11;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
      DATASPACE  SCALAR
      DATA {
      (0): "simple3D.h5"
      }
   }
   ATTRIBUTE "HDF5_Version" {
      DATATYPE  H5T_STRING {
            STRSIZE 5;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
      DATASPACE  SCALAR
      DATA {
```

(continues on next page)

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

(continued from previous page)

```
      (0): "1.6.6"
      }
   }
   ATTRIBUTE "file_time" {
      DATATYPE  H5T_STRING {
            STRSIZE 24;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
      DATASPACE  SCALAR
      DATA {
      (0): "2011-11-18 17:26:27+0100"
      }
   }
   GROUP "entry" {
      ATTRIBUTE "NX_class" {
         DATATYPE  H5T_STRING {
               STRSIZE 7;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "NXentry"
         }
      }
      GROUP "data" {
         ATTRIBUTE "NX_class" {
            DATATYPE  H5T_STRING {
                  STRSIZE 6;
                  STRPAD H5T_STR_NULLTERM;
                  CSET H5T_CSET_ASCII;
                  CTYPE H5T_C_S1;
               }
            DATASPACE  SCALAR
            DATA {
            (0): "NXdata"
            }
         }
         DATASET "test" {
            DATATYPE  H5T_STD_I32LE
            DATASPACE  SIMPLE { ( 2, 3, 4 ) / ( 2, 3, 4 ) }
            DATA {
            (0,0,0): 0, 1, 2, 3,
            (0,1,0): 4, 5, 6, 7,
            (0,2,0): 8, 9, 10, 11,
            (1,0,0): 12, 13, 14, 15,
            (1,1,0): 16, 17, 18, 19,
            (1,2,0): 20, 21, 22, 23
            }
```

(continues on next page)

(continued from previous page)

```
            ATTRIBUTE "signal" {
               DATATYPE  H5T_STD_I32LE
               DATASPACE  SCALAR
               DATA {
               (0): 1
               }
            }
         }
      }
   }
}
}
```

2.3 Examples for Specific Instruments

Examples of working with data from specific instruments.

2.3.1 Viewing 2-D Data from LRMECS

The IPNS LRMECS instrument stored data in NeXus HDF4 data files. One such example is available from the repository of NeXus data file examples.[1] For this example, we will start with a conversion of that original data file into *HDF5* format.

format	file name
HDF4	lrcs3701.nxs
HDF5	lrcs3701.nx5

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

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`):

[1] LRMECS example data: https://github.com/nexusformat/exampledata/tree/master/IPNS/LRMECS

LRMECS `lrcs3701` data: `h5dump` output

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

Visualize Using *HDFview*

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

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

Note

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

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

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

LRMECS `lrcs3701` data: image

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

Visualize Using *IgorPro*

Another way to visualize this data is to use a commercial package for scientific data visualization and analysis. One such package is *IgorPro* from http://www.wavemetrics.com

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

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

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

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

> **Note**
>
> In this image, dark represents low intensity while white represents high intensity. The image has been rotated for easier representation in this manual.

LRMECS `lrcs3701` data: image

2.3.2 EPICS Area Detector Examples

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

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

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

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

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

HDF5 File Writing Plugin

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

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

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

configuration files

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

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

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

`attributes.xml`

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

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

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

```xml
<?xml version="1.0" standalone="no" ?>
<!-- Attributes -->
<Attributes
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "https://github.com/areaDetector/ADCore/blob/master/iocBoot/NDAttributes.xsd"
    >
    <Attribute name="AcquireTime"
               type="EPICS_PV"
               source="13SIM1:cam1:AcquireTime"
               dbrtype="DBR_NATIVE"
```

(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)

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

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

Note

ndattribute : item specified by an `<Attribute />` element in the attributes file.

layout.xml

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

```xml
<?xml version="1.0" standalone="no" ?>
<hdf5_layout>
  <group name="entry">
    <attribute name="NX_class" source="constant" value="NXentry" type="string"/>
    <group name="instrument">
      <attribute name="NX_class" source="constant" value="NXinstrument" type="string"/>
      <group name="detector">
        <attribute name="NX_class" source="constant" value="NXdetector" type="string"/>
        <dataset name="data" source="detector" det_default="true">
          <attribute name="NX_class" source="constant" value="SDS" type="string"/>
          <attribute name="signal" source="constant" value="1" type="int"/>
          <attribute name="target" source="constant" value="/entry/instrument/detector/data" type="string"/>
        </dataset>
        <group name="NDAttributes">
          <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
          <dataset name="ColorMode" source="ndattribute" ndattribute="ColorMode"/>
        </group>          <!-- end group NDAttribute -->
      </group>            <!-- end group detector -->
      <group name="NDAttributes" ndattr_default="true">
        <attribute name="NX_class" source="constant" value="NXcollection" type="string"/>
      </group>            <!-- end group NDAttribute (default) -->
      <group name="performance">
        <dataset name="timestamp" source="ndattribute"/>
      </group>            <!-- end group performance -->
    </group>              <!-- end group instrument -->
    <group name="data">
      <attribute name="NX_class" source="constant" value="NXdata" type="string"/>
      <hardlink name="data" target="/entry/instrument/detector/data"/>
      <!-- The "target" attribute in /entry/instrument/detector/data is used to
           tell Nexus utilities that this is a hardlink -->
    </group>              <!-- end group data -->
  </group>                <!-- end group entry -->
</hdf5_layout>
```

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

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

(continues on next page)

(continued from previous page)

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

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

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

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

Note

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

You can control when a value is written to the file, using `when=""` in the layout file. This can be set to one of these values: `OnFileOpen`, `OnFileClose`

Such as:

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

or:

```
<attribute
   name="exposure_s"
   source="ndattribute"
   ndattribute="AcquireTime"
   when="OnFileClose"/>
```

additional configuration

Additional configurations of the EPICS Area Detector and the HDF5 File Plugin are done using the EPICS screens (shown here using caQtDM[6]):

Additional configuration on the **ADBase** screen:

- Set *Image mode* to "Single"
- Set *Exposure time* as you wish

[5] NeXus manual: https://manual.nexusformat.org/
[6] caQtDM: http://epics.web.psi.ch/software/caqtdm/

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

- 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".
- 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
`attributes.xml`	NDAttributes.xsd
`layout.xml`	hdf5_xml_layout_schema.xsd
`nxdl.xsd`	XMLSchema.xsd
`nxdlTypes.xsd`	XMLSchema.xsd
any NXDL file	`nxdl.xsd`

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.

```python
import numpy as np
import h5py
import datetime

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

    Parameters
    ----------
    fname : str
        name of the file (relative or absolute) to be written
    image : numpy array
        the image data
    md : dictionary
        key: value where value is something that can be written by h5py
             (such as str, int, float, numpy array, ...)
    """
    nexus = h5py.File(fname, "w")
    nexus.attrs["filename"] = fname
    nexus.attrs["file_time"] = datetime.datetime.now().astimezone().isoformat()
    nexus.attrs["creator"] = "write_nexus_file()"
    nexus.attrs["H5PY_VERSION"] = h5py.__version__
```

(continues on next page)

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

(continued from previous page)

```python

    # /entry
    nxentry = nexus.create_group("entry")
    nxentry.attrs["NX_class"] = "NXentry"
    nexus.attrs["default"] = nxentry.name

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

    # /entry/instrument/detector
    nxdetector = nxinstrument.create_group("detector")
    nxdetector.attrs["NX_class"] = "NXdetector"

    # /entry/instrument/detector/image
    ds = nxdetector.create_dataset("image", data=image, compression="gzip")
    ds.attrs["units"] = "counts"
    ds.attrs["target"] = "/entry/instrument/detector/image"

    # /entry/data
    nxdata = nxentry.create_group("data")
    nxdata.attrs["NX_class"] = "NXdata"
    nxentry.attrs["default"] = nxdata.name

    # /entry/data/data --> /entry/instrument/detector/image
    nxdata["data"] = nexus["/entry/instrument/detector/image"]
    nxdata.attrs["signal"] = "data"

    if len(md) > 0:
        # /entry/instrument/metadata (optional, for metadata)
        metadata = nxinstrument.create_group("metadata")
        metadata.attrs["NX_class"] = "NXcollection"
        for k, v in md.items():
            try:
                metadata.create_dataset(k, data=v)
            except Exception:
                metadata.create_dataset(k, data=str(v))

    nexus.close()


if __name__ == "__main__":
    """demonstrate how to use this code"""
    import epics
    prefix = "13SIM1:"
    img = epics.caget(prefix+"image1:ArrayData")
    size_x = epics.caget(prefix+"cam1:ArraySizeX_RBV")
    size_y = epics.caget(prefix+"cam1:ArraySizeY_RBV")
    # edit the full image for just the binned data
    img = img[:size_x*size_y].reshape((size_x, size_y))

    extra_information = dict(
```

(continues on next page)

(continued from previous page)

```
        unique_id = epics.caget(prefix+"image1:UniqueId_RBV"),
        size_x = size_x,
        size_y = size_y,
        detector_state = epics.caget(prefix+"cam1:DetectorState_RBV"),
        bitcoin_value="15000",
    )
    write_nexus_file("example.h5", img, md=extra_information)
```

The output from that code is given in the example.h5 file. It has this tree structure:

```
example.h5 : NeXus data file
  @H5PY_VERSION = "3.6.0"
  @creator = "write_nexus_file()"
  @default = "entry"
  @file_time = "2022-03-07 14:34:04.418733"
  @filename = "example.h5"
  entry:NXentry
    @NX_class = "NXentry"
    @default = "data"
    data:NXdata
      @NX_class = "NXdata"
      @signal = "data"
      data --> /entry/instrument/detector/image
    instrument:NXinstrument
      @NX_class = "NXinstrument"
      detector:NXdetector
        @NX_class = "NXdetector"
        image:NX_UINT8[1024,1024] = __array
          __array = [
              [76, 77, 78, '...', 75]
              [77, 78, 79, '...', 76]
              [78, 79, 80, '...', 77]
              ...
              [75, 76, 77, '...', 74]
            ]
          @target = "/entry/instrument/detector/image"
          @units = "counts"
      metadata:NXcollection
        @NX_class = "NXcollection"
        bitcoin_value:NX_CHAR = b'15000'
        detector_state:NX_INT64[] =
        size_x:NX_INT64[] =
        size_y:NX_INT64[] =
        unique_id:NX_INT64[] =
```

Note

Alternatively, the metadata shown in this example might be placed in the `/entry/instrument/detector` (*NXdetector*) group along with the image `data` since it provides image-related information such as size.

In the interest of keeping this example simpler and similar to the one above using the HDF5 File Writing Plugin, the metadata has been written into a *NXcollection* group at `/entry/instrument/metadata` location. (Compare

with the *NXcollection* group `/entry/instrument/NDAttributes` above.)

using the *nexusformat* package

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

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


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

    Parameters
    ----------
    fname : str
        name of the file (relative or absolute) to be written
    image : numpy array
        the image data
    md : dictionary
        key: value where value is something that can be written by h5py
             (such as str, int, float, numpy array, ...)
    """
    nx = NXroot()
    nx['/entry'] = NXentry(NXinstrument(NXdetector()))
    nx['entry/instrument/detector/image'] = NXfield(image, units='counts',
                                                    compression='gzip')
    nx['entry/data'] = NXdata()
    nx['entry/data'].makelink(nx['entry/instrument/detector/image'])
    nx['entry/data'].nxsignal = nx['entry/data/image']

    if len(md) > 0:
        # /entry/instrument/metadata (optional, for metadata)
        metadata = nx['/entry/instrument/metadata'] = NXcollection()
        for k, v in md.items():
            metadata[k] = v

    nx.save(fname, 'w')


if __name__ == "__main__":
    """demonstrate how to use this code"""
    import epics
    prefix = "13SIM1:"
    img = epics.caget(prefix+"image1:ArrayData")
    size_x = epics.caget(prefix+"cam1:ArraySizeX_RBV")
    size_y = epics.caget(prefix+"cam1:ArraySizeY_RBV")
    # edit the full image for just the binned data
```

(continues on next page)

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

(continued from previous page)

```
    img = img[:size_x*size_y].reshape((size_x, size_y))

    extra_information = dict(
        unique_id = epics.caget(prefix+"image1:UniqueId_RBV"),
        size_x = size_x,
        size_y = size_y,
        detector_state = epics.caget(prefix+"cam1:DetectorState_RBV"),
        bitcoin_value="15000",
    )
    write_nexus_file("example.h5", img, md=extra_information)
```

Visualization

You can visualize the HDF5 files with several programs, such as: hdfview[9], nexpy[10], or pymca[11]. Views of the test image shown using **NeXPy** (from the HDF5 file) and **caQtDM** (the image from EPICS) are shown.

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

Get the installation instructions for any of these programs from a web search. Other data analysis programs such as MatLab, IgorPro, and IDL can also read HDF5 files but you might have to work a bit more to get the data to a plot.

[9] hdfview: https://support.hdfgroup.org/products/java/hdfview/
[10] nexpy: https://nexpy.github.io/nexpy/
[11] pymca: http://pymca.sourceforge.net/

Downloads

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

Footnotes

2.4 Other tools to handle NeXus data files

The number of tools that read NeXus data files, either for general use or to read a specific application definition, is growing. Many of these are open source and so also serve as code examples. In the section *NeXus Utilities*, we describe many applications and software packages that can read, write, browse, and use NeXus data files. Examples of code (mostly from the NeXus community) that read NeXus data are listed in section *Language APIs for NeXus and HDF5*.

The NIAC welcomes your continued contributions to this documentation.

CHAPTER

THREE

NEXUS: REFERENCE DOCUMENTATION

3.1 Introduction to NeXus definitions

While the design principles of NeXus are explained in the *NeXus: User Manual*, this Reference Documentation specifies all allowed *base classes* and all standardized *application definitions*. Furthermore, it also contains *contributed definitions* of new bases classes or application definitions that are currently under review.

Base class definitions and application definitions have basically the same structure, but different semantics:

- Base class definitions define the *complete* set of terms that *might* be used in an instance of that class.
- Application definitions define the *minimum* set of terms that *must* be used in an instance of that class.

Base classes and application definitions are specified using a domain-specific XML scheme, the *NXDL: The NeXus Definition Language*.

3.1.1 Overview of NeXus definitions

For each class definition, the documentation is derived from content provided in the NXDL specification.

The documentation for each class consists of sections describing the *Status*, *Description*, table of *Symbols* (if defined), other NeXus base class *Groups cited*, an annotated *Structure*, and a link to the *NXDL Source* (XML) file.

Each of the NXDL files has its own tag in the version repository. Such as *NXcrystal-1.0* is tagged in GiHub and accessible via URL: https://github.com/nexusformat/definitions/releases/tag/NXcrystal-1.0

Description

General documentation if this NXDL file.

Symbols table

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

- All symbols used in the application definition are defined in a single `Symbols` table.
- The *name* of a symbol uses camel case without any white space or underscores.

  examples:

  nP: Total number of scan points

  nE: Number of photon energies scanned

  nFrames: Number of frames

  detectorRank: Rank of data array provided by the detector for a single measurement
- the `Symbols` table appears early in the .nxdl file above the `NXentry` group

  example from NXtomo.nxdl.xml

```
<definition name="NXtomo" extends="NXobject" type="group"
  category="application"
  xmlns="http://definition.nexusformat.org/nxdl/3.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://definition.nexusformat.org/nxdl/3.1 ../nxdl.xsd"
>
        <symbols>
                <doc>
                        These symbols will be used below to coordinate datasets with the␣
↪same shape.
                </doc>
                <symbol name="nFrames">
                        <doc>Number of frames</doc>
                </symbol>
                <symbol name="xSize">
                        <doc>Number of pixels in X direction</doc>
                </symbol>
                <symbol name="ySize">
                        <doc>Number of pixels in Y direction</doc>
                </symbol>
        </symbols>
        <doc>
                This is the application definition for x-ray or neutron tomography raw␣
↪data.

                In tomography
                a number of dark field images are measured, some bright field images and,
↪ of course the sample.
                In order to distinguish between them images carry a image_key.
```

(continues on next page)

(continued from previous page)

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

Annotated Structure

A representation of the basic structure (groups, fields, dimensions, attributes, and links) is prepared for each NXDL specification. Indentation shows nested structure. Attributes are prepended with the `@` symbol. Links use the characters `->` to represent the path to the intended source of the information.

Indentation is used to indicate nesting of subgroups (a feature common to application definitions). Within each indentation level, NeXus *fields* are listed first in the order presented in the NXDL file, then *groups*. *Attributes* are listed after the documentation of each item and are prefixed with the letter `@` (do not use the `@` symbol in the actual attribute name). The name of each item is in **bold**, followed by either *optional* or *required* and then the NXDL base class name (for groups) or the NeXus data type (for fields). If units are to be provided with the *field*, the type of the units is described, such as `NX_DATE_TIME`.

NeXus Links (these specifications are typically present only in application definitions) are described by a local name, the text `->`, then a suggested path to the source item to be linked to the local name.

Names (groups, fields, links, and attributes)

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

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

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

Attributes, identified with a leading "at" symbol (`@`) and belong with the preceding field or group, are additional metadata used to define this field or group. In the example above, the `program_name` element has the `configuration` (optional) attribute while the `thumbnail` element has the `mime_type` (optional) attribute.

For groups, the name may not be declared in the NXDL specification. In such instances, the *value shown in parentheses* in the *Name and Attributes* column is a suggestion, obtained from the group by removing the "NX" prefix. See *NXentry* for examples.

When the name is allowed to be *flexible* (the exact name given by this NXDL specification is not required but is set at the time the HDF file is written), the flexible part of the name will be written in all capital letters. For example, in the *NXdata* group, the `DATA`, `VARIABLE`, and `VARIABLE_errors` fields are *flexible*.

NeXus data type

Type of data to be represented by this variable. The type is one of those specified in *NXDL: The NeXus Definition Language*. In the case where the variable can take only one value from a known list, the list of known values is presented, such as in the `target_material` field above: `Ta | W | depleted_U | enriched_U | Hg | Pb | C`. Selections with included whitespace are surrounded by quotes. See the example above for usage.

For fields, the data type may not be specified in the NXDL file. The *default data type* is `NX_CHAR`. See *NXdata* for examples.

Units

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

Description

A simple text description of the field. No markup or formatting is allowed.

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

Choice

The `choice` element allows one to create a group with a defined name that is one specific NXDL base class from a defined list of possibilities

In some cases when creating an application definition, more than one choice of base class might be used to define a particular subgroup. For this particular situation, the `choice` was added to the NeXus NXDL Schema.

In this example fragment of an NXDL application definition, the `pixel_shape` could be represented by *either* `NXoff_geometry` or `NXcylindrical_geometry`.

```
<choice name="pixel_shape">
  <group type="NXoff_geometry">
    <doc>
      Shape description of each pixel. Use only if all pixels in the detector
      are of uniform shape.
    </doc>
  </group>
  <group type="NXcylindrical_geometry">
    <doc>
      Shape description of each pixel. Use only if all pixels in the detector
      are of uniform shape and require being described by cylinders.
    </doc>
  </group>
</choice>
```

[1] For NXDL *base classes*, `minOccurs=0` is the default, for NXDL *application definitions* and *contributed definitions*, `minOccurs=1` is the default. In all cases, the `minOccurs` attribute in the NXDL file will override the default for that element (group, field, attribute, or link).

The `@name` attribute of the `choice` element specifies the name that will appear in the HDF5 data file using one of the groups listed within the choice. Thus, it is not necessary to specify the name in each group. (At some point, the NXDL Schema may be modified to enforce this rule.)

A `choice` element may be used wherever a `group` element is used. It **must** have at least two groups listed (otherwise, it would not be useful).

3.2 NXDL: The NeXus Definition Language

Information in NeXus data files is arranged by a set of rules. These rules facilitate the exchange of data between scientists and software by standardizing common terms such as the way engineering units are described and the names for common things and the way that arrays are described and stored.

The set of rules for storing information in NeXus data files is declared using the NeXus Definition Language. NXDL itself is governed by a set of rules (a *schema*) that should simplify learning the few terms in NXDL. In fact, the NXDL rules, written as an XML Schema, are machine-readable using industry-standard and widely-available software tools for XML files such as `xsltproc` and `xmllint`. This chapter describes the rules and terms from which NXDL files are constructed.

3.2.1 Introduction

NeXus Definition Language (NXDL) files allow scientists to define the nomenclature and arrangement of information in NeXus data files. These NXDL files can be specific to a scientific discipline such as tomography or small-angle scattering, specific analysis or data reduction software, or even to define another component (base class) used to design and build NeXus data files.

In addition to this chapter and the *Tutorial* chapter, look at the set of NeXus NXDL files to learn how to read and write NXDL files. These files are available from the NeXus *definitions* repository and are most easily viewed on GitHub: https://github.com/nexusformat/definitions in the `base_classes`, `applications`, and `contributed` directories. The rules (expressed as XML Schema) for NXDL files may also be viewed from this URL. See the files `nxdl.xsd` for the main XML Schema and `nxdlTypes.xsd` for the listings of allowed data types and categories of units allowed in NXDL files.

NXDL files can be checked (validated) for syntax and content. With validation, scientists can be certain their definitions will be free of syntax errors. Since NXDL is based on the XML standard, there are many editing programs[1] available to ensure that the files are *well-formed*.[2] There are many standard tools such as `xmllint` and `xsltproc` that can process XML files. Further, NXDL files are backed by a set of rules (an *XML Schema*) that define the language and can be used to check that an NXDL file is both correct by syntax and valid by the NeXus rules.

NXDL files are machine-readable. This enables their automated conversion into schema files that can be used, in combination with other NXDL files, to validate NeXus data files. In fact, all of the tables in the *Class Definitions* Chapter have been generated directly from the NXDL files.

Writing references and anchors in the documentation.

[1] For example *XML Copy Editor* (http://xml-copy-editor.sourceforge.net/)
[2] http://en.wikipedia.org/wiki/XML#Well-formedness_and_error-handling

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.

For more details, see: *attributeType*

choice

A `choice` element is used when a named group might take one of several possible NeXus base classes. Logically, it must have at least two group children.

For more details, see: *choiceType*

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

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}:

In documentation, it may be useful to use an element that is not directly specified by the NXDL language. The *any* element here says that one can use any element at all in a `doc` element and NXDL will not process it but pass it through.

For more details, see: *docType*

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

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

Fig. 4: Graphical representation of the NXDL `doc` element

enumeration

An `enumeration` element can *only* be a child of a `field` or `attribute` element. It is used to restrict the available choices to a predefined list, such as to control varieties in spelling of a controversial word (such as *metre* vs. *meter*).

For more details, see: *enumerationType*

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

field

The `field` element provides the value of a named item. Many different attributes are available to further define the `field`. Some of the attributes are not allowed to be used together (such as `axes` and `axis`); see the documentation of each for details. It is used *only* as a child of a `group` element.

For more details, see: *fieldType*

group

A `group` element can *only* be a child of a `definition` or `group` element. It describes a common level of organization in a NeXus data file, similar to a subdirectory in a file directory tree.

For more details, see: *groupType*

link

A `link` element can *only* be a child of a `definition`, `field`, or `group` element. It describes the path to the original source of the parent `definition`, `field`, or `group`.

For more details, see: *linkType*

symbols

A `symbols` element can *only* be a child of a `definition` element. It defines the array index symbols to be used when defining arrays as `field` elements with common dimensions and lengths.

For more details, see: *symbolsType*

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

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

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

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

NXDL Field Types (internal)

Field types that define the NXDL language are described here. These data types are defined in the XSD Schema (`nxdl.xsd`) and are used in various parts of the Schema to define common structures or to simplify a complicated entry. While the data types are not intended for use in NXDL specifications, they define structures that may be used in NXDL specifications.

attributeType

Any new group or field may expect or require some common attributes.

(This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Attributes of attributeType

@name

Name of the attribute (unique within the enclosing group).

@optional

Is this attribute *optional* (if **true**) or *required* (if **false**)?

@recommended

A synonym for optional, but with the recommendation that this attribute be specified.

@type

Type of the attribute. For `group` specifications, the class name. For `field` or `attribute` specifications, the NXDL field type.

Elements of attributeType

dimensions

dimensions of an attribute with data value(s) in a NeXus file

doc

Description of this `attribute`. This documentation will go into the manual.

enumeration

An enumeration specifies the values to be used.

definition

A `definition` element is the `group` at the root of every NXDL specification. It may *only* appear at the root of an NXDL file and must only appear **once** for the NXDL to be *well-formed*.

definitionType

A `definition` is the root element of every NXDL definition. It may *only* appear at the root of an NXDL file and must only appear **once** for the NXDL to be *well-formed*.

The `definitionType` defines the documentation, attributes, fields, and groups that will be used as children of the `definition` element. Could contain these elements:

- `attribute`
- `doc`
- `field`
- `group`
- `link`

Note that a `definition` element also includes the definitions of the `basicComponent` data type. (The `definitionType` data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Note that the first line of text in a `doc` element in a `definition` is used as a summary in the manual. Follow the pattern as shown in the base class NXDL files.

Attributes of definitionType

@category

NXDL `base` definitions define the dictionary of terms to use for these components. All terms in a `base` definition are optional. NXDL `application` definitions define what is required for a scientific interest. All terms in an `application` definition are required. NXDL `contributed` definitions may be considered either base or applications. Contributed definitions must indicate their intended use, either as a base class or as an application definition.

@extends

The `extends` attribute allows this definition to *subclass* from another NXDL, otherwise `extends="NXobject"` should be used.

@ignoreExtraAttributes

Only validate known attributes; do not not warn about unknowns. The `ignoreExtraAttributes` attribute is a flag to the process of validating NeXus data files. By setting `ignoreExtraAttributes="true"`, presence of any undefined attributes in this class will not generate warnings during validation. Normally, validation will check all the attributes against their definition in the NeXus base classes and application definitions. Any items found that do not match the definition in the NXDL will generate a warning message.

The `ignoreExtraAttributes` attribute should be used sparingly!

@ignoreExtraFields

Only validate known fields; do not not warn about unknowns. The `ignoreExtraFields` attribute is a flag to the process of validating NeXus data files. By setting `ignoreExtraFields="true"`, presence of any undefined fields in this class will not generate warnings during validation. Normally, validation will check all the fields against their definition in the NeXus base classes and application definitions. Any items found that do not match the definition in the NXDL will generate a warning message.

The `ignoreExtraFields` attribute should be used sparingly!

@ignoreExtraGroups

Only validate known groups; do not not warn about unknowns. The `ignoreExtraGroups` attribute is a flag to the process of validating NeXus data files. By setting `ignoreExtraGroups="true"`, presence of any undefined groups in this class will not generate warnings during validation. Normally, validation will check all the groups against their definition in the NeXus base classes and application definitions. Any items found that do not match the definition in the NXDL will generate a warning message.

The `ignoreExtraGroups` attribute should be used sparingly!

@name

The `name` of this NXDL file (case sensitive without the file extension). The name must be unique amongst all the NeXus base class, application, and contributed definitions. For the class to be adopted by the NIAC, the first two letters must be "`NX`" (in uppercase). Any other use must *not* begin with "`NX`" in any combination of upper or lower case.

@restricts

The `restricts` attribute is a flag to the data validation. When `restricts="1"`, any non-standard component found (and checked for validity against this NXDL specification) in a NeXus data file will be flagged as an error. If the `restricts` attribute is not present, any such situations will produce a warning.

@svnid

(2014-08-19: deprecated since switch to GitHub version control) The identifier string from the subversion revision control system. This reports the time stamp and the revision number of this file.

@type

Must be `type="group"`

Elements of definitionType

symbols

Use a `symbols` list to define each of the mnemonics that represent the length of each dimension in a vector or array.

Groups under definitionType

In addition to an optional `symbols` list, a `definition` may contain any of the items allowed in a `group`.

definitionTypeAttr

Prescribes the allowed values for `definition` `type` attribute. (This data type is used internally in the NXDL schema to define a data type.)

The value may be any one from this list only:

- `group`
- `definition`

dimensionsType

dimensions of a data element in a NeXus file (This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Attributes of dimensionsType

@rank

Rank (number of dimensions) of the data structure.

Value could be either an unsigned integer or a symbol as defined in the *symbol* table of the NXDL file.

For example: `a[5]` has `rank="1"` while `b[8,5,6,4]` has `rank="4"`. See https://en.wikipedia.org/wiki/Rank_%28computer_programming%29 for more details.

Elements of dimensionsType

dim

Specify the parameters for each index of the `dimensions` element with a `dim` element. The number of `dim` entries should be equal to the `rank` of the array. For example, these terms describe a 2-D array with lengths (`nsurf`, `nwl`):

```
```

The `value` attribute is used by NXDL and also by the NeXus data file validation tools to associate and coordinate the same array length across multiple fields in a group.

@incr

Deprecated: 2016-11-23 telco (https://github.com/nexusformat/definitions/issues/330)

The dimension specification is related to the `refindex` axis within the `ref` field by an offset of `incr`. Requires `ref` and `refindex` attributes to be present.

@index

Number or symbol indicating which axis (subscript) is being described, ranging from 1 up to `rank` (rank of the data structure). For example, given an array `A[i,j,k]`, `index="1"` would refer to the `i` axis (subscript).

@ref

Deprecated: 2016-11-23 telco (https://github.com/nexusformat/definitions/issues/330)

The dimension specification is the same as that in the `ref` field, specified either by a relative path, such as `polar_angle` or `../Qvec` or absolute path, such as `/entry/path/to/follow/to/ref/field`.

@refindex

Deprecated: 2016-11-23 telco (https://github.com/nexusformat/definitions/issues/330)

The dimension specification is the same as the `refindex` axis within the `ref` field. Requires `ref` attribute to be present.

@required

This dimension is required (true: default) or not required (false).

The default value is `true`.

When `required="false"` is specified, all subsequent `<dim` nodes (with higher `index` value) **must** also have `required="false"`.

@value

Integer length (number of values), or mnemonic symbol representing the length of this axis.

doc

Documentation might be necessary to describe how the parts of the `dimensions` element are to be used.

docType

NXDL allows for documentation on most elements using the `doc` element. The documentation is useful in several contexts. The documentation will be rendered in the manual. Documentation, is provided as tooltips by some XML editors when editing NXDL files. Simple documentation can be typed directly in the NXDL:

```
Descriptive name of sample
```

This is suitable for basic descriptions that do not need extra formatting such as a bullet-list or a table. For more advanced control, use the rules of restructured text, such as in the *NXdetector* specification. Refer to examples in the NeXus base class NXDL files such as *NXdata*.

Could contain these elements:

- *any*

(This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Note: For documentation of `definition` elements, the first line of text in a `doc` is used as a summary in the manual. Follow the pattern as shown in the base class NXDL files.

enumerationType

An `enumeration` restricts the values allowed for a specification. Each value is specified using an `item` element, such as: `<item value="Synchrotron X-ray Source" />`. Could contain these elements:

- `doc`
- `item`

(This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

```
source operating mode

        for storage rings
        for storage rings
```

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

The `data_offset` attribute contains a comma-separated list of integers. (In addition to the required comma delimiter, whitespace is also allowed to improve readability.) The number of items in the list is equal to the rank of the data being stored. The value of each item is the offset in the array of the first data item of that subscript of the array.

@interpretation

This instructs the consumer of the data what the last dimensions of the data are. It allows plotting software to work out the natural way of displaying the data.

For example a single-element, energy-resolving, fluorescence detector with 512 bins should have `interpretation="spectrum"`. If the detector is scanned over a 512 x 512 spatial grid, the data reported will be of dimensions: 512 x 512 x 512. In this example, the initial plotting representation should default to data of the same dimensions of a 512 x 512 pixel `image` detector where the images where taken at 512 different pressure values.

In simple terms, the allowed values mean:

- `scalar` = 0-D data to be plotted
- `scaler` = DEPRECATED, use `scalar`
- `spectrum` = 1-D data to be plotted
- `image` = 2-D data to be plotted
- `rgb-image` = 3-D data to be plotted
- `rgba-image` = 3-D data to be plotted
- `hsl-image` = 3-D data to be plotted
- `hsla-image` = 3-D data to be plotted
- `cmyk-image` = 3-D data to be plotted
- `vertex` = 3-D data to be plotted

@long_name

Descriptive name for this field (may include whitespace and engineering units). Often, the long_name (when defined) will be used as the axis label on a plot.

@maxOccurs

Defines the maximum number of times this element may be used. Its value is confined to zero or greater. Must be greater than or equal to the value for the "minOccurs" attribute. A value of "unbounded" is allowed.

@minOccurs

Defines the minimum number of times this `field` may be used. Its value is confined to zero or greater. Must be less than or equal to the value for the "maxOccurs" attribute.

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

The `stride` attribute contains a comma-separated list of integers. (In addition to the required comma delimiter, whitespace is also allowed to improve readability.) The number of items in the list is equal to the rank of the data being stored. The value of each item is the spacing of the data items in that subscript of the array.

@type

Defines the type of the element as allowed by NeXus.

See *here* and *elsewhere* for the complete list of allowed types.

@units

String describing the engineering units. The string should be appropriate for the value.

- 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

[1] https://www.unidata.ucar.edu/software/udunits/

dimensions

dimensions of a data element in a NeXus file

enumeration

A field can specify which values are to be used

choiceType

A `choice` element is used when a named group might take one of several possible NeXus base classes. Logically, it must have at least two group children.

Attributes of choiceType

@name

The name to be applied to the selected child group. None of the child groups should define a `@name` attribute.

Elements of choiceType

group

NeXus base class that could be used here. The group will take the `@name` attribute defined by the parent `choice` element so do not specify the `@name` attribute of the group here.

groupType

A group element refers to the definition of an existing NX object or a locally-defined component. Could contain these elements:

- `attribute`
- `doc`
- `field`
- `group`
- `link`

Note that a `group` element also includes the definitions of the `basicComponent` data type. (The `groupType` data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Attributes of groupType

@maxOccurs

Maximum number of times this `group` is allowed to be present within its parent `group`. Note each `group` must have a `name` attribute that is unique among all `group` and `field` declarations within a common parent `group`.

@minOccurs

Minimum number of times this `group` is allowed to be present within its parent group. Note each `group` must have a `name` attribute that is unique among all `group` and `field` declarations within a common parent group.

@name

A particular scientific application may expect a name of a `group` element. It is helpful but not required to specify the `name` attribute in the NXDL file. It is suggested to always specify a `name` to avoid ambiguity. It is also suggested to derive the `name` from the type, using an additional number suffix as necessary. For example, consider a data file with only one `NXentry`. The suggested default `name` would be `entry`. For a data file with two or more `NXentry` groups, the suggested names would be `entry1`, `entry2`, … Alternatively, a scientific application such as small-angle scattering might require a different naming procedure; two different `NXaperture` groups might be given the names `beam_defining_slit` and `scatter_slit`.

@optional

A 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*.

https://github.com/nexusformat/code/blob/master/doc/api/NeXusIntern.pdf

@target

Declares the absolute HDF5 address of an existing field or group.

The target attribute is added for NeXus to distinguish the HDF5 path to the original dataset.

Could contain these elements:

- `doc`

Matching regular expression:

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

For example, given a `/entry/instrument/detector/polar_angle` field, link it into the `NXdata` group (at `/entry/data/polar_angle`). This would be the NeXus data file structure:

```
/: NeXus/HDF5 data file
        /entry:NXentry
                /data:NXdata
                        /polar_angle:NX_NUMBER
                                @target="/entry/instrument/detector/polar_angle
↪"
                /instrument:NXinstrument
                        /detector:NXdetector
                                /polar_angle:NX_NUMBER
                                        @target="/entry/instrument/detector/
↪polar_angle"
```

symbolsType

Each `symbol` has a `name` and optional documentation. Please provide documentation that indicates what each symbol represents. For example:

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

Elements of symbolsType

doc

Describe the purpose of this list of `symbols`. This documentation will go into the manual.

symbol

When multiple `field` elements share the same dimensions, such as the dimension scales associated with plottable data in an `NXdata` group, the length of each dimension written in a NeXus data file should be something that can be tested by the data file validation process.

@name

Mnemonic variable name for this array index symbol.

doc

Describe the purpose of the parent `symbol`. This documentation will go into the manual.

basicComponent

A `basicComponent` defines the allowed name format and attributes common to all `field` and `group` specifications. (This data type is used internally in the NXDL schema to define elements and attributes to be used by users in NXDL specifications.)

Attributes of basicComponent

@name

The `name` attribute is the identifier string for this entity. It is required that `name` must be unique within the enclosing `group`. The name must match the regular expression defined by `validItemName`. (Historical note: Originally, the rule (`validItemName`) was defined to allow only names that can be represented as valid variable names in most computer languages.)

Elements of basicComponent

doc

Describe this `basicComponent` and its use. This documentation will go into the manual.

validItemName

Used for allowed names of elements and attributes. Note: No - characters (among others) are allowed and you cannot start or end with a period (`.`). HDF4 had a 64 character limit on names (possibly including NULL) and the NAPI enforces this via the `NX_MAXNAMELEN` variable with a **64** character limit (which may be 63 on a practical basis if one considers a NULL terminating byte). (This data type is used internally in the NXDL schema to define a data type.) NOTE: In some languages, it may be necessary to add a `^` at the start and a `$` at the end of the regular expression to constrain the match to an entire line.

The value may be any `xs:token` that *also* matches the regular expression:

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

validNXClassName

Used for allowed names of NX class types (e.g. NXdetector). Note this is *not* the instance name (e.g. `bank1`) which is covered by `validItemName`. (This data type is used internally in the NXDL schema to define a data type.)

The value may be any `nx:validItemName` that *also* matches the regular expression:

```
NX.+
```

validTargetName

This is a valid link target - currently it must be an absolute path made up of valid names with the `/` character delimiter. But we may want to consider allowing "`..`" (parent of directory) at some point. If the `name` attribute is helpful, then use it in the path with the syntax of *name:type* as in these examples:

```
/NXentry/NXinstrument/analyzer:NXcrystal/ef
/NXentry/NXinstrument/monochromator:NXcrystal/ei
/NX_other
```

Must also consider use of `name` attribute in resolving `link` targets. (This data type is used internally in the NXDL schema to define a data type.)

From the HDF5 documentation:

> *Note that relative path names in HDF5 do not employ the ``../`` notation, the UNIX notation indicating a parent directory, to indicate a parent group.*

Thus, if we only consider the case of `[name:]type`, the matching regular expression syntax is written: `/[a-zA-Z_][\w_]*(:[a-zA-Z_][\w_]*)?)+`. Note that HDF5 also permits relative path names, such as: `GroupA/GroupB/Dataset1` but this is not permitted in the matching regular expression and not supported in NAPI.

The value may be any `xs:token` that *also* matches the regular expression:

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

nonNegativeUnbounded

A `nonNegativeUnbounded` allows values including all positive integers, zero, and the string `unbounded`. (This data type is used internally in the NXDL schema to define a data type.)

The `xs:string` data type

The `xs:string` data type can contain characters, line feeds, carriage returns, and tab characters. See https://www.w3schools.com/xml/schema_dtypes_string.asp for more details.

The `xs:token` data type

The `xs:string` data type is derived from the `xs:string` data type.

The `xs:token` data type also contains characters, but the XML processor will remove line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces. See https://www.w3schools.com/xml/schema_dtypes_string.asp for more details.

NXDL Data Types and Units

Data Types allowed in NXDL specifications

Data types for use in NXDL describe the expected type of data for a NeXus field or attribute. These terms are very broad. More specific terms are used in actual NeXus data files that describe size and array dimensions. In addition to the types in the following table, the `NAPI` type is defined when one wishes to permit a field with any of these data types. The default type `NX_CHAR` is applied in cases where a field or attribute is defined in an NXDL specification without explicit assignment of a `type`.

ISO8601
ISO8601 date/time stamp. It is recommended to add an explicit time zone, otherwise the local time zone *is assumed* per ISO8601. The norm is that if there is no time zone, it is assumed local time, however, when a file moves from one country to another it is undefined. If the local time zone is written, the ambiguity is gone.

NX_BINARY
any representation of binary data - if text, line terminator is [CR][LF]

NX_BOOLEAN
true/false value (true | 1 | false | 0)

NX_CCOMPLEX
Compound type cartesian representation of complex numbers (real and imaginary parts) in NeXus.

NX_CHAR
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): `m^2 | barns`

NX_CHARGE

units of electrical charge,
example(s): `C`

[1] https://www.unidata.ucar.edu/software/udunits/

NX_COUNT
units of quantity of item(s) such as number of photons, neutrons, pulses, or other counting events

NX_CROSS_SECTION

units of area (alias of NX_AREA),
example(s): `barn`

NX_CURRENT

units of electrical current,
example(s): `A`

NX_DIMENSIONLESS

units for fields where the units cancel out (NOTE: not the same as NX_UNITLESS),
example(s): `m/m`

NX_EMITTANCE

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

NX_ENERGY

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

NX_FLUX

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

NX_FREQUENCY

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

NX_LENGTH

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

NX_MASS

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

NX_MASS_DENSITY

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

NX_MOLECULAR_WEIGHT

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

NX_PERIOD

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

NX_PER_AREA

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

NX_PER_LENGTH

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

NX_POWER

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

NX_PRESSURE

units of pressure,
example(s): `Pa`

NX_PULSES
DEPRECATED: see NX_COUNT

units of clock pulses (alias to *NX_NUMBER*)

NX_SCATTERING_LENGTH_DENSITY

units of scattering length density,
example(s): `m/m^3`

NX_SOLID_ANGLE

units of solid angle,
example(s): `sr|steradian`

NX_TEMPERATURE

units of temperature,
example(s): `K`

NX_TIME

units of time,
example(s): `s`

NX_TIME_OF_FLIGHT

units of (neutron) time of flight (alias to *NX_TIME*),
example(s): `s`

NX_TRANSFORMATION
units of the specified transformation

could be any of these: `NX_LENGTH`, `NX_ANGLE`, or `NX_UNITLESS`

There will be one or more transformations defined by one or more fields for each transformation. The units type `NX_TRANSFORMATION` designates the particular axis generating a transformation (e.g. a rotation axis or a translation axis or a general axis). `NX_TRANSFORMATION` designates the units will be appropriate to the type of transformation, indicated in the *NXtransformations* base class by the `transformation_type` value:

- `NX_LENGTH` for `translation`
- `NX_ANGLE` for `rotation`
- `NX_UNITLESS` for axes for which no transformation type is specified.

NX_UNITLESS

for fields that don't have a unit (e.g. hkl) so that they don't inherit the wrong units (NOTE: not the same as NX_DIMENSIONLESS),
example(s): `""`

NX_VOLTAGE

units of voltage,
example(s): `V`

NX_VOLUME

units of volume,
example(s): `m^3`

NX_WAVELENGTH

units of wavelength,
example(s): `angstrom`

NX_WAVENUMBER

units of wavenumber or Q,
example(s): `1/nm` | `1/angstrom`

NXDL File Organisation

NXDL File Name

In order for the XML machinery to find and link the code in the various files, the name of the file must be composed of the definition name (matching both the spelling and the case) and a ".nxdl.xml" extension. For example, the base class `NXarbitrary_example` should be defined by NXDL code within the `NXarbitrary_example.nxdl.xml` file. Note also that the definition name is stated twice in application definitions, once in the `definition` tag, and again as the value of an `item` contained within the `field` tag that is named "definition".

Listing 1: NXarbitrary_example.nxdl.xml

```
<definition name="NXarbitrary_example">

<!-- later -->

   <field name="definition">
      <doc>Official NeXus NXDL schema to which this file conforms.</doc>
      <enumeration>
         <item value="NXarbitrary_example"/>
      </enumeration>
   </field>
</definition>
```

Documentation Images

Including images (or other related content) in the documentation of NXDL definitions can be very effective for communicating how different parts of the definition interact. To be properly included in the compilation of the NeXus documentation, the extra files must go into a directory having the same name as the definition without the NX prefix. For example, if the `NXarbitrary_example` base class has a `pretty_picture.jpg` image included in its documentation, then the image file should be located by the path (relative to `NXarbitrary_example.xml`) `arbitrary_example/pretty_picture.jpg`.

3.3 NeXus Class Definitions

Definitions of NeXus classes. These are split into base_classes (low level objects), application definitions (groupings of objects for a particular technique) and contributed_definitions (proposed definitions from the community)

The complete vocabulary of terms used in NeXus NXDL files (names of groups, fields, attributes, and links) is available for *download*.

Base classes

NeXus *base class* definitions define the set of terms that *might* be used in an instance of that class. Consider the base classes as a set of *components* that are used to construct a data file.

Base class definitions are permissive rather than restrictive. While the terms defined aim to cover most possible use cases, and to codify the spelling and meaning of such terms, the class specifications cannot list all acceptable groups and fields. To be able to progress the NeXus standard, additional data (groups, fields, attributes) are acceptable in NeXus HDF5 data files.

Users are encouraged to find the best *defined* location in which to place their information. It is understood there is not a predefined place for all possible data.

Validation procedures should treat such additional items (not covered by a base class specification) as notes or warnings rather than errors.

Application Definitions

NeXus *application definitions* define the *minimum* set of terms that *must* be used in an instance of that class. Application definitions also may define terms that are optional in the NeXus data file.

As in base classes (see above), additional terms that are not described by the application definition may be added to data files that incorporate or adhere to application definitions.

Use NeXus links liberally in data files to reduce duplication of data. In application definitions involving raw data, write the raw data in the *NXinstrument* tree and then link to it from the location(s) defined in the relevant application definition. See figure *NeXus Multi Method Hierarchy* for an example.

To write a data file with an application definition, start with either a *NXentry* (or *NXsubentry*) group 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.

For data files involving just an application definition, use the *NXentry* group. Such as this structure:

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

For files that describe multi-modal data and require use of two or more application definitions (such as *NXsas and NXcanSAS*), you must place each application definition in a *NXsubentry* of the *NXentry* group. Such as this structure:

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

If you anticipate your data file will eventually require an additional application definition, you should start with each application definition in a *NXsubentry* group.

Contributed Definitions

NXDL files in the NeXus *contributed definitions* include propositions from the community for NeXus base classes or application definitions, as well as other NXDL files for long-term archival by NeXus. Consider the contributed definitions as either in *incubation* or a special case not for general use.

3.3.1 Base Class Definitions

A description of each NeXus base class definition is given. NeXus base class definitions define the set of terms that *might* be used in an instance of that class. Consider the base classes as a set of *components* that are used to construct a data file.

A grouping of base classes is offered to assist users with navigating the full list of base classes along the following topics:

Core Classes

Instrument Components

Working with Samples

Working with Computers

Conventions, Reference Frames, and Data Analysis

Computational and Constructive Solid Geometry

Atom Probe Microscopy

Electron Microscopy

Multi-Dimensional Photoemission Spectroscopy

Optical Spectroscopy

Complete List

Core classes

Base Classes

NXcite
A literature reference

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

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

NXentry
(**required**) *NXentry* describes the measurement.

NXenvironment
Parameters for controlling external conditions

NXevent_data

NXlog
Information recorded as a function of time.

NXmonitor
A monitor of incident beam data.

NXnote
Any additional freeform information not covered by the other base classes.

NXobject
This is the base object of NeXus.

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

NXprocess
The *NXprocess* class describes an operation of data processing.

NXroot
The root of a NeXus file.

NXsample
Any information on the sample.

NXsubentry
Group of multiple application definitions for "multi-modal" (e.g. SAXS/WAXS) measurements.

NXuser
Contact information for a user.

Parts of instruments

Data schemas to describe parts, components, or sets of components for building an instrument.

Base Classes

NXactuator
An actuator used to control an external condition.

NXaperture
A beamline aperture.

NXattenuator
A device that reduces the intensity of a beam by attenuation.

NXbeam_stop
A device that blocks the beam completely, usually to protect a detector.

NXbending_magnet
A bending magnet

NXcapillary
A capillary lens to focus the X-ray beam.

NXcircuit
Base class for documenting circuit devices.

NXcollectioncolumn
Electron collection column of an electron analyzer.

NXcollimator
A beamline collimator.

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

NXcorrector_cs
Base class for a corrector reducing (spherical) aberrations of an electron optical setup.

NXcrystal
A crystal monochromator or analyzer.

NXdeflector
Component of an electron analyzer that deflects the paths of electrons. This includes electrostatic and electromagnetic deflectors.

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.

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

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

NXelectron_detector
A subclass of NXdetector for detectors that detect electrons.

NXelectronanalyzer
Basic class for describing an electron analyzer.

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

NXem_optical_system
Base class for qualifying an electron optical system.

NXenergydispersion
Energy dispersion section of an electron analyzer.

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

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.

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

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

NXinstrument
Collection of the components of the instrument or beamline.

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.

NXoptical_lens
Description of an optical lens.

NXoptical_window
A window of a cryostat, heater, vacuum chamber or a simple glass slide.

NXpdb
A NeXus transliteration of a PDB file, to be validated only as a PDB

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.

NXpump
Device to reduce an atmosphere to a controlled pressure.

NXreflections
Reflection data from diffraction experiments

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

NXsensor
A sensor used to monitor an external condition

NXslit
A simple slit.

NXsource
Radiation source emitting a beam.

NXspindispersion
Class to describe spin filters in photoemission experiments.

NXvelocity_selector
A neutron velocity selector

NXwaveplate
A waveplate or retarder.

NXxraylens
An X-ray lens, typically at a synchrotron X-ray beam line.

Working with Samples

Base Classes

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

NXatom
Base class for documenting a set of atoms.

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

NXenvironment
Parameters for controlling external conditions

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

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

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

NXnote
Any additional freeform information not covered by the other base classes.

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

NXroi_process
Base class to report on the characterization of an area or volume of material.

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.

NXunit_cell
Base class to describe structural aspects of an arrangement of atoms.

Working with Computers

Data schemas to describe a computer that was used for collecting or processing data.

Base Classes

NXcircuit
Base class for documenting circuit devices.

NXcs_computer
Base class for reporting the description of a computer.

NXcs_filter_boolean_mask
Base class for packing and unpacking booleans.

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

NXcs_prng
Computer science description of pseudo-random number generator.

NXcs_processor
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.

NXnote
Any additional freeform information not covered by the other base classes.

NXprocess
The *NXprocess* class describes an operation of processing data.

NXprogram
Base class to describe a software tool or library.

NXuser
Contact information for a user.

Conventions, Reference Frames, and Data Analysis

Base Classes

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.

NXcoordinate_system
Base class to detail a coordinate system (CS).

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

NXevent_data
NXevent_data is a special group for storing data from neutron

NXfilter
For band pass beam filters.

NXfit
Description of a fit procedure using a scalar valued global function

NXfit_function
This describes a fit function that is used to fit data to any functional form.

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

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

NXlog
Information recorded as a function of time.

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.

NXprocess
The *NXprocess* class describes an operation of processing data.

NXprogram
Base class to describe a software tool or library.

NXreflections
Reflection data from diffraction experiments

NXregistration
Describes image registration procedures.

NXresolution
Describes the resolution of a physical quantity.

NXroi_process
Base class to report on the characterization of an area or volume of material.

NXspectrum
Base class container for reporting a set of spectra.

NXtransformations
Collection of axis-based translations and rotations to describe a geometry.

Computational and Constructive Solid Geometry

Introduction

A set of base classes to describe the location and shape of objects using concepts from the fields of computer graphics, computational geometry, and constructive solid geometry (CSG).

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_primitive
Base class from which most other NXcg classes that define specific primitives inherit.

NXcg_ellipsoid
A description for a set of possibly dissimilar oriented ellipsoids.

NXcg_cylinder
A description for a set of possibly dissimilar oriented cylinders.

NXcg_point
A collection of points with labels.

***NXcg_polyline*:**
A collection of lines and linear segments.

NXcg_triangle
A collection of triangles.

NXcg_parallelogram A collection of possibly dissimilar parallelograms.

NXcg_polygon
A collection of polygons.

NXcg_polyhedron
A collection of polyhedra.

NXcg_roi
A container to host a number of different types of primitives.

NXcg_tetrahedron
A collection of tetrahedra.

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

An example how to use these classes follows:

```
/entry1:NXentry
    /reference_frame:NXcoordinate_system
        /x = [1, 0, 0]
        /y = [0, 1, 0]
        /z = [0, 0, 1]
    /unit_cube:NXcg_hexahedron
        /is_axis_aligned = True
        /hexahedron1:NXcg_face_list_data_structure
            /@depends_on = "/entry1/reference_frame"
            /number_of_vertices = 8
            /vertices = [[0, 0, 0], [1, 0, 0], [1, 1, 0], [0, 1, 0],
                [0., 0, 1], [1, 0, 1], [1, 1, 1], [0, 1, 1]]
```

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

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

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

NXcg_grid
A grid of cells.

NXisocontour
A description for isocontour descriptions.

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

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

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

Atom Probe Microscopy / Tomography

Introduction

The *NXapm* application definition uses base classes that describe the acquisition, i.e., the measurement side, the extraction of hits from detector raw data, processing steps to compute mass-to-charge-state ratios from uncorrected time of flight data, the reconstruction, and the ranging, i.e., identification of peaks in the mass-to-charge-state ratio histogram to detect (molecular) ions. The base classes can be useful to generate data artifacts also for field-ion microscopy experiments.

Base Classes

NXapm_charge_state_analysis
Base class to document the parameters, configuration, and results of a processing for recovering

NXapm_event_data
Base class to store state and (meta)data of events over the course of an atom probe experiment.

NXapm_instrument
Base class for instrument-related details of a real or simulated

NXapm_measurement
Base class for collecting a run with a real or a simulated atom probe or field-ion microscope.

NXapm_ranging
Base class for the configuration and results of ranging definitions.

NXapm_reconstruction
Base class for the configuration and results of a reconstruction algorithm.

NXapm_simulation
Base class for simulation of ion extraction from matter via laser and/or voltage

To see a full list of all base classes which NXapm uses, inspect the **Groups cited** section the *NXapm* application definition. Consider also the alignment between the design of the atom-probe- and electron-microscopy-specific definitions that is detailed in *Electron Microscopy*.

Electron Microscopy

Introduction

These are the base classes to describe components of an electron microscope and its focused-ion beam capabilities, as well as the associated data analyses. These base classes are used within the EM-related *application definitions*, specifically in NXem. Some of the base classes are specific to EM, whereas others are used in other techniques as well.

Base Classes

The design of NXem uses several existent base class and added concepts to existent base classes to make these applicable for electron microscopy (*NXactuator*, *NXaperture*, *NXbeam*, *NXcite*, *NXcollection*, *NXcomponent*, *NXcoordinate_system*, *NXdata*, *NXdeflector*, *NXdetector*, *NXfabrication*, *NXmanipulator*, *NXmonochromator*, *NXnote*, *NXparameters*, *NXprocess*, *NXsample*, *NXsensor*, *NXsource*, and *NXuser*).

Many design decisions of the application definitions *NXem* and *NXapm* are aligned. Examples are the use of base classes for instrument-specific events *NXem_event_data*, the grouping of measurements *NXem_measurement* and simulations *NXem_simulation*, and the encapsulating of *NXparameters* and *NXdata* in *NXprocess* instances to describe workflows of processing. The base classes *NXatom*, *NXunit_cell*, and *NXphase* were introduced to document sets of atoms, the spatial arrangement of atoms, and offer concepts for documenting when regions-of-interest *NXroi_process* in a material represent thermodynamic phases.

In addition to these considerations, there exist base classes to define concepts that are specific for electron microscopy:

***NXaberration*:**
A base class to describe procedures and values for the calibration of aberrations.

***NXcorrector_cs*:**
A base class to describe a corrective lens or compound lens sets to reduce the aberration of an electron beam.

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

***NXibeam_column*:**
A base class to group the components relevant for generating and shaping an ion beam.

***NXimage*:**
A base class to store individual images or stacks of images.

***NXem_instrument*:**
A base class to document all components that make up an instrument (real or simulated) when using it for studying electron matter interaction. This base class is used in NXem in two places: Firstly, 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. Secondly, inside ENTRY/measurement/eventID groups; these hold all those (meta)data data that change during a session.

***NXroi_process* and specialization *NXem_interaction_volume*:**
A base class to document the region-of-interest within an area or volume of material. The region of material where the electron beam interacts with the sample is called the interaction volume.

***NXelectromagnetic_lens*:**
A base class to describe 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.

***NXem_optical_system*:**
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, the magnification, or the camera length.

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

***NXscan_controller*:**
A base class to represent a component that is used to deflect a beam of charged particles in a controlled manner. This can be used to document the scan pattern.

***NXspectrum*:**
A base class to store individual spectra and stacks of 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 many data and metadata which are relevant for a given experiment have usually only few connections to the detailed description of the instrument. Instead, these are steps of data analysis and data processing workflows. This motivated a granularization of these concepts into own method-specific base classes:

***NXem_ebsd*, *NXem_eds*, *NXem_eels*, *NXem_img*:**
These base classes provide concepts for specific data acquisition modes and associated analyses as are used in electron microscopy such as for collecting and indexing Kikuchi diffraction patterns into orientation maps for two-dimensional, three-dimensional point cloud data, reporting X-ray spectroscopy (EDS/EDXS), different imaging modes, or documenting electron energy loss spectroscopy (EELS). A substantial further number of such base class could be designed that can build on the ideas and principles that are suggested via these four base classes.

Photoemission & Core-Level Spectroscopy

Introduction

These are a set of base classes to describe multidimensional photoemission (MPES) 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).

Some of the base classes are specific to MPES (like *NXelectronanalyzer*), whereas others are used in other techniques as well.

These base classes are used within the MPES-related *application definitions*.

Base Classes

Base classes describing instrument components used in photemission experiments:

***NXelectronanalyzer*:**
A base class to describe electron kinetic energy analyzers. Contains the collective characteristics of the instrument such as energy resolution, and includes the following classes:

- ***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 spin filters in photoemission experiments.
- ***NXelectron_detector*:**
  Specialization of *NXdetector* to describe electron detectors used in photoemission experiments.

Base classes (which are subclasses of *NXprocess*) to describe data (post-)processing:

***NXcalibration*:**
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*:**
Base class to describe the 2D distortion correction of an axis, with a matrix mapping a raw data image to a undistorted image.

***NXregistration*:**
Base class to describe the rigid transformations that are applied to an image.

***NXfit*:**
Base class to describe a fit procedure (e.g., peak fitting in XPS). This comes with its own set of base classes:

- ***NXfit_function*:**
  Base class to describe a fit function that is used to fit data to any functional form.
- ***NXpeak*:**
  Base class to describe a peak, its functional form, and support values (i.e., the discretization (points) at which the function has been evaluated).

Common Base Classes

There are additional base classes that were introduced in the context of photoemission spectroscopy, but that are commonly used in other techniques as well.

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

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

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

NXresolution:
Describes the resolution of a physical quantity, e.g. the resolution of the MPES spectrometer.

Optical Spectroscopy

Introduction

These are a set of base classes to describe optical spectroscopy, including *Ellipsometry* and *Raman spectroscopy*

These base classes are used within the *application definitions* related to optical spectroscopy.

Base Classes

The application definitions for optical spectroscopy use several existent base class and add edits and additions of some concepts to make these base classes applicable for the field of optical spectroscopy (*NXactuator*, *NXbeam*, *NXcalibration*, *NXcomponent*, *NXcoordinate_system*, *NXdata*, *NXdetector*, *NXenvironment*, *NXfabrication*, *NXhistory*, *NXinstrument*, *NXmanipulator*, *NXmonochromator*, *NXpid_controller*, *NXprocess*, *NXprogram*, *NXresolution*, *NXsample*, *NXsensor*, *NXsource*, *NXtransformations*, and *NXuser*).

In addition, there exist base classes to define concepts that are specific for optical spectroscopy:

NXbeam_transfer_matrix_table
Used to relate physical properties of two beams (*NXbeam*) which have one common optical component (*NXcomponent*) inbetween.

NXoptical_lens
Description of an optical lens.

NXoptical_window
Description of an optical window.

NXwaveplate
Description of a waveplate or retarder.

Base Classes

This is the complete list of base classes:

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.

NXapm_charge_state_analysis
Base class to document the parameters, configuration, and results of a processing for recovering

NXapm_event_data
Base class to store state and (meta)data of events over the course of an atom probe experiment.

NXapm_instrument
Base class for instrument-related details of a real or simulated

NXapm_measurement
Base class for collecting a run with a real or a simulated atom probe or field-ion microscope.

NXapm_ranging
Base class for the configuration and results of ranging definitions.

NXapm_reconstruction
Base class for the configuration and results of a reconstruction algorithm.

NXapm_simulation
Base class for simulation of ion extraction from matter via laser and/or voltage

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.

NXbeam_transfer_matrix_table
Contains data structures of an experimental optical setup (i.e., multiple

NXbending_magnet
A bending magnet

NXcalibration
Subclass of NXprocess to describe post-processing calibrations.

NXcapillary
A capillary lens to focus the X-ray beam.

NXcg_alpha_complex
Computational geometry of alpha complexes (alpha shapes or alpha wrappings) about primitives.

NXcg_cylinder
Computational geometry description of a set of cylinders or (truncated) cones.

NXcg_ellipsoid
Computational geometry description of a set of ellipsoids.

NXcg_face_list_data_structure
Computational geometry of primitives via a face-and-edge-list data structure.

NXcg_grid
Computational geometry description of a grid of Wigner-Seitz cells in Euclidean space.

NXcg_half_edge_data_structure
Computational geometry description of a half-edge data structure.

NXcg_hexahedron
Computational geometry description of a set of hexahedra in Euclidean space.

NXcg_parallelogram
Computational geometry description of a set of parallelograms.

NXcg_point
Computational geometry description of a set of points.

NXcg_polygon
Computational geometry description of a set of polygons in Euclidean space.

NXcg_polyhedron
Computational geometry description of a set of polyhedra in Euclidean space.

NXcg_polyline
Computational geometry description of a set of polylines.

NXcg_primitive
Computational geometry description of a set of primitives in Euclidean space.

NXcg_roi
Base class for a region-of-interest (ROI) bound by geometric primitives.

NXcg_tetrahedron
Computational geometry description of a set of tetrahedra.

NXcg_triangle
Computational geometry description of a set of triangles.

NXcg_unit_normal
Computational geometry description of a set of (oriented) unit normal vectors.

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.

NXcollectioncolumn
Electron collection column of an electron analyzer.

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 of an electron optical setup.

NXcrystal
A crystal monochromator or analyzer.

NXcs_computer
Base class for reporting the description of a computer

NXcs_filter_boolean_mask
Base class for packing and unpacking booleans.

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

NXcs_prng
Computer science description of pseudo-random number generator.

NXcs_processor
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
Component of an electron analyzer that deflects the paths of electrons. This includes electrostatic and electromagnetic deflectors.

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.

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

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

NXelectron_detector
A subclass of NXdetector for detectors that detect electrons.

NXelectronanalyzer
Basic class for describing an electron analyzer.

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

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

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

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

NXem_measurement
Base class for documenting a measurement with an electron microscope.

NXem_optical_system
Base class for qualifying an electron optical system.

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

NXenergydispersion
Energy dispersion section of an electron analyzer.

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

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.

NXfit
Description of a fit procedure using a scalar valued global function

NXfit_function
This describes a fit function that is used to fit data to any functional form.

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.

NXibeam_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.

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_lens
Description of an optical lens.

NXoptical_window
A window of a cryostat, heater, vacuum chamber or a simple glass slide.

NXorientation
legacy class - recommend to use *NXtransformations* now

NXparameters
Container for parameters used in processing or analysing data.

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
The *NXprocess* class describes an operation used to

NXprogram
Base class to describe a software tool or library.

NXpump
Device to reduce an atmosphere to a controlled pressure.

NXreflections
Reflection data from diffraction experiments

NXregistration
Describes image registration procedures.

NXresolution
Describes the resolution of a physical quantity.

NXroi_process
Base class to report on the characterization of an area or volume of material.

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.

NXscan_controller
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.

NXspindispersion
Class to describe spin filters in photoemission experiments.

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

NXwaveplate
A waveplate or retarder.

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 details about aberrations with relevance for 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.

The use of the base class is not restricted to electron microscopy but can also be useful for classical optics.

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 *NXcomponent*

Description:

An actuator used to control an external condition.

The condition itself is described in *NXenvironment*.

Symbols:

No symbol table

Groups cited:
NXpid_controller

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).

PID_CONTROLLER: (optional) *NXpid_controller*

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

Hypertext Anchors

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

- */NXactuator/actuation_target-field*
- */NXactuator/name-field*
- */NXactuator/outputVALUE-field*
- */NXactuator/physical_quantity-field*
- */NXactuator/PID_CONTROLLER-group*
- */NXactuator/short_name-field*
- */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/GEOMETRY-group*
- */NXaperture/material-field*
- */NXaperture/NOTE-group*
- */NXaperture/OFF_GEOMETRY-group*
- */NXaperture/POSITIONER-group*
- */NXaperture/shape-field*
- */NXaperture/size-field*

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

NXapm_charge_state_analysis

Status:

base class, extends *NXprocess*

Description:

Base class to document the parameters, configuration, and results of a processing for recovering the charge state and nuclide composition of an ion from ranging definitions as used in the research field of atom probe microscopy.

A ranging definition classically reports only the mass-to-charge-state-ratio interval plus the elemental composition, but not necessarily the nuclide that compose the ion.

As the mass-resolving-power in an atom probe instrument is finite and typically lower than for cutting edge tandem mass spectrometry it is possible that different combinations of nuclides are indistinguishable and thus multiple ions in eventually even different charge states can be valid labels for a given mass-to-charge-state-ratio peak. Enumerating the possible combinations is a programmatic approach that can help with peak identification.

Symbols:

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

n_cand: The number of ion candidates.

n_ivec_max: Maximum number of allowed atoms per ion.

n_variable: Number of entries

Groups cited:
NXparameters

Structure:

charge_state: (optional) *NX_INT* (Rank: 1, Dimensions: [n_cand]) {units=*NX_UNITLESS*}

Signed charge, i.e. integer multiple of the elementary charge of each candidate.

nuclide_hash: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_cand, n_ivec_max]) {units=*NX_UNITLESS*}

Table of nuclide instances of which each candidate is composed. Each row vector is sorted in descending order. Unused entries in the matrix should be set to 0. Use the hashing rule that is defined in nuclide_hash of *NXatom*.

mass: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_cand]) {units=*NX_MASS*}

Accumulated mass of the nuclides in each candidate. Not corrected for quantum effects.

natural_abundance_product: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_cand]) {units=*NX_DIMENSIONLESS*}

The product of the natural abundances of the nuclides for each candidate.

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

For each candidate the half life of the nuclide that has the shortest half life.

config: (optional) *NXparameters* <=

Parameters for the algorithm used to recover which combinations of nuclides have a mass and charge that matches a set of constraints.

Each parameter in this group is defines one constraint.

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

Parameter that defines the elements considered in the combinatorial search. The array contains nuclides as many times as their multiplicity and must not be empty. Nuclides are encoded using the hashing rule that is defined in by nuclide_hash of *NXatom*.

Constraining the elements or nuclides instead of providing all nuclides reduces the time to perform an exhaustive combinatorial search.

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

Parameter that defines the interval $[\frac{m}{q}_{min}, \frac{m}{q}_{max}]$ within which ions with given mass-to-charge-state-ratio qualify as candidates.

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

Parameter that defines the minimum half life for how long each nuclide of each ion needs to be stable such that the ion qualifies as a candidate.

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

Parameter that defines the minimum natural abundance of each nuclide of each ion such that the ion qualifies as a candidate.

sacrifice_isotopic_uniqueness: (optional) *NX_BOOLEAN*

If the value is false, it means that non-unique solutions are accepted. These are solutions where multiple candidates have been built from different nuclide instances but the charge_state of all the ions is the same.

Hypertext Anchors

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

- */NXapm_charge_state_analysis/charge_state-field*
- */NXapm_charge_state_analysis/config-group*
- */NXapm_charge_state_analysis/config/mass_to_charge_range-field*
- */NXapm_charge_state_analysis/config/min_abundance-field*
- */NXapm_charge_state_analysis/config/min_half_life-field*
- */NXapm_charge_state_analysis/config/nuclides-field*
- */NXapm_charge_state_analysis/config/sacrifice_isotopic_uniqueness-field*
- */NXapm_charge_state_analysis/mass-field*
- */NXapm_charge_state_analysis/natural_abundance_product-field*
- */NXapm_charge_state_analysis/nuclide_hash-field*
- */NXapm_charge_state_analysis/shortest_half_life-field*

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

NXapm_event_data

Status:

base class, extends *NXobject*

Description:

Base class to store state and (meta)data of events over the course of an atom probe experiment.

Having at least one instance for an instance of NXapm is recommended.

This base class applies the concept of the *NXem_event_data* base class to the specific needs of atom probe research. Again static and dynamic quantities are split to avoid a duplication of information. Specifically, the time interval considered is the entire time starting at start_time until end_time during which we assume the pulser triggered pulses. These pulses are identified via the pulse_id field. The point in time when each pulse was fired can be recovered from analyzing start_time and delta_time.

Which temporal granularity is adequate 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 atom probe experiments or simulation. To monitor the course of an ion extraction experiment (or simulation) it makes sense to track time explicitly via time stamps or implicitly via e.g. a clock inside the instrument, such as the clock of the pulser and respective pulse_id.

Symbols:

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

p: Number of pulses collected in between start_time and end_time.

Groups cited:
NXapm_instrument

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, the left bound of the time interval, while the end_time specifies the end, the 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.

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

Delta time array which resolves for each pulse_id the time difference between when that pulse was fired and start_time.

In summary, using start_time, end_time, delta_time, pulse_id_offset, and pulse_id provides temporal context information when a pulse was fired relative to start_time and when it is relevant to translate this into coordinated world time UTC.

Note that pulses in reality have a shape and thus additional documentation is required to assure that the entries in delta_time are always taken at at points in time that, relative to the triggering of the pulse, represent an as close as possible state of the pulse.

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

Integer which defines the first pulse_id. Typically, this is either zero or one.

pulse_id: (optional) *NX_INT* (Rank: 1, Dimensions: [p]) {units=*NX_UNITLESS*}

An integer to identify a specific pulse in a sequence.

There are two possibilities to report pulse_id values: If pulse_id_offset is provided, the pulse_id values are defined by the sequence $[pulse_id_offset, pulse_id_offset+p]$ with p the number of pulses collected in between start_time and end_time.

Alternatively, pulse_id_offset is not provided but instead a sequence of p values is defined. These integer values do not need to be sorted.

instrument: (optional) *NXapm_instrument*

Place to store dynamic metadata of the instrument to document as close as possible the state of the instrument during the event, i.e. in between start_time and end_time.

Hypertext Anchors

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

- */NXapm_event_data/delta_time-field*
- */NXapm_event_data/end_time-field*
- */NXapm_event_data/instrument-group*
- */NXapm_event_data/pulse_id-field*
- */NXapm_event_data/pulse_id_offset-field*
- */NXapm_event_data/start_time-field*

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

NXapm_instrument

Status:

base class, extends *NXinstrument*

Description:

Base class for instrument-related details of a real or simulated atom probe tomograph or field-ion microscope.

For collecting data and experiments which are simulations of an atom probe microscope or a session with such instrument use the *NXapm* application definition and the *NXapm_event_data* groups it provides.

This base class implements the concept of *NXapm* whereby (meta)data are distinguished whether these typically change during a session, so-called dynamic, or not, so-called 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:

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

p: Number of pulses collected in between start_time and end_time inside a parent instance of *NXapm_event_data*.

Groups cited:
NXbeam, *NXcomponent*, *NXdetector*, *NXfabrication*, *NXmanipulator*, *NXparameters*, *NXpump*, *NXsensor*, *NXsource*, *NXtransformations*

Structure:

type: (optional) *NX_CHAR*

Which type of instrument.

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

- `Inspico`
- `3DAP`
- `LAWATAP`
- `LEAP 3000 Si`
- `LEAP 3000X Si`
- `LEAP 3000 HR`
- `LEAP 3000X HR`
- `LEAP 4000 Si`
- `LEAP 4000X Si`
- `LEAP 4000 HR`
- `LEAP 4000X HR`
- `LEAP 5000 XS`
- `LEAP 5000 XR`
- `LEAP 5000 R`
- `EIKOS`
- `EIKOS-UV`
- `LEAP 6000 XR`
- `LEAP INVIZO`
- `Photonic AP`
- `TeraSAT`
- `TAPHR`
- `Modular AP`
- `Titanium APT`
- `Extreme UV APT`

location: (optional) *NX_CHAR*

Location of the lab or place where the instrument is installed. Using GEOREF is preferred.

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

Nominal flight path

The value can be extracted from the CAnalysis.CSpatial.fFlightPath field of a CamecaRoot ROOT file.

comment: (optional) *NX_CHAR*

Free-text field for additional comments.

fabrication: (optional) *NXfabrication* <=

reflectron: (optional) *NXcomponent*

Device which reduces ToF differences of ions in ToF experiments.

For atom probe the reflectron can be considered an energy compensation device. Such a device can be realized technically e.g. with a Poschenrieder lens.

Consult the following U.S. patents for further details:

- 3863068 and 6740872 for the reflectron
- 8134119 for the curved reflectron

applied: (optional) *NX_BOOLEAN* <=

Was the reflectron used?

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

The maximum voltage applied to the reflectron, relative to system ground.

decelerate_electrode: (optional) *NXcomponent*

A counter electrode of the LEAP 6000 series atom probes.

local_electrode: (optional) *NXcomponent*

A local electrode guiding the ion flight path. Also called counter or extraction electrode.

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

Acceleration voltage

aperture_type: (optional) *NX_CHAR*

The type of aperture used when the local_electrode has an aperture or acts as an aperture in addition to acting as an extraction electrode.

The local electrode is a component which combines functionalities of *NXelectromagnetic_lens*, *NXaperture*, if not even *NXdeflector*:

- "n/a", use when no aperture is present in the experiment
- "conical", conical aperture with a circular hole
- "feedthrough", an aperture where the specimen protrudes through a circular hole
- "custom", a user modified aperture, which is otherwise non-standard

Any of these values: `n/a` | `conical` | `feedthrough` | `custom`

ion_detector: (optional) *NXdetector* <=

Detector for taking raw time-of-flight and ion/hit impact positions data.

signal_amplitude: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [p]) {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 when the detector was an MCP.

The ATO file format is used primarily by the atom probe group of the GPM in Rouen, France.

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

The value can be extracted from the CRunHeader.fMcpEfficiency field of a Cameca-Root RHIT file.

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

The value can be extracted from the CRunHeader.fMeshEfficiency field of a Cameca-Root RHIT file.

pulser: (optional) *NXcomponent*

Laser- and/or voltage-pulsing device to trigger ion removal.

When the base class NXapm_instrument is used in the NXapm application definition, the values for the following fields:

- pulse_frequency
- pulse_fraction
- pulse_voltage
- pulse_number
- standing_voltage
- pulse_energy
- incidence_vector
- pinhole_position
- spot_position

should be recorded in the order of, and assumed associated, with the pulse_id in an instance of *NXapm_event_data*.

pulse_mode: (optional) *NX_CHAR*

Detail whereby ion extraction is triggered methodologically.

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

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

Frequency with which the pulser fire(s).

pulse_fraction: (optional) *NX_FLOAT* {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.

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

Pulsed voltage, in laser pulsing mode this field can be omitted.

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

Absolute number of pulses starting from the beginning of the experiment.

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

Direct current voltage between the specimen and the (local electrode) in the case of local electrode atom probe (LEAP) instrument. Otherwise, the standing voltage applied to the sample, relative to system ground.

fabrication: (optional) *NXfabrication* <=

sourceID: (optional) *NXsource*

Group to store details about components that enable laser pulsing strategies.

When multiple sources are available, these should be named source1, source2; the LEAP 6000 series instruments have two sources. The majority of instruments still has one source. In this case the variable part "ID" can be omitted. Consequently the group should be named "source" when writing instance data.

Atom probe microscopes use controlled laser, voltage, or a combination of pulsing strategies to trigger ion extraction via exciting and eventual field evaporation field emission of ion at the specimen surface.

wavelength: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=

The wavelength of the radiation emitted by the source.

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.

@logged_against: (optional) *NX_CHAR*

Path to pulse_id

beamID: (optional) *NXbeam*

Details about specific positions along the laser beam which illuminates the (atom probe) specimen.

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

Track time-dependent settings over the course of the measurement how the laser beam shines on the specimen, i.e. the mean vector is parallel to the laser propagation direction.

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

Track time-dependent settings over the course of the measurement where the laser beam exits the focusing optics.

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

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.

stage: (optional) *NXmanipulator*

temperature_sensor: (optional) *NXsensor* <=

value: (optional) *NX_FLOAT* <=

The value can be extracted from the CRunHeader.CAnalysis.fSpecimenTemperature field of a CamecaRoot RHIT file.

analysis_chamber: (optional) *NXcomponent*

flight_path: (optional) *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.

pressure_sensor: (optional) *NXsensor*

measurement: (optional) *NX_CHAR* <=

Obligatory value: `pressure`

value: (optional) *NX_FLOAT* {units=*NX_PRESSURE*} <=

The value can be extracted from the CRunHeader.CLasHeader.fAnalysisPressure field of a CamecaRoot RHIT file.

buffer_chamber: (optional) *NXcomponent*

load_lock_chamber: (optional) *NXcomponent*

getter_pump: (optional) *NXpump*

roughening_pump: (optional) *NXpump*

turbomolecular_pump: (optional) *NXpump*

control: (optional) *NXparameters* <=

Relevant quantities during a measurement with a LEAP system as were suggested by T. Blum et al..

evaporation_control: (optional) *NX_CHAR*

Parameter that defines the rules and control loops whereby the pulser and other components of the instrument are controlled during evaporation.

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

Parameter that assure maintenance of a significant yet not too high ion influx on the detector to avoid detection losses.

Hypertext Anchors

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

- */NXapm_instrument/analysis_chamber-group*
- */NXapm_instrument/analysis_chamber/flight_path-field*
- */NXapm_instrument/analysis_chamber/pressure_sensor-group*
- */NXapm_instrument/analysis_chamber/pressure_sensor/measurement-field*
- */NXapm_instrument/analysis_chamber/pressure_sensor/value-field*
- */NXapm_instrument/buffer_chamber-group*
- */NXapm_instrument/comment-field*

- */NXapm_instrument/control-group*
- */NXapm_instrument/control/evaporation_control-field*
- */NXapm_instrument/control/target_detection_rate-field*
- */NXapm_instrument/decelerate_electrode-group*
- */NXapm_instrument/fabrication-group*
- */NXapm_instrument/flight_path-field*
- */NXapm_instrument/getter_pump-group*
- */NXapm_instrument/ion_detector-group*
- */NXapm_instrument/ion_detector/mcp_efficiency-field*
- */NXapm_instrument/ion_detector/mesh_efficiency-field*
- */NXapm_instrument/ion_detector/signal_amplitude-field*
- */NXapm_instrument/load_lock_chamber-group*
- */NXapm_instrument/local_electrode-group*
- */NXapm_instrument/local_electrode/aperture_type-field*
- */NXapm_instrument/local_electrode/voltage-field*
- */NXapm_instrument/location-field*
- */NXapm_instrument/pulser-group*
- */NXapm_instrument/pulser/fabrication-group*
- */NXapm_instrument/pulser/pulse_fraction-field*
- */NXapm_instrument/pulser/pulse_frequency-field*
- */NXapm_instrument/pulser/pulse_mode-field*
- */NXapm_instrument/pulser/pulse_number-field*
- */NXapm_instrument/pulser/pulse_voltage-field*
- */NXapm_instrument/pulser/sourceID-group*
- */NXapm_instrument/pulser/sourceID/beamID-group*
- */NXapm_instrument/pulser/sourceID/beamID/incidence_vector-field*
- */NXapm_instrument/pulser/sourceID/beamID/pinhole_position-field*
- */NXapm_instrument/pulser/sourceID/beamID/spot_position-field*
- */NXapm_instrument/pulser/sourceID/power-field*
- */NXapm_instrument/pulser/sourceID/pulse_energy-field*
- */NXapm_instrument/pulser/sourceID/pulse_energy@logged_against-attribute*
- */NXapm_instrument/pulser/sourceID/TRANSFORMATIONS-group*
- */NXapm_instrument/pulser/sourceID/wavelength-field*
- */NXapm_instrument/pulser/standing_voltage-field*
- */NXapm_instrument/reflectron-group*
- */NXapm_instrument/reflectron/applied-field*

- */NXapm_instrument/reflectron/voltage-field*
- */NXapm_instrument/roughening_pump-group*
- */NXapm_instrument/stage-group*
- */NXapm_instrument/stage/temperature_sensor-group*
- */NXapm_instrument/stage/temperature_sensor/value-field*
- */NXapm_instrument/turbomolecular_pump-group*
- */NXapm_instrument/type-field*

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

NXapm_measurement

Status:

base class, extends *NXobject*

Description:

Base class for collecting a run with a real or a simulated atom probe or field-ion microscope.

The term run is understood as an exact synonym for session, i.e. the usage of a real or simulated tomograph or microscope for a certain amount of time during which one characterizes a single specimen.

Research workflows for experiments and simulations of atom probe and related field-evaporation evolve continuously and become increasingly connected with other methods used for material characterization specifically electron microscopy. A few examples in this direction are:

- T. Kelly et al.
- C. Fleischmann et al.
- W. Windl et al.
- C. Freysoldt et al.
- G. da Costa et al.

The majority of atom probe research is performed using the so-called Local Electrode Atom Probe (LEAP) instruments from AMETEK/Cameca. In addition, several research groups have built their own instruments and shared different aspects of the technical specifications and approaches including how these groups apply data processing e.g.:

- M. Monajem et al.
- P. Stender et al.
- I. Dimkou et al.

to name but a few.

Symbols:

No symbol table

Groups cited:
NXapm_event_data, *NXapm_instrument*

Structure:

status: (optional) *NX_CHAR*

A statement whether the measurement completed successfully, or was aborted.

Any of these values: `success` | `aborted`

quality: (optional) *NX_CHAR*

Statement about the quality of the measurement.

The value can be extracted from the CAnalysis.CResults.fQuality field of a CamecaRoot ROOT file.

APM_INSTRUMENT: (optional) *NXapm_instrument*

APM_EVENT_DATA: (optional) *NXapm_event_data*

Hypertext Anchors

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

- */NXapm_measurement/APM_EVENT_DATA-group*
- */NXapm_measurement/APM_INSTRUMENT-group*
- */NXapm_measurement/quality-field*
- */NXapm_measurement/status-field*

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

NXapm_ranging

Status:

base class, extends *NXprocess*

Description:

Base class for the configuration and results of ranging definitions.

Ranging is a data post-processing step used in the research field of atom probe during which elemental, isotopic, and/or molecular identities are assigned to mass-to-charge-state ratios within certain intervals. The documentation of these steps is based on ideas that have been described in the literature:

- M. K. Miller
- D. Haley et al.
- M. Kühbach et al.

Symbols:

No symbol table

Groups cited:
NXatom, *NXdata*, *NXnote*, *NXpeak*, *NXprocess*, *NXprogram*

Structure:

PROGRAM: (optional) *NXprogram*

NOTE: (optional) *NXnote* <=

mass_to_charge_distribution: (optional) *NXprocess*

Specifies the mass-to-charge-state ratio histogram.

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

Smallest $\frac{m}{q}_{min}$ mass-to-charge-state ratio value.

The lower (left-hand side) inclusive bound of the interval [$\frac{m}{q}_{min}$, {frac{m}{q}}_{max}]`.

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

Largest $\frac{m}{q}_{max}$ mass-to-charge-state ratio value.

The upper (right-hand side) inclusive bound of the interval [$\frac{m}{q}_{min}$, {frac{m}{q}}_{max}]`.

n_mass_to_charge: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

The number of bins on the interval [$\frac{m}{q}_{min}$, {frac{m}{q}}_{max}]`.

PROGRAM: (optional) *NXprogram*

mass_spectrum: (optional) *NXdata* <=

A default histogram aka mass spectrum of the mass-to-charge-state ratio values.

background_quantification: (optional) *NXprocess*

Details of the background model that was used to correct the total counts per bin into counts.

description: (optional) *NX_CHAR*

Free-text field to describe how atom probers define a background model.

Thereby, community feedback can be collected to inform an improved version of this base class in the future.

PROGRAM: (optional) *NXprogram*

peak_search_and_deconvolution: (optional) *NXprocess*

How were peaks in the mass-to-charge-state ratio histogram identified.

PROGRAM: (optional) *NXprogram*

PEAK: (optional) *NXpeak*

peak_identification: (optional) *NXprocess*

Details about how peaks, with taking into account error models, were interpreted as ion types or not.

number_of_ion_types: (optional) *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 representing a reserved value.

Consequently, start counting iontypes from 1.

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

Assumed maximum value that suffices to store all relevant molecular ions, even the most complicated ones that one can typically observe and distinguish typically. Currently, a value of 32 is used (see M. Kühbach et al. <https://doi.org/10.1017/S1431927621012241>`_).

PROGRAM: (optional) *NXprogram*

ATOM: (optional) *NXatom*

Hypertext Anchors

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

- */NXapm_ranging/background_quantification-group*
- */NXapm_ranging/background_quantification/description-field*
- */NXapm_ranging/background_quantification/PROGRAM-group*
- */NXapm_ranging/mass_to_charge_distribution-group*
- */NXapm_ranging/mass_to_charge_distribution/mass_spectrum-group*
- */NXapm_ranging/mass_to_charge_distribution/max_mass_to_charge-field*
- */NXapm_ranging/mass_to_charge_distribution/min_mass_to_charge-field*
- */NXapm_ranging/mass_to_charge_distribution/n_mass_to_charge-field*
- */NXapm_ranging/mass_to_charge_distribution/PROGRAM-group*
- */NXapm_ranging/NOTE-group*
- */NXapm_ranging/peak_identification-group*
- */NXapm_ranging/peak_identification/ATOM-group*
- */NXapm_ranging/peak_identification/maximum_number_of_atoms_per_molecular_ion-field*
- */NXapm_ranging/peak_identification/number_of_ion_types-field*
- */NXapm_ranging/peak_identification/PROGRAM-group*
- */NXapm_ranging/peak_search_and_deconvolution-group*
- */NXapm_ranging/peak_search_and_deconvolution/PEAK-group*
- */NXapm_ranging/peak_search_and_deconvolution/PROGRAM-group*
- */NXapm_ranging/PROGRAM-group*

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

NXapm_reconstruction

Status:

base class, extends *NXprocess*

Description:

Base class for the configuration and results of a reconstruction algorithm.

Generating a tomographic reconstruction of the specimen uses selected and 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.

Symbols:

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

n: Number of ions spatially filtered from results of the hit_finding algorithm from which an instance of a reconstructed volume has been generated. These ions get new identifier assigned in the process - the so-called evaporation_id, which must not be confused with the pulse_id!

Groups cited:
NXcollection, *NXdata*, *NXnote*, *NXparameters*, *NXprocess*, *NXprogram*

Structure:

reconstructed_positions: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n, 3]) {units=*NX_LENGTH*}

Three-dimensional positions of the ions in the reconstructed volume.

@depends_on: (optional) *NX_CHAR*

The instance of *NXcoordinate_system* in which the positions are defined.

quality: (optional) *NX_CHAR*

Qualitative statement about the reconstruction.

The value can be extracted from the CAnalysis.CResults.fQuality field of a CamecaRoot ROOT file.

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

Sum of ion volumes

The value can be extracted from the CAnalysis.CSpatial.fRecoVolume field of a CamecaRoot ROOT file.

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

The nominal diameter of the specimen ROI which is measured in the experiment. The physical specimen cannot be measured completely because ions may launch but hit in locations other than the detector.

PROGRAM: (optional) *NXprogram*

NOTE: (optional) *NXnote* <=

config: (optional) *NXparameters* <=

Parameters that configure a reconstruction algorithm which takes hit data and mass-to-charge-state ratio values to construct a model of the evaporated specimen. This model is called the reconstructed volume. Researchers in the field of atom probe call these algorithms reconstruction protocols.

Different such protocols exist. Although these are qualitatively similar, each protocol uses and interprets the parameters slightly differently.

The majority of reconstructions is performed with the proprietary software APSuite / IVAS, the source code for the reconstruction protocols that this software implements in detail is not open but the parameters and their qualitative effect on the reconstructed volume follows the protocols that are discussed in the atom probe literature. This group allows to document these parameters in a standardized manner.

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

Lowest voltage at which an ion that is considered in the reconstructed volume has been extracted from the specimen.

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

Highest voltage at which an ion that is considered in the reconstructed volume has been extracted from the specimen.

protocol_name: (optional) *NX_CHAR*

Qualitative statement about which reconstruction protocol was used.

For reconstructions performed with APSuite / IVAS the value "cameca" should be used.

Any of these values or a custom value (if you use a custom value, also set @custom=True): `bas` | `geiser` | `gault` | `cameca`

primary_element: (optional) *NX_CHAR*

Assumed primary element based on which the reconstruction is calibrated.

The value can be extracted from the CAnalysis.CSpatial.fPrimaryElement field of a CamecaRoot ROOT file.

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

Assumed detection efficiency

The value can be extracted from the CAnalysis.CSpatial.fEfficiency field of a CamecaRoot ROOT file.

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

Nominal flight path

The value can be extracted from the CAnalysis.CSpatial.fFlightPath field of a CamecaRoot ROOT file.

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

Assumed evaporation electric field

The value can be extracted from the CAnalysis.CSpatial.fEvaporationField field of a CamecaRoot ROOT file.

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

Image compression factor (ICF)

The value can be extracted from the CAnalysis.CSpatial.fImageCompression field of a CamecaRoot ROOT file.

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

The factor k in $R_0 = \frac{V}{kF}$ with R_0 tip_radius_zero V the voltage and F the evaporation field.

The value can be extracted from the CAnalysis.CSpatial.fKfactor field of a CamecaRoot ROOT file.

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

Shank angle

The value can be extracted from the CAnalysis.CSpatial.fShankAngle field of a CamecaRoot ROOT file.

ion_volume: (optional) *NX_FLOAT* {units=*NX_VOLUME*}

Assumed atomic volume

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

The value can be extracted from the CAnalysis.CSpatial.fTipRadius field of a CamecaRoot ROOT file.

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

The value can be extracted from the CAnalysis.CSpatial.fTipRadius0 field of a CamecaRoot ROOT file.

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

The value can be extracted from the CAnalysis.CSpatial.fVoltage0 field of a CamecaRoot ROOT file.

crystallographic_calibration: (optional) *NX_CHAR*

Different strategies for crystallographic calibration of the reconstruction are possible. Therefore, we collect first such feedback before parametrizing this further.

If no crystallographic calibration was performed, the field should be filled with the n/a, meaning not applied.

comment: (optional) *NX_CHAR*

Possibility of a free text field that allows to report additional details related to the reconstruction protocol. For LEAP systems and reconstructions that are performed with APSuite / IVAS see also *B. Gault et al. <https://doi.org/10.1093/mam/ozae081>_* and T. Blum et al. (page 371). for best practices on the reporting of metadata in atom probe tomography.

The value can be extracted from the CAnalysis.CResults.fComments field of a CamecaRoot ROOT file.

naive_discretization: (optional) *NXprocess*

PROGRAM: (optional) *NXprogram*

DATA: (optional) *NXdata* <=

Visual overview of the reconstructed dataset via a three-dimensional histogram of ion counts. Ion counts are characterized using one nanometer cubic bins without applying any smoothening of reconstructed positions during the histogram computation.

Such preview is useful to get an impression of the macroscopic shape of the reconstructed volume. Visualizing by ion counts highlights density variations the reconstructed volume that are signatures of features such as poles, interfaces or irregularities of the specimen shape.

obb: (optional) *NXcollection* <=

Tight, axis-aligned bounding box about the point cloud of the reconstruction.

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

Minimum coordinate value along the x-direction

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

Maximum coordinate value along the x-direction

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

Minimum coordinate value along the y-direction

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

Maximum coordinate value along the y-direction

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

Minimum coordinate value along the z-direction

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

Maximum coordinate value along the z-direction

Hypertext Anchors

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

- */NXapm_reconstruction/config-group*
- */NXapm_reconstruction/config/comment-field*
- */NXapm_reconstruction/config/crystallographic_calibration-field*
- */NXapm_reconstruction/config/efficiency-field*
- */NXapm_reconstruction/config/evaporation_field-field*
- */NXapm_reconstruction/config/flight_path-field*
- */NXapm_reconstruction/config/image_compression-field*
- */NXapm_reconstruction/config/ion_volume-field*
- */NXapm_reconstruction/config/kfactor-field*
- */NXapm_reconstruction/config/primary_element-field*
- */NXapm_reconstruction/config/protocol_name-field*
- */NXapm_reconstruction/config/shank_angle-field*
- */NXapm_reconstruction/config/tip_radius-field*
- */NXapm_reconstruction/config/tip_radius_zero-field*
- */NXapm_reconstruction/config/voltage_filter_final-field*
- */NXapm_reconstruction/config/voltage_filter_initial-field*
- */NXapm_reconstruction/config/voltage_zero-field*
- */NXapm_reconstruction/field_of_view-field*
- */NXapm_reconstruction/naive_discretization-group*
- */NXapm_reconstruction/naive_discretization/DATA-group*
- */NXapm_reconstruction/naive_discretization/PROGRAM-group*
- */NXapm_reconstruction/NOTE-group*
- */NXapm_reconstruction/obb-group*
- */NXapm_reconstruction/obb/xmax-field*
- */NXapm_reconstruction/obb/xmin-field*
- */NXapm_reconstruction/obb/ymax-field*
- */NXapm_reconstruction/obb/ymin-field*
- */NXapm_reconstruction/obb/zmax-field*
- */NXapm_reconstruction/obb/zmin-field*

- */NXapm_reconstruction/PROGRAM-group*
- */NXapm_reconstruction/quality-field*
- */NXapm_reconstruction/reconstructed_positions-field*
- */NXapm_reconstruction/reconstructed_positions@depends_on-attribute*
- */NXapm_reconstruction/volume-field*

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

NXapm_simulation

Status:

base class, extends *NXobject*

Description:

Base class for simulation of ion extraction from matter via laser and/or voltage pulsing.

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.

- */NXapm_simulation/DATA-group*
- */NXapm_simulation/PARAMETERS-group*
- */NXapm_simulation/PROCESS-group*
- */NXapm_simulation/PROGRAM-group*

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

n_ivec_max: Maximum number of atoms/isotopes allowed per ion.

n_ranges: Number of mass-to-charge-state-ratio range intervals for ion type.

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 ion like such as Al +++ or 12C +.

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

Given numerical identifier for the set.

The identifier zero is reserved for the special unknown ion type.

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 individual hash values. Consult the docstring of `nuclide_hash` for further details.

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

Position of each atom.

@depends_on: (optional) *NX_CHAR*

Path to an instance of *NXcoordinate_system* to document the reference frame in which the positions are defined.

This resolves 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*.

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

Vector of nuclide hash values. The vector is sorted in decreasing order.

Individual hash values H encode for each nuclide or element the number of protons Z and a constant c via the following hashing rule $H = Z + c \cdot 256$. Z and c must be 8-bit unsigned integers.

The constant c is either set to number of neutrons N or to the special value 255. The special value 255 is used to refer to all isotopes of an element from the IUPAC periodic table.

Some examples:

- The element hydrogen (meaning irrespective which isotope), its hash value is $H = 1 + 255 \cdot 256 = 65281$.
- The ^{1}H hydrogen isotope ($Z = 1, N = 0$), its hash value is $H = 1 + 0 \cdot 256 = 1$.
- The ^{2}H deuterium isotope ($Z = 1, N = 1$), its hash value is $H = 1 + 1 \cdot 256 = 257$.
- The ^{3}H tritium isotope ($Z = 1, N = 2$), its hash value is $H = 1 + 2 \cdot 256 = 513$.
- The ^{99}Tc technetium isotope ($Z = 43, N = 56$), its hash value is $H = 43 + 56 \cdot 256 = 14379$.

The special hash value 0 is a placeholder.

This hashing rule implements a bitshift operation. The benefit is that this enables encoding of all currently known nuclides and elements efficiently into an 16-bit unsigned integer. Sufficient unused indices remain to case situations when new elements will be discovered.

nuclide_list: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_ivec_max, 2]) {units=*NX_UNITLESS*}

Table which decodes the entries in nuclide_hash into a human-readable matrix instances for either nuclides or elements. Specifically, the first row specifies the nuclide mass number. When the nuclide_hash values are used this means the row should report the sum $Z + N$ or 0. The value 0 documents that an element from the IUPAC periodic table is meant. The second row specifies the number of protons Z. 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 column vector (14, 6). 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 $\frac{m}{q}_{min}$ and upper $\frac{m}{q}_{max}$ bounds of the mass-to-charge-state ratio interval(s) $[\frac{m}{q}_{min}, \frac{m}{q}_{max}]$. (boundaries inclusive). This field is primarily of interest for documenting *NXprocess* steps of indexing a ToF/mass-to-charge-state ratio histogram.

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/id-field*
- */NXatom/identifier_chemical-field*
- */NXatom/indices-field*
- */NXatom/mass_to_charge_range-field*
- */NXatom/name-field*
- */NXatom/nuclide_hash-field*
- */NXatom/nuclide_list-field*
- */NXatom/occupancy-field*
- */NXatom/position-field*
- */NXatom/position@depends_on-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/cm^2}

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.

BEAM**direction**: (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

NXbeam_transfer_matrix_table

Status:

base class, extends *NXobject*

Description:

Contains data structures of an experimental optical setup (i.e., multiple transfer matrix tables). These data structures are used to relate physical properties of two beams (NXbeam) which have one common optical component (NXcomponent) (one specific transfer matrix). One of these beams is an input beam and the other one is an output beam.

The data describes the change of beam properties, e.g. the intensity of a beam is reduced because the transmission coefficient of the beam device is lower than 1.

Symbols:

Variables used throughout the document, e.g. dimensions or parameters.

N_variables: Length of the array associated to the data type.

Groups cited:
none

Structure:

datatype_N: (optional) *NX_CHAR*

Select which type of data was recorded, for example aperture and focal length. It is possible to have multiple selections. This selection defines how many columns (N_variables) are stored in the data array. N in the name, is the index number in which order the given property is listed.

Any of these values:

- `aperture`
- `focal length`
- `orientation`
- `jones matrix`

matrix_elements: (optional) *NX_CHAR* (Rank: 1, Dimensions: [N_variables])

Please list in this array the column and row names used in your actual data. That is in the case of aperture ['diameter'] or focal length ['focal_length_value'] and for orientation matrix ['OM1', 'OM2', 'OM3'] or for jones matrix ['JM1','JM2']

TRANSFER_MATRIX: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [N_variables, N_variables])

Contains the datastructure which relates beam properties of an input and output beam as result of the input beam interaction with the beam device.

Transfer matrix relationship between N input beams and M output beams. It contains a table with the relevant matrices to be used for different transmitted properties (such as polarization, intensity, phase).

Data structure for all transfermatrices of a beam device in a setup. For each combination of N input and M output beams and for L physical concept (i.e. beam intensity), one matrix can be defined.

In this way, the transfer matrix table has the dimension NxM.

For each entry, in this transfer matrix, there are L formalisms. Each formalism has the dimension math:*dim(L_i)xdim(L_i)*, whereby math:*L_i* is the specific physical concept (Intensity, polarization, direction).

A beamsplitter with two input laser beams can have a total of four transfermatrices (2 Input x 2 Output).

The dimension of the transfer matrix depends on the parameters. Examples are: 1x1 for intensity/power 2x2 for jones formalism 3x3 for direction

@input: (optional) *NX_CHAR*

Specific name of input beam which the transfer matrix table is related to.

@output: (optional) *NX_CHAR*

Specific name of output beam which the transfer matrix table is related to.

Hypertext Anchors

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

- */NXbeam_transfer_matrix_table/datatype_N-field*
- */NXbeam_transfer_matrix_table/matrix_elements-field*
- */NXbeam_transfer_matrix_table/TRANSFER_MATRIX-field*
- */NXbeam_transfer_matrix_table/TRANSFER_MATRIX@input-attribute*
- */NXbeam_transfer_matrix_table/TRANSFER_MATRIX@output-attribute*

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

Here we can store a description of the formula used for the fit function.

For polynomial fits, use a0, a1, ..., an for the coefficients, corresponding to the values in the coefficients group. Use x0, x1, ..., xm for the mth 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 *fit_formula_inputs* group. The data should then be referred here by the *SYMBOL* name, e.g., for a field name `my_field` in `fit_formula_inputs`, it should be referred here by `my_field` or `my_field0` if you want to read the zeroth element of the array.

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 serialization 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.

PARAMETER: (optional) *NX_CHAR*

The name of each `PARAMETER` 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.

calibration_parameters: (optional) *NXparameters* <=

Fit coefficients to be used in `fit_formula_description`.

As an example, for nonlinear 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 for that procedure.

aN: (optional) *NX_NUMBER*

Use a0, a1, …, an for the coefficients of a polynomial fit, corresponding to the values in the `fit_formula_description`.

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*.

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/calibration_parameters-group*
- */NXcalibration/calibration_parameters/aN-field*
- */NXcalibration/calibration_parameters/offset-field*
- */NXcalibration/calibration_parameters/scaling_factor-field*
- */NXcalibration/DATA-group*

- */NXcalibration/description-field*
- */NXcalibration/fit_formula_description-field*
- */NXcalibration/fit_formula_inputs-group*
- */NXcalibration/fit_formula_inputs/PARAMETER-field*
- */NXcalibration/fit_formula_inputs/PARAMETER@input_path-attribute*
- */NXcalibration/identifier_calibration_method-field*
- */NXcalibration/identifier_calibration_reference-field*
- */NXcalibration/last_process-field*
- */NXcalibration/mapping_MAPPING-field*
- */NXcalibration/original_axis-field*
- */NXcalibration/original_axis@input_path-attribute*
- */NXcalibration/original_axis@symbol-attribute*
- */NXcalibration/physical_quantity-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

NXcg_alpha_complex

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry of alpha complexes (alpha shapes or alpha wrappings) about 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 of alpha shapes, and

- https://doc.cgal.org/latest/Manual/packages.html#PkgAlphaWrap3 for 3D alpha wrappings

in CGAL, the Computational Geometry Algorithms Library respectively. As a starting point, we follow the conventions of the CGAL library.

In general, an alpha complex is a not necessarily connected or not necessarily pure complex, i.e. singular faces may exist. The number of cells, faces, and edges depends on how a specific alpha complex is filtered for lower-dimensional simplices. The fields is_regularized and regularization can be used to provide details about regularization procedures.

Symbols:

No symbol table

Groups cited:
NXcg_point, *NXcg_tetrahedron*, *NXcg_triangle*

Structure:

type: (optional) *NX_CHAR*

Type of alpha complex following the terminology used by CGAL for now.

Alpha_shape means meshes created using one of the alpha_shape algorithm. Alpha_wrapping means meshes created using the alpha_wrapping algorithm.

Any of these values: `convex_hull` | `alpha_shape` | `alpha_wrapping`

regularization: (optional) *NX_CHAR*

Human-readable description about regularization procedures.

is_regularized: (optional) *NX_BOOLEAN*

Was the alpha complex regularized, i.e. have singular faces been removed, or not.

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

The alpha parameter, i.e. the squared radius of the alpha-sphere that is used when computing the alpha complex.

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

The offset distance parameter used when computing alpha_wrappings.

CG_POINT: (optional) *NXcg_point*

Point cloud serving as input for the computation of the alpha complex.

TRIANGLE_SOUP: (optional) *NXcg_triangle*

Triangle soup serving as input for the computation of the alpha complex.

ALPHA_COMPLEX: (optional) *NXcg_triangle*

Triangle mesh representing the output of the computation, i.e. the alpha complex.

TETRAHEDRALIZATION: (optional) *NXcg_tetrahedron*

Tetrahedra representing an interior volume of the alpha complex (if such exists).

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/ALPHA_COMPLEX-group*
- */NXcg_alpha_complex/CG_POINT-group*
- */NXcg_alpha_complex/is_regularized-field*
- */NXcg_alpha_complex/offset-field*
- */NXcg_alpha_complex/regularization-field*
- */NXcg_alpha_complex/TETRAHEDRALIZATION-group*
- */NXcg_alpha_complex/TRIANGLE_SOUP-group*
- */NXcg_alpha_complex/type-field*

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

NXcg_cylinder

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of cylinders or (truncated) cones.

The radius can either be defined in the radii field or by filling the upper_cap_radii and lower_cap_radii fields respectively. The latter field case can thus be used to represent (truncated) cones.

It is possible to define only one of the cap_radii fields to represent half-open cylinder.

Symbols:

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

d: The dimensionality of the space in which the members are assumed embedded.

c: The cardinality of the set, i.e. the number of members.

Groups cited:
none

Structure:

height: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*} <=

A direction vector which is parallel to the cylinder/cone axis and whose magnitude is the height of the cylinder/cone.

The upper_cap is assumed to represent the end while the lower_cap is assumed to represent the start of the respective cylinder instances when inspecting along the direction vector.

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

Radius of the cylinder if all have the same radius.

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

Radii of the cylinder.

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

Radii of the upper circular cap.

This field, combined with lower_cap_radius can be used to describe (eventually truncated) circular cones.

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

Radii of the upper circular cap.

This field, combined with upper_cap_radius can be used to describe (eventually truncated) circular cones.

lateral_surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Lateral surface area of each cylinder.

upper_cap_surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Area of the upper cap of each cylinder.

lower_cap_surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Area of the lower cap of each cylinder.

total_surface_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Sum of upper and lower cap area and lateral surface area of each cylinder.

Hypertext Anchors

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

- */NXcg_cylinder/height-field*
- */NXcg_cylinder/lateral_surface_area-field*
- */NXcg_cylinder/lower_cap_radii-field*
- */NXcg_cylinder/lower_cap_surface_area-field*
- */NXcg_cylinder/radii-field*
- */NXcg_cylinder/radius-field*
- */NXcg_cylinder/total_surface_area-field*
- */NXcg_cylinder/upper_cap_radii-field*
- */NXcg_cylinder/upper_cap_surface_area-field*

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

NXcg_ellipsoid

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of ellipsoids.

Symbols:

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

d: The dimensionality of the space in which the members are assumed embedded.

c: The cardinality of the set, i.e. the number of members.

Groups cited:
none

Structure:

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

Length of the semi-axes (e.g. semi-major and semi-minor respectively for an ellipse).

Use if all ellipsoids in the set have the same half-axes.

semi_axes_values: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Length of the semi-axes if ellipsoids have individually different lengths.

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

In the case that all ellipsoids are spheres.

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

In the case that all ellipsoids are spheres whose radii differ. For a mixture of spheres use semi_axes_values.

Hypertext Anchors

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

- */NXcg_ellipsoid/radii-field*
- */NXcg_ellipsoid/radius-field*
- */NXcg_ellipsoid/semi_axes_value-field*
- */NXcg_ellipsoid/semi_axes_values-field*

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

NXcg_face_list_data_structure

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry of primitives via a face-and-edge-list data structure.

Primitives must neither be degenerated nor self-intersect but can have different properties. A face-and-edge-list-based description of primitives is frequently used for triangles and polyhedra to store them on disk for visualization purposes (see OFF, PLY, VTK, or STL file formats).

Although this description is storage efficient, it is not well-suited for topological analyses. In this case using a half-edge data structure is an alternative.

Having an own base class for the data structure how primitives are stored is useful to embrace both users with small or detailed specification demands.

Indices can be used as identifier and thus names for individual instances.

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.

Groups cited:
none

Structure:

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

Number of vertices for each face.

Each entry represents the total number of vertices for that face, irrespectively whether vertices are shared among faces or not.

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

Number of edges for each face.

Each entry represents the total number of edges for that face, irrespectively whether edges are shared across faces or not.

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

Number of faces of the primitives.

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

Integer offset whereby the identifier of the first member of the vertices differs from zero.

Identifier can be defined explicitly or implicitly. Inspect the definition of NXcg_primitive for further details.

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

Integer offset whereby the identifier of the first member of the edges differs from zero.

Identifier can be defined explicitly or implicitly. Inspect the definition of NXcg_primitive for further details.

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

Integer offset whereby the identifier of the first member of the faces differs from zero.

Identifier can be defined explicitly or implicitly. Inspect the definition of NXcg_primitive for further details.

indices_vertex: (optional) *NX_INT* (Rank: 1, Dimensions: [n_v]) {units=*NX_UNITLESS*}

Integer identifier to distinguish all vertices explicitly.

indices_edge: (optional) *NX_INT* (Rank: 1, Dimensions: [n_e]) {units=*NX_UNITLESS*}

Integer used to distinguish all edges explicitly.

indices_face: (optional) *NX_INT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Integer used to distinguish all faces explicitly.

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

Positions of the vertices.

Users are encouraged to reduce the vertices to a unique set as this may result in more efficient storage. Alternatively, storing vertex positions naively should be indicated with setting vertices_are_unique to False. Naively means that each vertex is stored even though many vertices may share the same positions.

edges: (optional) *NX_INT* (Rank: 2, Dimensions: [n_e, 2]) {units=*NX_UNITLESS*}

The edges are stored as pairs of vertex identifier.

faces: (optional) *NX_INT* (Rank: 1, Dimensions: [n_total]) {units=*NX_UNITLESS*}

The faces are stored as a concatenated array of vertex identifier tuples.

The first entry is the identifier of the start vertex of the first face, followed by the second vertex of the first face, until the last vertex of the first face. Thereafter, the start vertex of the second face, the second vertex of the second face, and so on and so forth.

Therefore, summating over the number_of_vertices, allows to extract the vertex identifiers for the i-th face on the following index interval of the faces array: $[\sum_{i=0}^{i=n-1}, \sum_{i=0}^{i=n}]$.

vertices_are_unique: (optional) *NX_BOOLEAN*

If true, indicates that the vertices are all placed at different positions and have different identifiers, i.e. no points overlap or are counted more than once.

edges_are_unique: (optional) *NX_BOOLEAN*

If true, indicates that no edge is stored more than once.

Users are encouraged to consider using a half_edge_data_structure instead.

faces_are_unique: (optional) *NX_BOOLEAN*

If true, indicates that no face is stored more than once.

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/edges-field*
- */NXcg_face_list_data_structure/edges_are_unique-field*
- */NXcg_face_list_data_structure/faces-field*
- */NXcg_face_list_data_structure/faces_are_unique-field*
- */NXcg_face_list_data_structure/index_offset_edge-field*
- */NXcg_face_list_data_structure/index_offset_face-field*
- */NXcg_face_list_data_structure/index_offset_vertex-field*
- */NXcg_face_list_data_structure/indices_edge-field*
- */NXcg_face_list_data_structure/indices_face-field*
- */NXcg_face_list_data_structure/indices_vertex-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/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/base_classes/NXcg_face_list_data_structure.nxdl.xml

NXcg_grid

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a grid of Wigner-Seitz cells in Euclidean space.

Three-dimensional grids with cubic cells are if not the most frequently used example of such grids. Numerical methods and models that use grids are used in many cases in the natural sciences and engineering disciplines. Examples are discretizations in space and time used for phase-field, cellular automata, or Monte Carlo modeling.

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 aka grid points.

n_b: Number of boundaries of the bounding box or primitive housing the grid.

Groups cited:
NXcg_polyhedron

Structure:

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

Location of the origin of the grid.

Use the depends_on field that is inherited from the *NXcg_primitive* class to specify the coordinate system in which the origin location is defined.

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_INT* (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.

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

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_UINT* {units=*NX_UNITLESS*}

Number of boundaries 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

surface_reconstruction: (optional) *NX_CHAR*

Details about the computational geometry method and implementation used for discretizing internal surfaces as e.g. obtained with marching methods, like marching squares or marching cubes.

Documenting which specific version was used helps with understanding how robust the results are with respect to the topology of the triangulation. 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 ideally be an identifier of a program. If not possible, an identifier for a paper, technical report, or free-text description can be used instead.

bounding_box: (optional) *NXcg_polyhedron*

A tight bounding box 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/cell_dimensions-field*
- */NXcg_grid/coordinate-field*
- */NXcg_grid/extent-field*
- */NXcg_grid/number_of_boundaries-field*
- */NXcg_grid/origin-field*
- */NXcg_grid/position-field*
- */NXcg_grid/surface_reconstruction-field*
- */NXcg_grid/symmetry-field*

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

NXcg_half_edge_data_structure

Status:

base class, extends *NXcg_primitive*

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. The data structure is also known as a doubly connected edge list.

Indices can be used as identifier and thus names for individual instances.

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*} <=

Dimensionality of the primitives described.

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

Number of vertices for each face.

Each entry represents the total number of vertices for that face, irrespectively whether vertices are shared among faces or not.

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

Number of edges for each face.

Each entry represents the total number of edges for that face, irrespectively whether edges are shared across faces or not.

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

Integer offset whereby the identifier of the first member of the vertices differs from zero.

Identifier can be defined explicitly or implicitly. Inspect the definition of *NXcg_primitive* for further details.

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

Integer offset whereby the identifier of the first member of the edges differs from zero.

Identifier can be defined explicitly or implicitly. Inspect the definition of *NXcg_primitive* for further details.

index_offset_face: (optional) *NX_INT*

Integer offset whereby the identifier of the first member of the faces differs from zero.

Identifier can be defined explicitly or implicitly. Inspect the definition of *NXcg_primitive* for further details.

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

The position of the vertices.

vertex_incident_half_edge: (optional) *NX_INT* (Rank: 1, Dimensions: [n_v]) {units=*NX_UNITLESS*}

Identifier of the incident half-edge.

face_half_edge: (optional) *NX_INT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Identifier of the (starting)/associated half-edge of the face.

half_edge_vertex_origin: (optional) *NX_INT* (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_INT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

Identifier of the associated oppositely pointing half-edge.

half_edge_incident_face: (optional) *NX_INT* (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_INT* (Rank: 1, Dimensions: [n_he]) {units=*NX_UNITLESS*}

Identifier of the next half-edge.

half_edge_prev: (optional) *NX_INT* (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/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/index_offset_edge-field*
- */NXcg_half_edge_data_structure/index_offset_face-field*
- */NXcg_half_edge_data_structure/index_offset_vertex-field*
- */NXcg_half_edge_data_structure/number_of_edges-field*
- */NXcg_half_edge_data_structure/number_of_vertices-field*
- */NXcg_half_edge_data_structure/position-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/base_classes/NXcg_half_edge_data_structure.nxdl.xml

NXcg_hexahedron

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of hexahedra in Euclidean space.

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 an approach whereby different specific views can be implemented using the same base class:

- In the simplest case experimentalists may use this base class to describe the dimensions or size of a specimen. In this case the alignment with axes is not relevant as eventually only the volume of the specimen is of interest.
- In many cases, take for example an experiment where a specimen was cut out from a specifically deformed piece of material, the orientation of the specimen's edges with the experiment coordinate system is of high relevance. Examples include knowledge about the specimen edge, whether it is parallel to the rolling, the transverse, or the normal direction.
- While the above-mentioned use cases are sufficient to pinpoint the sample within a known laboratory/experiment coordinate system, these descriptions are not detailed enough to specify e.g. a CAD model of the specimen.
- Therefore, groups and fields for an additional, computational-geometry- based view of hexahedra is offered to serve additional 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.

As a specialization of the *NXcg_primitive* base class hexahedra are assumed non-degenerated, closed, and built of polygons that are not self-intersecting.

The term hexahedra will be used throughout this base class but includes the special cases cuboid, cube, box, axis-aligned bounding box (AABB), and optimal bounding box (OBB).

An axis-aligned bounding box is a common data object in computational science and simulation codes to represent a cuboid whose edges are aligned with the base vectors of a coordinate system. As a part of binary trees, these data objects are important for making 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 tightly fitting box about an arbitrary object. In general, such boxes are rotated. Exact and substantially faster in practice approximate algorithms exist to compute optimal or near optimal bounding boxes for sets of points.

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*

Structure:

shape: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*} <=

Qualifier for the shape of each hexahedron.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*} <=

Qualifier that is useful in cases when one edge is longer than all other edges of the hexahedra. Often the term length is associated with the assumption that one edge is 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 extent of an object in the horizontal direction assuming a specific coordinate system.

For the sake of explicitness quantities like length, width, and height should not be reported without specifying also the assumed reference frame.

height: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_LENGTH*} <=

Qualifier often used to describe the extent of an object in the vertical direction assuming a specific coordinate system.

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_VOLUME*} <=

Volume of each hexahedron.

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*} <=

Total (surface) 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 coordinate system.

vertex_normal: (optional) *NXcg_unit_normal* <=

edge_normal: (optional) *NXcg_unit_normal* <=

face_normal: (optional) *NXcg_unit_normal* <=

hexahedra: (optional) *NXcg_face_list_data_structure*

Combined storage of all primitives of all hexahedra.

hexahedronID: (optional) *NXcg_face_list_data_structure*

Individual storage of each hexahedron.

hexahedron_half_edgeID: (optional) *NXcg_half_edge_data_structure*

Individual storage of each hexahedron as a graph.

Hypertext Anchors

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

- */NXcg_hexahedron/area-field*
- */NXcg_hexahedron/edge_normal-group*
- */NXcg_hexahedron/face_area-field*
- */NXcg_hexahedron/face_normal-group*
- */NXcg_hexahedron/height-field*
- */NXcg_hexahedron/hexahedra-group*
- */NXcg_hexahedron/hexahedron_half_edgeID-group*
- */NXcg_hexahedron/hexahedronID-group*
- */NXcg_hexahedron/is_axis_aligned-field*
- */NXcg_hexahedron/is_box-field*
- */NXcg_hexahedron/length-field*
- */NXcg_hexahedron/shape-field*
- */NXcg_hexahedron/vertex_normal-group*
- */NXcg_hexahedron/volume-field*
- */NXcg_hexahedron/width-field*

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

NXcg_parallelogram

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of parallelograms.

This class can also be used to describe rectangles or squares, irrespective whether these are axis-aligned or not. The class represents different access and description levels to embrace applied scientists and computational geometry experts with their different views:

- The simplest case is the communication of dimensions aka the size of a region of interest in the 2D plane. In this case, communicating the alignment with axes is maybe not as relevant as it is to report the area of the ROI.
- In other cases the extent of the parallelogram is relevant though.
- Finally, in CAD models it should be possible to specify the polygons which the parallelograms represent with exact numerical details.

Parallelograms are important geometrical primitives as their usage for describing many scanning experiments shows where typically parallelogram-shaped ROIs are scanned across the surface of a sample.

The term parallelogram will be used throughout this base class thus including the important special cases rectangle, square, 2D box, axis-aligned bounding box (AABB), or optimal bounding box (OBB) as analogous 2D variants to their 3D counterparts. See *NXcg_hexahedron* for the generalization in 3D.

An axis-aligned bounding box is a common data object in computational science and simulation codes to represent a rectangle whose edges are aligned with the axes of a coordinate system. As a part of binary trees AABBs are important data objects for executing 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, i.e. most tightly fitting box about an arbitrary object. In general such boxes are rotated. Other than in 3D dimensions, the rotation caliper method offers a rigorous approach to compute an optimal bounding box to a point set 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

Structure:

is_rectangle: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

To specify which parallelogram is a rectangle.

is_axis_aligned: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

Only to be used if is_rectangle is present. In this case, this field describes whether parallelograms are rectangles whose primary edges are parallel to the axes of the coordinate system.

parallelograms: (optional) *NXcg_face_list_data_structure*

Combined storage of all parallelograms.

parallelogramID: (optional) *NXcg_face_list_data_structure*

Individual storage of each parallelogram.

Hypertext Anchors

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

- */NXcg_parallelogram/is_axis_aligned-field*
- */NXcg_parallelogram/is_rectangle-field*
- */NXcg_parallelogram/parallelogramID-group*
- */NXcg_parallelogram/parallelograms-group*

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

NXcg_point

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of points.

Points may have an associated time value. Users are advised though to store time data of point sets rather as instances of time events, where for each point in time there is an *NXcg_point* 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 (real time or simulated physical time). Thereby, the storage of redundant timestamp information per point is considered as obsolete.

Symbols:

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

d: The dimensionality.

c: The cardinality of the set, i.e. the number of points.

Groups cited:
none

Structure:

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

Coordinates of the points.

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

(Elapsed) time for each point.

If the field time is needed contextualize the time_offset relative to which time values are defined. Alternative store timestamp.

timestamp: (optional) *NX_DATE_TIME* (Rank: 1, Dimensions: [c])

ISO8601 with local time zone offset for each point.

time_offset: (optional) *NX_DATE_TIME*

ISO8601 with local time zone offset that serves as the reference for values in the field time.

Hypertext Anchors

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

- */NXcg_point/position-field*
- */NXcg_point/time-field*
- */NXcg_point/time_offset-field*
- */NXcg_point/timestamp-field*

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

NXcg_polygon

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of polygons in Euclidean space.

Polygons 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* base class if they wish to describe closed polyhedra. Even more general complexes can be thought of. An example are the so-called piecewise-linear complexes used in the TetGen library.

As these complexes can have holes though, polyhedra without holes are one subclass of such complexes, users should rather design their own base class e.g. NXcg_polytope to describe such even more complex primitives instead of abusing this base class for such purposes.

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_half_edge_data_structure*

Structure:

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

The total number of vertices in the set.

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

For each polygon its accumulated length along its edges.

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

Interior angles for each polygon. There are as many values per polygon as there are 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

polygons: (optional) *NXcg_face_list_data_structure*

Combined storage of all primitives of all polygons.

polygonID: (optional) *NXcg_face_list_data_structure*

Individual storage of the mesh of each polygon.

polygon_half_edgeID: (optional) *NXcg_half_edge_data_structure*

Individual storage of each polygon as a graph.

Hypertext Anchors

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

- */NXcg_polygon/edge_length-field*
- */NXcg_polygon/interior_angle-field*
- */NXcg_polygon/number_of_total_vertices-field*
- */NXcg_polygon/polygon_half_edgeID-group*
- */NXcg_polygon/polygonID-group*
- */NXcg_polygon/polygons-group*
- */NXcg_polygon/shape-field*

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

NXcg_polyhedron

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of polyhedra in Euclidean space.

Polyhedra or so-called cells (especially in the convex of tessellations) are constructed from polygon meshes. Polyhedra may make contact to allow a usage of this base class for a description of tessellations.

For the description of more complicated manifolds and especially for polyhedra with holes, users are advised to check if their particular needs are described by creating customized instances of an *NXcg_polygon*.

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*

Structure:

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

The number of faces for each polyhedron. Faces of adjoining polyhedra are counted for each polyhedron.

face_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_f_total]) {units=*NX_AREA*}

Area of each of faces.

number_of_edges: (optional) *NX_UINT*

The number of edges for each polyhedron. Edges of adjoining polyhedra are counted for each polyhedron.

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

Length of each edge.

polyhedra: (optional) *NXcg_face_list_data_structure*

Combined storage of all primitives of all polyhedra.

polyhedronID: (optional) *NXcg_face_list_data_structure*

Individual storage of each polyhedron.

polyhedron_half_edgeID: (optional) *NXcg_half_edge_data_structure*

Individual storage of each polygon as a graph.

Hypertext Anchors

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

- */NXcg_polyhedron/edge_length-field*
- */NXcg_polyhedron/face_area-field*
- */NXcg_polyhedron/number_of_edges-field*
- */NXcg_polyhedron/number_of_faces-field*
- */NXcg_polyhedron/polyhedra-group*
- */NXcg_polyhedron/polyhedron_half_edgeID-group*
- */NXcg_polyhedron/polyhedronID-group*

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

NXcg_polyline

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of polylines.

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 traversal along the polyline when walking from the first to the 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:
none

Structure:

depends_on: (optional) *NX_CHAR* <=

Reference to an instance of *NXcg_point* which defines the location of the vertices that are referred to in this NXcg_polyline instance.

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

The total number of vertices that have different positions.

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

The total number of vertices, irrespective of their eventual uniqueness.

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

The total number of vertices of each polyline, irrespectively whether vertices are shared by vertices or not.

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

Positions of the vertices which support the members of the polyline set.

Users are encouraged to reduce the vertices to unique positions and vertices as this often supports with storing geometry data more efficiently. 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 stores each vertex of a triangle mesh even though many vertices are shared between triangles and thus storing multiple copies of their positions is redundant.

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 several times.

polylines: (optional) *NX_INT* (Rank: 1, Dimensions: [n_total]) {units=*NX_UNITLESS*}

Sequence of identifier for vertices how they build each polyline.

A trivial example is a set with two polylines with three vertices each. If the polylines meet at a vertex (assume for example that the second vertex is shared and marking the junction between the two polylines), it is possible that there are only five unique positions. This suggests to store five unique vertices.

A non-trivial example is a set with several polylines. Assume that each has a different number of vertices. The array stores the identifier of the vertices in the sequence how the polylines are visited:

The first entry is the identifier of the first vertex of the first polyline, followed by the second vertex of the first polyline, until the last vertex of the first polyline. Thereafter, the first vertex of the second polyline, and so on and so forth. Using the (cumulated) counts in number_of_vertices (n_i^v), the vertices of the N-th polyline can be accessed on the array index interval $[\sum_{i=0}^{i=N-1} n_i^v, \sum_{i=0}^{i=N} n_i^v]$.

Hypertext Anchors

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

- */NXcg_polyline/depends_on-field*
- */NXcg_polyline/number_of_total_vertices-field*
- */NXcg_polyline/number_of_unique_vertices-field*
- */NXcg_polyline/number_of_vertices-field*
- */NXcg_polyline/polylines-field*
- */NXcg_polyline/vertices-field*
- */NXcg_polyline/vertices_are_unique-field*

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

NXcg_primitive

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of primitives in Euclidean space.

Primitives must neither be degenerated nor self-intersect. Individual primitives can differ in their properties (e.g. size, shape, rotation).

Symbols:

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

d: The dimensionality of the embedding space.

c: The cardinality of the set, i.e. the number of members.

Groups cited:
NXcg_unit_normal

Structure:

depends_on: (optional) *NX_CHAR*

Reference to an instance of *NXcoordinate_system* in which these primitives are defined.

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

The dimensionality of the primitive set with value up to d.

Any of these values: 1 | 2 | 3

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

The cardinality of the primitive set. Value should be equal to c.

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

Integer offset whereby the identifier of the first member of the set differs from zero.

Indices can be used as identifiers and thus names of instances.

Identifiers can be defined either implicitly or explicitly. For implicit indexing identifiers are defined on the interval $[index_offset, index_offset + c - 1]$.

Therefore, implicit identifier are completely defined by the value of index_offset and cardinality. For example if identifier run from -2 to 3 the value for index_offset is -2.

For explicit indexing the field identifier has to be used. Fortran-/Matlab- and C-/Python-style indexing have specific implicit identifier conventions where index_offset is 1 and 0 respectively.

indices: (optional) *NX_INT* (Rank: 1, Dimensions: [c])

Identifier of each member for explicit indexing.

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

The center of each primitive

is_center_of_mass: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

True if the center is a center of mass.

shape: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Shape of each primitive

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

Length of each primitive

Often the term is associated with the assumption that one edge is parallel to an axis of the coordinate system.

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

Width of each primitive

Often the term is associated with the assumption that one edge is parallel to an axis of the coordinate system.

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

Height of each primitive

Often the term is associated with the assumption that one edge is parallel to an axis of the coordinate system.

is_closed: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [c])

True if primitive is closed such that it has properties like area or volume.

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

Volume of each primitive.

Set to NaN if does not apply for primitives for which is_closed is False. Volume is an N-D concept for values of dimensionality larger than 1, Area is an alias for the two-dimensional case.

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [c]) {units=*NX_AREA*}

Alias for surface_area of each primitive.

Set to NaN if does not apply for primitives for which is_closed is False.

orientation: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_DIMENSIONLESS*}

Direction unit vector which points along the longest principal axis of each primitive.

Use the depends_on attribute to specify in which coordinate system these direction unit vectors are defined.

is_mesh: (optional) *NX_BOOLEAN*

Do the primitives define a mesh.

is_triangle_mesh: (optional) *NX_BOOLEAN*

Do the primitives define a triangle mesh or not.

is_surface_mesh: (optional) *NX_BOOLEAN*

Do the primitives discretize the surface of an object or not.

is_geodesic_mesh: (optional) *NX_BOOLEAN*

Do the primitives define a geodesic mesh or not.

A geodesic surface mesh is a triangulated surface mesh with metadata which can be used as an approximation to describe the surface of a sphere. Triangulation of spheres are commonly used in Materials Science for quantifying texture of materials, i.e. the relative rotation of crystals to sample directions.

For additional details or an introduction into the topic of geodesic meshes see (from which specifically the section on subdivision schemes is relevant).

- E. S. Popko and C. J. Kitrick

Earth scientists 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 software. For now we propose to use this base class as a container for organizing data related to geodesic meshes.

Specifically an instance of this base class should detail the rule set how e.g. a geodesic (surface) mesh was instantiated as there are many possibilities to do so.

description: (optional) *NX_CHAR*

Possibility to store details such as when primitives form a (specific) type of mesh such as geodesic meshes.

vertex_normal: (optional) *NXcg_unit_normal*

edge_normal: (optional) *NXcg_unit_normal*

face_normal: (optional) *NXcg_unit_normal*

Hypertext Anchors

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

- */NXcg_primitive/area-field*
- */NXcg_primitive/cardinality-field*
- */NXcg_primitive/center-field*
- */NXcg_primitive/depends_on-field*
- */NXcg_primitive/description-field*
- */NXcg_primitive/dimensionality-field*
- */NXcg_primitive/edge_normal-group*
- */NXcg_primitive/face_normal-group*
- */NXcg_primitive/height-field*
- */NXcg_primitive/index_offset-field*
- */NXcg_primitive/indices-field*
- */NXcg_primitive/is_center_of_mass-field*
- */NXcg_primitive/is_closed-field*
- */NXcg_primitive/is_geodesic_mesh-field*
- */NXcg_primitive/is_mesh-field*
- */NXcg_primitive/is_surface_mesh-field*
- */NXcg_primitive/is_triangle_mesh-field*
- */NXcg_primitive/length-field*
- */NXcg_primitive/orientation-field*
- */NXcg_primitive/shape-field*
- */NXcg_primitive/vertex_normal-group*
- */NXcg_primitive/volume-field*
- */NXcg_primitive/width-field*

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

NXcg_roi

Status:

base class, extends *NXobject*

Description:

Base class for a region-of-interest (ROI) bound by geometric primitives.

So-called region-of-interest(s) (ROIs) are typically used to describe a region in space (and time) where an observation is made or for which a computer simulation is performed with given boundary conditions.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays. Use *NXcg_primitive* and *NXcoordinate_system* classes to define explicitly the reference frame in which the primitives are defined.

Groups cited:
NXcg_cylinder, *NXcg_ellipsoid*, *NXcg_hexahedron*, *NXcg_parallelogram*, *NXcg_polyhedron*

Structure:

CG_ELLIPSOID: (optional) *NXcg_ellipsoid*

CG_CYLINDER: (optional) *NXcg_cylinder*

CG_PARALLELOGRAM: (optional) *NXcg_parallelogram*

CG_HEXAHEDRON: (optional) *NXcg_hexahedron*

CG_POLYHEDRON: (optional) *NXcg_polyhedron*

Hypertext Anchors

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

- */NXcg_roi/CG_CYLINDER-group*
- */NXcg_roi/CG_ELLIPSOID-group*
- */NXcg_roi/CG_HEXAHEDRON-group*
- */NXcg_roi/CG_PARALLELOGRAM-group*
- */NXcg_roi/CG_POLYHEDRON-group*

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

NXcg_tetrahedron

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of tetrahedra.

Among hexahedral elements, tetrahedral elements are one of the most frequently used geometric primitive for meshing and describing volumetric objects in continuum-field simulations.

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*

Structure:

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.

tetrahedra: (optional) *NXcg_face_list_data_structure*

Combined storage of all primitives of all tetrahedra.

tetrahedronID: (optional) *NXcg_face_list_data_structure*

Individual storage of each tetrahedron.

tetrahedron_half_edgeID: (optional) *NXcg_half_edge_data_structure*

Individual storage of each tetrahedron as a graph.

Hypertext Anchors

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

- */NXcg_tetrahedron/edge_length-field*
- */NXcg_tetrahedron/face_area-field*
- */NXcg_tetrahedron/tetrahedra-group*
- */NXcg_tetrahedron/tetrahedron_half_edgeID-group*
- */NXcg_tetrahedron/tetrahedronID-group*

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

NXcg_triangle

Status:

base class, extends *NXcg_primitive*

Description:

Computational geometry description of a set of triangles.

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

Structure:

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

Number of unique vertices in the triangle set.

edge_length: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_LENGTH*}

Length of the edges of each triangle.

For each triangle values are reported via traversing the vertices in the sequence as these are defined.

interior_angle: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, 3]) {units=*NX_ANGLE*}

Interior angles of each triangle.

For each triangle values are reported for the angle opposite to the respective edges in the sequence how vertices are defined.

triangles: (optional) *NXcg_face_list_data_structure*

Combined storage of all primitives of all triangles.

This description resembles the typical representation of primitives in file formats such as OFF, PLY, VTK, or STL.

triangleID: (optional) *NXcg_face_list_data_structure*

Individual storage of each triangle. Users are advised that using such individual storage of primitives may be less storage efficient than creating a combined storage.

Hypertext Anchors

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

- */NXcg_triangle/edge_length-field*
- */NXcg_triangle/interior_angle-field*
- */NXcg_triangle/number_of_unique_vertices-field*
- */NXcg_triangle/triangleID-group*
- */NXcg_triangle/triangles-group*

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

NXcg_unit_normal

Status:

base class, extends *NXobject*

Description:

Computational geometry description of a set of (oriented) unit normal vectors.

Store normal vector information as properties of primitives. Use only only as a child of an instance of *NXcg_primitive* so that this instance acts as the parent to define a context.

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:

normals: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [c, d]) {units=*NX_LENGTH*}

Direction of each normal - a unit normal.

orientation: (optional) *NX_INT* (Rank: 1, Dimensions: [c]) {units=*NX_UNITLESS*}

An indicator which details the orientation of each normal vector in relation to its primitive, assuming the object is viewed from a position outside the object.

- 0 - undefined
- 1 - outer unit normal vector
- 2 - inner unit normal vector

Hypertext Anchors

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

- */NXcg_unit_normal/normals-field*
- */NXcg_unit_normal/orientation-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXcg_unit_normal.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 formula mass or number of atoms, depending on the normalization stated in the normalization field.

ELEMENT: (optional) *NXatom*

If this group is used to report the composition of elements from the periodic table, the group should use the chemical symbol of that element. For other case the group name is unconstrained.

amount: (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/amount-field*
- */NXchemical_composition/ELEMENT/composition-field*
- */NXchemical_composition/ELEMENT/composition_errors-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:

No symbol table

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 connected 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

NXcollectioncolumn

Status:

base class, extends *NXcomponent*

Description:

Electron collection column of an electron analyzer.

Symbols:

No symbol table

Groups cited:
NXaperture, *NXdeflector*, *NXelectromagnetic_lens*

Structure:

scheme: (optional) *NX_CHAR*

Scheme of the electron collection lens, i.e. angular dispersive, spatial dispersive, momentum dispersive, non-dispersive, 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

lens_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`

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

Acceptance angle of the collection column.

This concept is related to term 7.4 of the ISO 18115-1:2023 standard.

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

Acceptance length or area of the collection column.

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

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

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/angular_acceptance-field*
- */NXcollectioncolumn/APERTURE-group*
- */NXcollectioncolumn/DEFLECTOR-group*
- */NXcollectioncolumn/ELECTROMAGNETIC_LENS-group*
- */NXcollectioncolumn/extractor_current-field*
- */NXcollectioncolumn/extractor_voltage-field*
- */NXcollectioncolumn/lens_mode-field*
- */NXcollectioncolumn/magnification-field*

- */NXcollectioncolumn/projection-field*
- */NXcollectioncolumn/scheme-field*
- */NXcollectioncolumn/spatial_acceptance-field*
- */NXcollectioncolumn/working_distance-field*

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

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 of an electron optical setup.

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

Aberration correction components are especially important for (scanning) transmission electron microscopy. Composed of multiple lenses and multipole stigmators, their technical details are specific for

the technology partner as well as the microscope and instrument. 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

Although this base class currently provides concepts that are foremost used in the field of electron microscopy using this base class is not restricted to this research field. NXcorrector_cs can also serve as a container to detail, in combination with *NXaberration*, about measured aberrations in classical optics. In optics, though, the difference is that the design of the :ref:NXoptical_lens` itself (e.g., using aspheric lenses or combinations of lenses) enables to reduce spherical aberrations.

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*, *NXelectromagnetic_lens*, *NXimage*, *NXoptical_lens*, *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 modeled 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*

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

OPTICAL_LENS: (optional) *NXoptical_lens*

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/ELECTROMAGNETIC_LENS-group*
- */NXcorrector_cs/OPTICAL_LENS-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/GEOMETRY-group*
- */NXcrystal/is_cylindrical-field*
- */NXcrystal/mosaic_horizontal-field*
- */NXcrystal/mosaic_vertical-field*
- */NXcrystal/OFF_GEOMETRY-group*
- */NXcrystal/order_no-field*
- */NXcrystal/orientation_matrix-field*
- */NXcrystal/polar_angle-field*
- */NXcrystal/reflection-field*
- */NXcrystal/reflectivity-group*
- */NXcrystal/scattering_vector-field*
- */NXcrystal/segment_columns-field*
- */NXcrystal/segment_gap-field*
- */NXcrystal/segment_height-field*
- */NXcrystal/segment_rows-field*
- */NXcrystal/segment_thickness-field*
- */NXcrystal/segment_width-field*
- */NXcrystal/shape-group*
- */NXcrystal/space_group-field*
- */NXcrystal/temperature-field*
- */NXcrystal/temperature_coefficient-field*
- */NXcrystal/temperature_log-group*
- */NXcrystal/thickness-field*
- */NXcrystal/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_processor*, *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.

processorID: (optional) *NXcs_processor*

Multiple instances should be named processor1, processor2, etc.

memoryID: (optional) *NXcs_memory*

Multiple instances should be named memory1, memory2, etc.

storageID: (optional) *NXcs_storage*

Multiple instances should be named storage1, storage2, etc.

Hypertext Anchors

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

- */NXcs_computer/memoryID-group*
- */NXcs_computer/name-field*
- */NXcs_computer/operating_system-field*
- */NXcs_computer/operating_system@version-attribute*
- */NXcs_computer/processorID-group*
- */NXcs_computer/storageID-group*
- */NXcs_computer/uuid-field*

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

NXcs_filter_boolean_mask

Status:

base class, extends *NXobject*

Description:

Base class for packing and unpacking booleans.

The field mask should be constructed from packing a vector of booleans (a bitfield) into unsigned integers with bytesize bitdepth. Padding to an integer number of such integers is assumed.

Thereby, this base class can be used to inform software about necessary modulo operations to decode the mask to recover e.g. set membership of objects in sets whose membership has been encoded as a vector of booleans.

This is useful e.g. when processing object sets such as point cloud data. If e.g. a spatial filter has been applied to a set of points, we may wish to document memory-space efficiently which points were analyzed. An array of boolean values is one option to achieve this. A value is true if the point is included and false otherwise.

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:

depends_on: (optional) *NX_CHAR*

Possibility to refer to which set this mask applies.

If depends_on is not provided, it is assumed that the mask applies to its direct parent.

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* {units=*NX_UNITLESS*}

The content of the mask. If padding is used, padding bits have to be set to 0.

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/depends_on-field*
- */NXcs_filter_boolean_mask/mask-field*
- */NXcs_filter_boolean_mask/number_of_objects-field*

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

Typically, computers have multiple instances of memory.

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_prng

Status:

base class, extends *NXobject*

Description:

Computer science description of pseudo-random number generator.

The purpose of this base class is to identify if exactly the same sequence can be reproduced, like for a PRNG or not, like for a true physically random source.

Symbols:

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

Groups cited:
NXprogram

Structure:

type: (optional) *NX_CHAR*

Physical approach or algorithm whereby random numbers are generated.

Different approaches for generating random numbers with a computer exists. Some use a dedicated physical device whose the state is unpredictable physically. Some use a strategy of mangling information from the system clock. Also in this case the sequence is not reproducible without having additional pieces of information.

In most cases though so-called pseudo-random number generator (PRNG) algorithms are used. These yield a deterministic sequence of practically randomly appearing numbers. These algorithms differ in their quality in how random the resulting sequences actually are, i.e. sequentially uncorrelated. Nowadays one of the most commonly used algorithm is the MersenneTwister (mt19937).

Any of these values: `physical|system_clock|mt19937|other`

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

Parameter of the PRNG controlling its initialization and thus controlling the specific sequence generated.

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

Number of initial draws from the PRNG after its initialized with the seed. These initial draws are typically discarded in an effort to equilibrate the sequence. If no warmup was performed or if warmup procedures are unclear, users should set the value to zero.

PROGRAM: (optional) *NXprogram*

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.

Hypertext Anchors

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

- */NXcs_prng/PROGRAM-group*
- */NXcs_prng/seed-field*
- */NXcs_prng/type-field*
- */NXcs_prng/warmup-field*

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

NXcs_processor

Status:

base class, extends *NXcomponent*

Description:

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

Examples are e.g. classical so-called central processing units (CPUs), coprocessors, 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 e.g.

- pu, processing core or hyper-threading core
- cpu, (multi-)core central processing unit

- gpu, (multi-)core general purpose processing unit
- fpga, field programmable gate array

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

clock_speed: (optional) *NX_NUMBER*

Clock speed of the circuit

Hypertext Anchors

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

- */NXcs_processor/CIRCUIT-group*
- */NXcs_processor/CIRCUIT/clock_speed-field*
- */NXcs_processor/CIRCUIT/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXcs_processor.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, i.e. an 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.

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

The number of nominal processes that the app invoked at runtime.

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 one. If the app exclusively uses GPU parallelization, number_of_gpus can be larger than one. If no GPU is used, number_of_gpus is zero, even though the hardware may have GPUs installed.

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

The number of nominal threads that the app invoked at runtime. Specifically here the maximum number of threads used for the high-level threading library used (e.g. OMP_NUM_THREADS), posix.

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

The number of nominal GPUs that the app 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.

eventID: (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. ID is an increasing unsigned integer starting at 1.

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/current_working_directory-field*
- */NXcs_profiling/end_time-field*
- */NXcs_profiling/eventID-group*
- */NXcs_profiling/max_gpus-field*
- */NXcs_profiling/max_processes-field*
- */NXcs_profiling/max_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.

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

The number of nominal processes that the app invoked during the execution of this event.

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 one. If the app exclusively uses GPU parallelization, number_of_gpus can be larger than one. If no GPU is used, number_of_gpus is zero, even though the hardware may have GPUs installed.

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

The number of nominal threads that the app invoked at during the execution of this event. Specifically here the maximum number of threads used for the high-level threading library used (e.g. OMP_NUM_THREADS), posix.

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

The number of nominal GPUs that the app invoked 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_gpus-field*
- */NXcs_profiling_event/max_processes-field*
- */NXcs_profiling_event/max_resident_memory_snapshot-field*
- */NXcs_profiling_event/max_threads-field*
- */NXcs_profiling_event/max_virtual_memory_snapshot-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`
- `optical`
- `tape`

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

Total amount of data which the medium can hold.

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

Maximum read rate of the storage medium.

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

Maximum write rate of the storage medium.

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/max_read_rate-field*
- */NXcs_storage/CIRCUIT/max_write_rate-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. NXdata 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 0.057 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 0.057 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"
  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")
```

(continues on next page)

(continued from previous page)

```
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 0.024 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
```

(continues on next page)

(continued from previous page)

```
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 0.026 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)
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 0.036 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"
```

(continues on next page)

(continued from previous page)

```
    @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]
```

(continues on next page)

(continued from previous page)

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

plt.show()
```

Total running time of the script: (0 minutes 0.030 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 are 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)
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 0.023 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]
  @y_encoder_indices = 1           # or [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, 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*

DEPRECATED: Use the group `axes` attribute (NIAC2014)

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*

DEPRECATED: Use the group `signal` attribute (NIAC2014)

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*

DEPRECATED: Use the group `axes` attribute (NIAC2014)

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:

Component of an electron analyzer that deflects the paths of electrons. This includes electrostatic and electromagnetic deflectors.

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 order multipoles, 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.

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

Spatial offset of the deflector in x direction (perpendicular to `offset_y`).

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

Spatial offset of the deflector in y direction (perpendicular to `offset_x`).

Hypertext Anchors

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

- */NXdeflector/current-field*
- */NXdeflector/name-field*
- */NXdeflector/offset_x-field*
- */NXdeflector/offset_y-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*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

Obligatory value: 3

@primary: (optional) *NX_POSINT*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

Obligatory value: 1

@long_name: (optional) *NX_CHAR*

Total time of flight

raw_time_of_flight: (optional) *NX_INT* (Rank: 1, Dimensions: [tof+1]) {units=*NX_PULSES*}

In DAQ clock pulses

@frequency: (optional) *NX_NUMBER*

Clock frequency in Hz

detector_number: (optional) *NX_INT*

Identifier for detector (pixels) Can be multidimensional, if needed

data: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [nP, i, j, tof]) {units=*NX_ANY*}

Data values from the detector. The rank and dimension ordering should follow a principle of slowest to fastest measurement axes and may be explicitly specified in application definitions.

Mechanical scanning of objects (e.g. sample position/angle, incident beam energy, etc) tends to be the slowest part of an experiment and so any such scan axes should be allocated to the first dimensions of the array. Note that in some cases it may be useful to represent a 2D set of scan points as a single scan-axis in the data array, especially if the scan pattern doesn't fit a rectangular array nicely. Repetition of an experiment in a time series tends to be used similar to a slow scan axis and so will often be in the first dimension of the data array.

The next fastest axes are typically the readout of the detector. A point detector will not add any dimensions (as it is just a single value per scan point) to the data array, a strip detector will add one dimension, an imaging detector will add two dimensions (e.g. X, Y axes) and detectors outputting higher dimensional data will add the corresponding number of dimensions. Note that the detector dimensions don't necessarily have to be written in order of the actual readout speeds - the slowest to fastest rule principle is only a guide.

Finally, detectors that operate in a time-of-flight mode, such as a neutron spectrometer or a silicon drift detector (used for X-ray fluorescence) tend to have their dimension(s) added to the last dimensions in the data array.

The type of each dimension should should follow the order of scan points, detector pixels, then time-of-flight (i.e. spectroscopy, spectrometry). The rank and dimension sizes (see symbol list) shown here are merely illustrative of coordination between related datasets.

@long_name: (optional) *NX_CHAR*

Title of measurement

@check_sum: (optional) *NX_INT*

Integral of data as check of data integrity

data_errors: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [nP, i, j, tof]) {units=*NX_ANY*} <=

The best estimate of the uncertainty in the data value (array size should match the data field). Where possible, this should be the standard deviation, which has the same units as the data. The form data_error is deprecated.

x_pixel_offset: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@axis: (optional) *NX_POSINT*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

Obligatory value: 1

@primary: (optional) *NX_POSINT*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

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*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

Obligatory value: 2

@primary: (optional) *NX_POSINT*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

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*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

Obligatory value: 3

@primary: (optional) *NX_POSINT*

DEPRECATED: see: https://github.com/nexusformat/definitions/issues/436

Obligatory value: 1

@long_name: (optional) *NX_CHAR*

y-axis offset from detector center

distance: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [nP, i, j]) {units=*NX_LENGTH*}

This is the distance to the previous component in the instrument; most often the sample. The usage depends on the nature of the detector: Most often it is the distance of the detector assembly. But there are irregular detectors. In this case the distance must be specified for each detector pixel.

Note, it is recommended to use NXtransformations instead.

polar_angle: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [nP, i, j]) {units=*NX_ANGLE*}

This is the polar angle of the detector towards the previous component in the instrument; most often the sample. The usage depends on the nature of the detector. Most often it is the polar_angle of the detector assembly. But there are irregular detectors. In this case, the polar_angle must be specified for each detector pixel.

Note, it is recommended to use NXtransformations instead.

azimuthal_angle: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [nP, i, j]) {units=*NX_ANGLE*}

This is the azimuthal angle angle of the detector towards the previous component in the instrument; most often the sample. The usage depends on the nature of the detector. Most often it is the azimuthal_angle of the detector assembly. But there are irregular detectors. In this case, the azimuthal_angle must be specified for each detector pixel.

Note, it is recommended to use NXtransformations instead.

description: (optional) *NX_CHAR* <=

name/manufacturer/model/etc. information

serial_number: (optional) *NX_CHAR*

Serial number for the detector

local_name: (optional) *NX_CHAR*

Local name for the detector

solid_angle: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_SOLID_ANGLE*}

Solid angle subtended by the detector at the sample

x_pixel_size: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_LENGTH*}

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

y_pixel_size: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_LENGTH*}

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

dead_time: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [nP, i, j]) {units=*NX_TIME*}

Detector dead time

gas_pressure: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_PRESSURE*}

Detector gas pressure

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

maximum drift space dimension

crate: (optional) *NX_INT* (Rank: 2, Dimensions: [i, j])

Crate number of detector

@local_name: (optional) *NX_CHAR*

Equivalent local term

slot: (optional) *NX_INT* (Rank: 2, Dimensions: [i, j])

Slot number of detector

@local_name: (optional) *NX_CHAR*

Equivalent local term

input: (optional) *NX_INT* (Rank: 2, Dimensions: [i, j])

Input number of detector

@local_name: (optional) *NX_CHAR*

Equivalent local term

type: (optional) *NX_CHAR*

Description of type such as He3 gas cylinder, He3 PSD, scintillator, fission chamber, proportion counter, ion chamber, ccd, pixel, image plate, CMOS, …

real_time: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [nP, i, j]) {units=*NX_TIME*}

Real-time of the exposure (use this if exposure time varies for each array element, otherwise use `count_time` field).

Most often there is a single real time value that is constant across an entire image frame. In such cases, only a 1-D array is needed. But there are detectors in which the real time changes per pixel. In that case, more than one dimension is needed. Therefore the rank of this field should be less than or equal to (detector rank + 1).

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

start time for each frame, with the `start` attribute as absolute reference

@start: (optional) *NX_DATE_TIME*

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

stop time for each frame, with the `start` attribute as absolute reference

@start: (optional) *NX_DATE_TIME*

calibration_date: (optional) *NX_DATE_TIME*

date of last calibration (geometry and/or efficiency) measurements

layout: (optional) *NX_CHAR*

How the detector is represented

Any of these values: `point` | `linear` | `area`

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

Elapsed actual counting time

sequence_number: (optional) *NX_INT* (Rank: 1, Dimensions: [nP])

In order to properly sort the order of the images taken in (for example) a tomography experiment, a sequence number is stored with each image.

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

This is the x position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute.

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

This is the y position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute.

frame_start_number: (optional) *NX_INT*

This is the start number of the first frame of a scan. In protein crystallography measurements one often scans a couple of frames on a give sample, then does something else, then returns to the same sample and scans some more frames. Each time with a new data file. This number helps concatenating such measurements.

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

The diameter of a cylindrical detector

acquisition_mode: (optional) *NX_CHAR*

The acquisition mode of the detector.

Any of these values:

- `gated`
- `triggered`
- `summed`
- `event`
- `histogrammed`
- `decimated`
- `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 $countrate_correction_lookup_table[c]$.

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, anticlockwise when facing away from the source.

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

Radius of the disk

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

Total slit height

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

Chopper phase angle

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

Time difference between timing system t0 and chopper driving clock signal

ratio: (optional) *NX_INT*

Pulse reduction factor of this chopper in relation to other choppers/fastest pulse in the instrument

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

Effective distance to the origin. Note, it is recommended to use NXtransformations instead.

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

Low and high values of wavelength range transmitted

depends_on: (optional) *NX_CHAR <=*

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 *NXem_instrument* 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*, *NXelectromagnetic_lens*, *NXfabrication*, *NXmonochromator*, *NXscan_controller*, *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 is 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.

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

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, $textalpha$-, 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. µm/eV.

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

Corresponding voltage for that energy dispersion.

FABRICATION: (optional) *NXfabrication* <=

CORRECTOR_CS: (optional) *NXcorrector_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) *NXscan_controller*

Hypertext Anchors

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

- */NXebeam_column/ACTUATOR-group*
- */NXebeam_column/APERTURE-group*
- */NXebeam_column/BEAM-group*
- */NXebeam_column/biprismID-group*
- */NXebeam_column/blankerID-group*
- */NXebeam_column/COMPONENT-group*
- */NXebeam_column/corrector_ax-group*
- */NXebeam_column/corrector_ax/value_x-field*
- */NXebeam_column/corrector_ax/value_y-field*
- */NXebeam_column/CORRECTOR_CS-group*
- */NXebeam_column/DEFLECTOR-group*
- */NXebeam_column/ELECTROMAGNETIC_LENS-group*
- */NXebeam_column/electron_source-group*
- */NXebeam_column/electron_source/emission_current-field*
- */NXebeam_column/electron_source/emitter_material-field*

- */NXebeam_column/electron_source/emitter_type-field*
- */NXebeam_column/electron_source/extraction_voltage-field*
- */NXebeam_column/electron_source/filament_current-field*
- */NXebeam_column/electron_source/lifetime-field*
- */NXebeam_column/electron_source/probe-field*
- */NXebeam_column/electron_source/voltage-field*
- */NXebeam_column/MONOCHROMATOR-group*
- */NXebeam_column/MONOCHROMATOR/applied-field*
- */NXebeam_column/MONOCHROMATOR/dispersion-field*
- */NXebeam_column/MONOCHROMATOR/FABRICATION-group*
- */NXebeam_column/MONOCHROMATOR/type-field*
- */NXebeam_column/MONOCHROMATOR/voltage-field*
- */NXebeam_column/operation_mode-field*
- */NXebeam_column/phaseplateID-group*
- */NXebeam_column/phaseplateID/type-field*
- */NXebeam_column/scan_controller-group*
- */NXebeam_column/SENSOR-group*

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

NXelectromagnetic_lens

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 pole piece, 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`

number_of_poles: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Qualitative description of the lens based on the number of pole pieces.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXelectromagnetic_lens/current-field*
- */NXelectromagnetic_lens/description-field*
- */NXelectromagnetic_lens/mode-field*
- */NXelectromagnetic_lens/name-field*
- */NXelectromagnetic_lens/number_of_poles-field*
- */NXelectromagnetic_lens/power_setting-field*
- */NXelectromagnetic_lens/type-field*
- */NXelectromagnetic_lens/voltage-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXelectromagnetic_lens.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

NXelectronanalyzer

Status:

base class, extends *NXcomponent*

Description:

Basic class for describing an electron analyzer.

This concept is related to term 12.59 of the ISO 18115-1:2023 standard.

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 while scanning a physical quantity)

n_transmission_function: Number of data points in the transmission function.

Groups cited:
NXcollectioncolumn, *NXdata*, *NXdeflector*, *NXdetector*, *NXelectromagnetic_lens*, *NXenergydispersion*, *NXfabrication*, *NXresolution*, *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

work_function: (optional) *NX_FLOAT* {units=*NX_ENERGY*}

Work function of the electron analyzer.

The work function of a uniform surface of a conductor is the minimum energy required to remove an electron from the interior of the solid to a vacuum level immediately outside the solid surface.

The kinetic energy :math:E_K of a photoelectron emitted from an energy level with binding energy E_B below the Fermi level is given by $E_K = h\nu - E_B - W = h\nu - E_B - e\phi_{\mathrm{sample}}$.

Here, $W = e\phi_{\mathrm{sample}}$ is the work function of the sample surface, which is directly proportional to the potential difference ϕ_{sample} between the electrochemical potential of electrons in the bulk and the electrostatic potential of an electron in the vacuum just outside the surface.

In PES measurements, the sample and the spectrometer (with work function $W_{\mathrm{spectr.}} = e\phi_{\mathrm{spectr.}}$) are electrically connected and therefore their Fermi levels are aligned. Due to the difference in local vacuum level between the sample and spectrometer, there however exists an electric potential difference (contact potential) $\Delta\phi = \phi_{\mathrm{sample}} - \phi_{\mathrm{spectr.}}$. The measured kinetic energy of a photoelectron in PES is therefore given by $E_K^{\mathrm{meas.}} = E_K + e\Delta\phi = h\nu - E_B - e\phi_{\mathrm{spectr.}}$.

Hence, the measured kinetic energy $E_K^{\mathrm{meas.}}$ of a photoelectron is independent of the sample work function. Nonetheless, the work function $\phi_{\mathrm{spectr.}}$ needs to be known to accurately determine the binding energy scale.

voltage_range: (optional) *NX_FLOAT* {units=*NX_VOLTAGE*}

Voltage range of the power supply. This influences the noise of the supply and thereby the energy resolution.

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 analyzer. Other variables such as temperature, manipulator angles etc. can be labeled as fast or slow in the data.

The fast axes should be listed in order of decreasing speed if they describe the same physical quantity or different components of the same quantity (e.g., [‘kx’, ‘ky’] or [‘detector_x’, ‘detector_y’]). However, axes representing different physical quantities (e.g., [‘energy’, ‘kx’]) do not need to be ordered by speed.

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.

energy_resolution: (optional) *NXresolution*

Energy resolution of the analyzer with the current setting. May be linked from an NXcalibration.

physical_quantity: (optional) *NX_CHAR* <=

Obligatory value: `energy`

resolution: (optional) *NX_FLOAT* {units=*NX_ENERGY*} <=

Minimum distinguishable energy separation in the energy spectra.

This concept is related to term 10.24 of the ISO 18115-1:2023 standard.

resolution_errors: (optional) *NX_FLOAT* {units=*NX_ENERGY*} <=

relative_resolution: (optional) *NX_FLOAT* <=

Ratio of the energy resolution of the electron analyzer at a specified energy value to that energy value.

This concept is related to term 10.7 of the ISO 18115-1:2023 standard.

momentum_resolution: (optional) *NXresolution*

Momentum resolution of the electron analyzer (FWHM)

physical_quantity: (optional) *NX_CHAR* <=

Obligatory value: `momentum`

resolution: (optional) *NX_FLOAT* {units=*NX_WAVENUMBER*} <=

resolution_errors: (optional) *NX_FLOAT* {units=*NX_WAVENUMBER*} <=

angular_resolution: (optional) *NXresolution*

Angular resolution of the electron analyzer (FWHM)

physical_quantity: (optional) *NX_CHAR* <=

Obligatory value: `angle`

resolution: (optional) *NX_FLOAT* {units=*NX_ANGLE*} <=

resolution_errors: (optional) *NX_FLOAT* {units=*NX_ANGLE*} <=

spatial_resolution: (optional) *NXresolution*

Spatial resolution of the electron analyzer (Airy disk radius)

This concept is related to term 10.14 of the ISO 18115-1:2023 standard.

physical_quantity: (optional) *NX_CHAR* <=

Obligatory value: `length`

resolution: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

resolution_errors: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

transmission_function: (optional) *NXdata* <=

Transmission function of the electron analyzer.

The transmission function (TF) specifies the detection efficiency per solid angle for electrons of different kinetic energy passing through the electron analyzer. It depends on the spectrometer geometry as well as operation settings such as lens mode and pass energy. The transmission function is usually given as relative intensity vs. kinetic energy.

The TF is used for calibration of the intensity scale in quantitative XPS. Without proper transmission correction, a comparison of results measured from the same sample using different operating modes for an instrument would show significant variations in signal intensity for the same kinetic energies.

This concept is related to term 7.15 of the ISO 18115-1:2023 standard.

@signal: (optional) *NX_CHAR* <=

Obligatory value: `relative_intensity`

@axes: (optional) *NX_CHAR* <=

Obligatory value: `['kinetic_energy']`

kinetic_energy: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_transmission_function]) {units=*NX_ENERGY*}

Kinetic energy values

relative_intensity: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_transmission_function]) {units=*NX_UNITLESS*}

Relative transmission efficiency for the given kinetic energies

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 ensembles described by the subclasses

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

Individual lenses outside the main optics ensembles described by the subclasses

FABRICATION: (optional) *NXfabrication* <=

RESOLUTION: (optional) *NXresolution*

Any other resolution not explicitly named in this base class.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXelectronanalyzer/angular_resolution-group*
- */NXelectronanalyzer/angular_resolution/physical_quantity-field*
- */NXelectronanalyzer/angular_resolution/resolution-field*
- */NXelectronanalyzer/angular_resolution/resolution_errors-field*
- */NXelectronanalyzer/COLLECTIONCOLUMN-group*
- */NXelectronanalyzer/DEFLECTOR-group*
- */NXelectronanalyzer/description-field*

- */NXelectronanalyzer/DETECTOR-group*
- */NXelectronanalyzer/ELECTROMAGNETIC_LENS-group*
- */NXelectronanalyzer/energy_resolution-group*
- */NXelectronanalyzer/energy_resolution/physical_quantity-field*
- */NXelectronanalyzer/energy_resolution/relative_resolution-field*
- */NXelectronanalyzer/energy_resolution/resolution-field*
- */NXelectronanalyzer/energy_resolution/resolution_errors-field*
- */NXelectronanalyzer/ENERGYDISPERSION-group*
- */NXelectronanalyzer/FABRICATION-group*
- */NXelectronanalyzer/fast_axes-field*
- */NXelectronanalyzer/momentum_resolution-group*
- */NXelectronanalyzer/momentum_resolution/physical_quantity-field*
- */NXelectronanalyzer/momentum_resolution/resolution-field*
- */NXelectronanalyzer/momentum_resolution/resolution_errors-field*
- */NXelectronanalyzer/name-field*
- */NXelectronanalyzer/name@short_name-attribute*
- */NXelectronanalyzer/RESOLUTION-group*
- */NXelectronanalyzer/slow_axes-field*
- */NXelectronanalyzer/spatial_resolution-group*
- */NXelectronanalyzer/spatial_resolution/physical_quantity-field*
- */NXelectronanalyzer/spatial_resolution/resolution-field*
- */NXelectronanalyzer/spatial_resolution/resolution_errors-field*
- */NXelectronanalyzer/SPINDISPERSION-group*
- */NXelectronanalyzer/transmission_function-group*
- */NXelectronanalyzer/transmission_function/kinetic_energy-field*
- */NXelectronanalyzer/transmission_function/relative_intensity-field*
- */NXelectronanalyzer/transmission_function@axes-attribute*
- */NXelectronanalyzer/transmission_function@signal-attribute*
- */NXelectronanalyzer/voltage_range-field*
- */NXelectronanalyzer/work_function-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXelectronanalyzer.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 time slot 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 $Xg = 0, Yg = 0, Zg = 0$.

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 phase_id 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), phase_id has as many entries as scan points and matching_phase has also as many entries as scan points.

phase_id: (optional) *NX_INT* (Rank: 1, Dimensions: [n_solutions]) {units=*NX_UNITLESS*}

The array phases_per_scan_point details how the phase_id and the matching_phase arrays have to be interpreted.

For the example of a single-phase material phase_id has trivial values either 0 (no solution) or 1 (solution matching sufficiently significant with the model for phase1, an instance of *NXphase*).

For the example of multi-phase material, it is possible (although not frequently required) that a pattern agrees significantly with multiple patterns. Examples are cases of pseudosymmetry, insufficiently precise and accurate calibrated systems, or usage of 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, phase_id 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 phase_id and the matching_phase arrays are interpreted. See documentation of phase_id 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.

pixel_shape: (optional) *NX_CHAR*

The shape of the polygon or polyhedron that was used for the tiling respectively tessellation of the region-of-interest into scan points.

Any of these values: `square` | `hexagon` | `cube` | `other`

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/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/phase_id-field*
- */NXem_ebsd/indexing/phaseID-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/pixel_shape-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_event_data

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 an representation that facilitates interoperability. This is the idea behind this base class.

NXem_event_data 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.

NXem_event_data 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 *NXem_event_data*.

Symbols:

No symbol table

Groups cited:
NXem_instrument, *NXimage*, *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*

EM_INSTRUMENT: (optional) *NXem_instrument*

IMAGE: (optional) *NXimage*

SPECTRUM: (optional) *NXspectrum*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXem_event_data/EM_INSTRUMENT-group*
- */NXem_event_data/end_time-field*
- */NXem_event_data/identifier_event-field*
- */NXem_event_data/identifier_sample-field*
- */NXem_event_data/IMAGE-group*
- */NXem_event_data/SPECTRUM-group*
- */NXem_event_data/start_time-field*
- */NXem_event_data/type-field*
- */NXem_event_data/USER-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXem_event_data.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 mappings. The latter pattern is the most frequently used.

For now the base class provides for scans for which the settings, binning, and energy resolution is the same for each scan point.

Symbols:

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_instrument

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 *NXem_event_data* 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*, *NXebeam_column*, *NXem_optical_system*, *NXfabrication*, *NXibeam_column*, *NXmanipulator*, *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 NXibeam_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) *NXebeam_column*

IBEAM_COLUMN: (optional) *NXibeam_column*

EM_OPTICAL_SYSTEM: (optional) *NXem_optical_system*

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 NXem_instrument 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 NXem_instrument 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 NXem_instrument 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 NXem_instrument 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.

- */NXem_instrument/ACTUATOR-group*
- */NXem_instrument/DETECTOR-group*
- */NXem_instrument/EBEAM_COLUMN-group*
- */NXem_instrument/EM_OPTICAL_SYSTEM-group*

- */NXem_instrument/FABRICATION-group*
- */NXem_instrument/gas_injector-group*
- */NXem_instrument/IBEAM_COLUMN-group*
- */NXem_instrument/location-field*
- */NXem_instrument/name-field*
- */NXem_instrument/nanoprobeID-group*
- */NXem_instrument/PUMP-group*
- */NXem_instrument/SENSOR-group*
- */NXem_instrument/stageID-group*
- */NXem_instrument/stageID/alias-field*
- */NXem_instrument/stageID/design-field*
- */NXem_instrument/stageID/position-field*
- */NXem_instrument/stageID/rotation-field*
- */NXem_instrument/stageID/rotation@comment-attribute*
- */NXem_instrument/stageID/tilt1-field*
- */NXem_instrument/stageID/tilt1@comment-attribute*
- */NXem_instrument/stageID/tilt2-field*
- */NXem_instrument/stageID/tilt2@comment-attribute*
- */NXem_instrument/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXem_instrument.nxdl.xml

NXem_interaction_volume

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.

- */NXem_interaction_volume/DATA-group*
- */NXem_interaction_volume/PROCESS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXem_interaction_volume.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:
NXem_event_data, *NXem_instrument*

Structure:

instrument: (optional) *NXem_instrument*

eventID: (optional) *NXem_event_data*

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_optical_system

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-/(angstrom^2*s).

dose_rate: (optional) *NX_NUMBER* {units=1/(angstrom^2*s)}

Nominal dose rate.

rotation: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

In the process of passing through an *NXelectromagnetic_lens* 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.

- */NXem_optical_system/camera_length-field*
- */NXem_optical_system/defocus-field*
- */NXem_optical_system/dose_management-field*
- */NXem_optical_system/dose_rate-field*
- */NXem_optical_system/dynamic_focus_correction-field*
- */NXem_optical_system/dynamic_refocusing-field*
- */NXem_optical_system/field_of_view-field*
- */NXem_optical_system/focal_length-field*
- */NXem_optical_system/magnification-field*
- */NXem_optical_system/probe-field*
- */NXem_optical_system/probe_current-field*
- */NXem_optical_system/rotation-field*
- */NXem_optical_system/semi_convergence_angle-field*
- */NXem_optical_system/tilt_correction-field*
- */NXem_optical_system/working_distance-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXem_optical_system.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

NXenergydispersion

Status:

base class, extends *NXcomponent*

Description:

Energy dispersion section of an electron analyzer.

Symbols:

No symbol table

Groups cited:
NXaperture, *NXdeflector*, *NXelectromagnetic_lens*, *NXfabrication*

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*}

Mean kinetic energy of the electrons in this energy-dispersive section of the analyzer. This term should be used for hemispherical analyzers.

This concept is related to term 12.63 of the ISO 18115-1:2023 standard.

kinetic_energy: (optional) *NX_FLOAT* {units=*NX_ENERGY*}

Kinetic energy set for this dispersive section. Can be either the set kinetic energy, or the whole calibrated energy axis of a scan.

drift_energy: (optional) *NX_FLOAT* {units=*NX_ENERGY*}

Drift energy for time-of-flight energy dispersive elements.

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

radius: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Radius of the dispersive orbit

energy_scan_mode: (optional) *NX_CHAR*

Way of scanning the energy axis

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `fixed_analyzer_transmission`: constant ΔE mode, where the electron retardation (i.e., the fraction of pass energy to kinetic energy, $R = (E_K - W)/E_p$, is scanned, but the pass energy E_p is kept constant. Here, $W = e\phi$ is the spectrometer work function (with the potential difference ϕ between the electrochemical potential of electrons in the bulk and the electrostatic potential of an electron in the vacuum just outside the surface). This mode is often used in X-ray/ultraviolet photoemission spectroscopy (XPS/UPS) because the energy resolution does not change with changing energy (due to the constant pass energy). Synonyms: constant ΔE mode, constant analyzer energy mode, CAE mode, FAT mode This concept is related to term 12.64 of the ISO 18115-1:2023 standard.
- `fixed_retardation_ratio`: constant $\Delta E/E$ mode, where the pass energy is scanned such that the electron retardation ratio is constant. In this mode, electrons of all energies are decelerated with this same fixed factor. Thus, the pass energy is proportional to the kinetic energy. This mode is often used in Auger electron spectroscopy (AES) to improve S/N for high-KE electrons, but this leads to a changing energy resolution ($\Delta E \sim E_p$) at different kinetic energies. It can however also be used in XPS. Synonyms: constant $\Delta E/E$ mode, constant retardation ratio mode, CRR mode, FRR mode This concept is related to term 12.66 of the ISO 18115-1:2023 standard.
- `fixed_energy`: In the fixed energy (FE) mode, the intensity for one single kinetic energy is measured for a specified time. This mode is particularly useful during setup or alignment of the electron analyzer, for analysis of stability of the excitation source or for sample alignment. Since the mode measures intensity as a function of time, the difference in channel signals is not of interest. Therefore, the signals from all channels are summed. Synonym: FE mode
- `snapshot`: Snapshot mode does not involve an energy scan and instead collects data from all channels of the detector without averaging. The resulting spectrum reflects the energy distribution of particles passing through the analyzer using the current settings. This mode is commonly used to position the detection energy at the maximum of a peak and record the signal, enabling faster data acquisition within a limited energy range compared to FAT. Snapshot measurements are particularly suitable for CCD and DLD detectors, which have multiple channels and can accurately display the peak shape. While five or nine-channel detectors can also be used for snapshot measurements, their energy resolution is relatively lower.
- `dither`: In dither acquisition mode, the kinetic energy of the analyzer is randomly varied by a small value around a central value and at fixed pass energy. This reduces or removes inhomogeneities of the detector efficiency, such as e.g. imposed by a mesh in front of the detector. Mostly relevant for CCD/DLD type of detectors.

tof_distance: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Length of the time-of-flight 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

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

Individual lenses in the energy dispersive section

FABRICATION: (optional) *NXfabrication* <=

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/drift_energy-field*
- */NXenergydispersion/ELECTROMAGNETIC_LENS-group*
- */NXenergydispersion/energy_interval-field*
- */NXenergydispersion/energy_scan_mode-field*
- */NXenergydispersion/FABRICATION-group*
- */NXenergydispersion/kinetic_energy-field*
- */NXenergydispersion/pass_energy-field*
- */NXenergydispersion/radius-field*
- */NXenergydispersion/scheme-field*
- */NXenergydispersion/tof_distance-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXenergydispersion.nxdl.xml

NXentry

Status:

base class, extends *NXobject*

Description:

(**required**) *NXentry* describes the measurement.

The top-level NeXus group which contains all the data and associated information that comprise a single measurement. It is mandatory that there is at least one group of this type in the NeXus file.

Symbols:

No symbol table

Groups cited:

NXcollection, *NXdata*, *NXinstrument*, *NXmonitor*, *NXnote*, *NXparameters*, *NXprocess*, *NXsample*, *NXsubentry*, *NXuser*

Structure:

@default: (optional) *NX_CHAR* <=

Declares which *NXdata* group contains the data to be shown by default. It is used to resolve ambiguity when one *NXdata* group exists. The value *names* a child group. If that group itself has a `default` attribute, continue this chain until an *NXdata* group is reached.

For more information about how NeXus identifies the default plottable data, see the *Find Plottable Data, v3* section.

@IDF_Version: (optional) *NX_CHAR*

ISIS Muon IDF_Version

title: (optional) *NX_CHAR*

Extended title for entry

experiment_identifier: (optional) *NX_CHAR*

Unique identifier for the experiment, defined by the facility, possibly linked to the proposals

experiment_description: (optional) *NX_CHAR*

Brief summary of the experiment, including key objectives.

collection_identifier: (optional) *NX_CHAR*

User or Data Acquisition defined group of NeXus files or NXentry

collection_description: (optional) *NX_CHAR*

Brief summary of the collection, including grouping criteria.

entry_identifier: (optional) *NX_CHAR*

Unique identifier for the measurement, defined by the facility.

entry_identifier_uuid: (optional) *NX_CHAR*

UUID identifier for the measurement.

@version: (optional) *NX_CHAR*

Version of UUID used

features: (optional) *NX_CHAR*

Reserved for future use by NIAC.

See https://github.com/nexusformat/definitions/issues/382

definition: (optional) *NX_CHAR*

(alternate use: see same field in *NXsubentry* for preferred)

Official NeXus NXDL schema to which this entry conforms which must be the name of the NXDL file (case sensitive without the file extension) that the NXDL schema is defined in.

For example the `definition` field for a file that conformed to the *NXarpes.nxdl.xml* definition must contain the string **NXarpes**.

This field is provided so that *NXentry* can be the overlay position in a NeXus data file for an application definition and its set of groups, fields, and attributes.

It is advised to use *NXsubentry*, instead, as the overlay position.

@version: (optional) *NX_CHAR*

NXDL version number

@URL: (optional) *NX_CHAR*

URL of NXDL file

definition_local: (optional) *NX_CHAR*

DEPRECATED: see same field in *NXsubentry* for preferred use

Local NXDL schema extended from the entry specified in the `definition` field. This contains any locally-defined, additional fields in the entry.

@version: (optional) *NX_CHAR*

NXDL version number

@URL: (optional) *NX_CHAR*

URL of NXDL file

start_time: (optional) *NX_DATE_TIME*

Starting time of measurement

end_time: (optional) *NX_DATE_TIME*

Ending time of measurement

duration: (optional) *NX_INT* {units=*NX_TIME*}

Duration of measurement

collection_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time transpired actually collecting data i.e. taking out time when collection was suspended due to e.g. temperature out of range

run_cycle: (optional) *NX_CHAR*

Such as "2007-3". Some user facilities organize their beam time into run cycles.

program_name: (optional) *NX_CHAR*

Name of program used to generate this file

@version: (optional) *NX_CHAR*

Program version number

@configuration: (optional) *NX_CHAR*

configuration of the program

revision: (optional) *NX_CHAR*

Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument definition format, …

@comment: (optional) *NX_CHAR*

pre_sample_flightpath: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

This is the flightpath before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

DATA: (optional) *NXdata* <=

The data group

Note

Before the NIAC2016 meeting[1], at least one *NXdata* group was required in each *NXentry* group. At the NIAC2016 meeting, it was decided to make *NXdata* an optional group in *NXentry* groups for data files that do not use an application definition. It is recommended strongly that all NeXus data files provide a NXdata group. It is 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*

DEPRECATED: Use the *type* field instead

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

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

NXfit

Status:

base class, extends *NXprocess*

Description:

Description of a fit procedure using a scalar valued global function

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

dimRank: Rank of the dependent and independent data arrays (for multivariate scalar-valued fit.)

Groups cited:
NXdata, *NXfit_function*, *NXpeak*

Structure:

label: (optional) *NX_CHAR*

Human-readable label for this fit procedure.

figure_of_meritMETRIC: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Figure-of-merit to determine the goodness of fit, i.e., how well the fit model (i.e., the set of peaks and backgrounds) fits the measured observations.

This value (which is a single number) is often used to guide adjustments to the fitting parameters in the peak fitting process.

@metric: (optional) *NX_CHAR*

Metric used to determine the goodness of fit. Examples include:

- χ^2, the squared sum of the sigma-weighted residuals
- reduced χ^2:, χ^2: per degree of freedom
- R^2, the coefficient of determination

data: (optional) *NXdata* <=

Data and results of the fit.

input_independent: (optional) *NX_NUMBER* (Rank: dimRank) {units=*NX_ANY*}

Independent variable(s) for this fit procedure, representing the values to be fitted by the `global_fit_function`.

input_dependent: (optional) *NX_NUMBER* (Rank: dimRank) {units=*NX_ANY*}

Dependent variable(s) for this fit procedure (i.e., the observed data).

fit_sum: (optional) *NX_NUMBER* (Rank: dimRank) {units=*NX_ANY*}

Resulting fit obtained by evaluating the `global_fit_function` at the points specified in `input_independent` using the optimized fit parameters. This represents the best-fit curve or surface approximating the input_dependent data.

residual: (optional) *NX_NUMBER* (Rank: dimRank) {units=*NX_ANY*}

The difference between the observed data (`input_dependent`) and the predicted fit values (`fit_sum`). A lower magnitude of residuals indicates a better fit.

peakPEAK: (optional) *NXpeak*

An instance of the peak model. If there is no characteristic name for each peak component, the peaks could be labeled as peak_0, peak_1, and so on.

total_area: (optional) *NX_NUMBER* {units=*NX_ANY*} <=

Total area under the curve (can also be used for the total area minus any background values).

relative_sensitivity_factor: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Relative sensitivity for this peak, to be used for quantification in an NXprocess.

As an example, in X-ray spectroscopy could depend on the energy scale (see position), the ionization cross section, and the element probed.

relative_area: (optional) *NX_NUMBER* {units=*NX_ANY*}

Relative area of this peak compared to other peaks.

The relative area can simply be derived by dividing the total_area by the total area of all peaks or by a more complicated method (e.g., by additionally dividing by the relative sensitivity factors). Details shall be given in *global_fit_function*.

backgroundBACKGROUND: (optional) *NXpeak*

One fitted background (functional form, position (see *data/input_independent*), and intensities) of the peak fit. If there is no characteristic name for each background component, it is envisioned that backgrounds are labeled as background_0, background_1, and so on.

global_fit_function: (optional) *NXfit_function*

Function used to describe the overall fit to the data, taking into account the parameters of the individual *NXpeak* components.

formula_description: (optional) *NX_CHAR* <=

Often, if the peaks and fit backgrounds are defined independently (i.e, with their own parameter sets), the resulting global fit is a function of the form $model = peak_1(p_1) + peak2(p_2) + backgr(p_3)$., where each p_x describes the set of parameters for one peak/background.

error_function: (optional) *NXfit_function*

Function used to optimize the parameters during peak fitting.

description: (optional) *NX_CHAR* <=

Description of the method used to optimize the parameters during peak fitting. Examples:

- least squares
- nonlinear least squares
- Levenberg-Marquardt algorithm (damped least-squares)
- linear regression
- Bayesian linear regression

formula_description: (optional) *NX_CHAR* <=

For the optimization, the formula is any optimization process on the `global_fit_function` given above. As an example, for a least squares algorithm on independent components, the formula of the `error_function` would be $LLS(peak_1(p_1) + peak_2(p_2) + backgr(p_3))$, where each p_i describes the set of parameters for one peak/background. In this case, the `formula_description` can be expressed as $min(\chi^2)$, where χ^2 is the sum of squared residuals between the model and the observed data: $min(\chi^2) = \sum_{i=1}^{N} \left(y_i - (\text{peak}_1(p_1, x_i) + \text{peak}_2(p_2, x_i) + \text{backgr}(p_3, x_i))\right)^2$

It is however also possible to supply more involved formulas (e.g., in the case of constrained fits).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXfit/backgroundBACKGROUND-group*
- */NXfit/data-group*
- */NXfit/data/fit_sum-field*
- */NXfit/data/input_dependent-field*
- */NXfit/data/input_independent-field*
- */NXfit/data/residual-field*
- */NXfit/error_function-group*
- */NXfit/error_function/description-field*
- */NXfit/error_function/formula_description-field*

- */NXfit/figure_of_meritMETRIC-field*
- */NXfit/figure_of_meritMETRIC@metric-attribute*
- */NXfit/global_fit_function-group*
- */NXfit/global_fit_function/formula_description-field*
- */NXfit/label-field*
- */NXfit/peakPEAK-group*
- */NXfit/peakPEAK/relative_area-field*
- */NXfit/peakPEAK/relative_sensitivity_factor-field*
- */NXfit/peakPEAK/total_area-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXfit.nxdl.xml

NXfit_function

Status:

base class, extends *NXobject*

Description:

This describes a fit function that is used to fit data to any functional form.

A fit function is used to describe a set of data $y_k, k = 1...M$, which are collected as a function of one or more independent variables x at the points x_k. The fit function f describes these data in an approximate way as $y_k \approx f(a_0, ...a_n, x_k)$, where $a_i, i = 0...n$ are the *fit parameters* (which are stored the instances of `NXfit_parameter`).

Symbols:

No symbol table

Groups cited:
NXparameters

Structure:

function_type: (optional) *NX_CHAR*

Type of function used.

Examples include "Gaussian" and "Lorentzian". In case a complicated functions, the the functional form of the function should be given by the `formula_description` field . The user is also encouraged to use the `description` field for describing the fit function in a human-readable way.

Application definitions may limit the allowed fit functions by using an enumeration for the `function_type` field.

description: (optional) *NX_CHAR*

Human-readable short description of this fit function. Software tools may use this field to write their local description of the fit function.

formula_description: (optional) *NX_CHAR*

Description of the mathematical formula of the function, taking into account the instances of `TERM` in `fit_parameters`.

fit_parameters: (optional) *NXparameters* <=

PARAMETER: (optional) *NX_CHAR_OR_NUMBER* {units=*NX_ANY*} <=

A parameter for a fit function. This would typically be a variable that is optimized in a fit.

@description: (optional) *NX_CHAR* <=

A description of what this parameter represents.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXfit_function/description-field*
- */NXfit_function/fit_parameters-group*
- */NXfit_function/fit_parameters/PARAMETER-field*
- */NXfit_function/fit_parameters/PARAMETER@description-attribute*
- */NXfit_function/formula_description-field*
- */NXfit_function/function_type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXfit_function.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

NXibeam_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*, *NXatom*, *NXbeam*, *NXcomponent*, *NXdeflector*, *NXelectromagnetic_lens*, *NXmonochromator*, *NXscan_controller*, *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 is 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) *NXatom*

Which elements, ions, or molecular ions form the beam. Examples are gallium, helium, neon, argon, krypton, or xenon, O2+.

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

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) *NXscan_controller*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXibeam_column/ACTUATOR-group*
- */NXibeam_column/APERTURE-group*
- */NXibeam_column/BEAM-group*
- */NXibeam_column/blankerID-group*
- */NXibeam_column/COMPONENT-group*
- */NXibeam_column/DEFLECTOR-group*
- */NXibeam_column/ELECTROMAGNETIC_LENS-group*
- */NXibeam_column/ion_source-group*
- */NXibeam_column/ion_source/brightness-field*
- */NXibeam_column/ion_source/current-field*
- */NXibeam_column/ion_source/description-field*
- */NXibeam_column/ion_source/emitter_type-field*
- */NXibeam_column/ion_source/flux-field*
- */NXibeam_column/ion_source/ion_energy_profile-field*
- */NXibeam_column/ion_source/name-field*
- */NXibeam_column/ion_source/probe-group*
- */NXibeam_column/ion_source/voltage-field*
- */NXibeam_column/MONOCHROMATOR-group*
- */NXibeam_column/operation_mode-field*
- */NXibeam_column/scan_controller-group*
- */NXibeam_column/SENSOR-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXibeam_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. Points are considered exact synonyms for pixel and voxel, which are terms used for 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.

Classically, images depict objects in real space. Such usage of NXimage essentially is equivalent to storing pictures. For this purpose the image_1d, image_2d, or image_3d NXdata instances respectively should be used such that all their axes axis_i, axis_j, axis_k are constrained to NeXus Unit Category NX_LENGTH.

Imaging modes in electron microscopy are typically more versatile, specifically for use cases in scanning transmission electron microscopy, so-called 4DSTEM. In this case, one two-dimensional diffraction image is taken for each point that gets scanned in real space. Consequently, image_3d and image_4d NXdata instances should be used for these cases with axis_k and axis_m respectively of NeXus Unit Category NX_LENGTH and axis_i and axis_j respectively of NeXus Unit Category NX_WAVENUMBER or NX_UNITLESS.

Symbols:

n_img: Number of images in the stack, for stacks the slowest dimension.

n_m: Number of image points along 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_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@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_ANY*}

Point coordinate along the fast dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@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_ANY*}

Point coordinate along the slow dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_ANY*}

Point coordinate along the fast dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fastest dimension.

image_4d: (optional) *NXdata* <=

Four-dimensional image.

intensity: (optional) *NX_NUMBER* (Rank: 4, Dimensions: [n_m, 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_m, 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_m, 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_m, 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_m: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_m]) {units=*NX_ANY*}

Point coordinate along the slowest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the slowest dimension.

axis_k: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_k]) {units=*NX_ANY*}

Point coordinate along the slow dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_ANY*}

Point coordinate along the fast dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@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_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@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_ANY*}

Point coordinate along the fast dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@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_ANY*}

Point coordinate along the slow dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the slow dimension.

axis_j: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) {units=*NX_ANY*}

Point coordinate along the fast dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@long_name: (optional) *NX_CHAR*

Point coordinate along the fast dimension.

axis_i: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Point coordinate along the fastest dimension.

Different NeXus Unit Category are allowed:

- NX_LENGTH for images slicing real space.
- NX_WAVENUMBER or NX_UNITLESS respectively for images slicing reciprocal space.

@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/image_4d-group*
- */NXimage/image_4d/axis_i-field*
- */NXimage/image_4d/axis_i@long_name-attribute*
- */NXimage/image_4d/axis_j-field*

- */NXimage/image_4d/axis_j@long_name-attribute*
- */NXimage/image_4d/axis_k-field*
- */NXimage/image_4d/axis_k@long_name-attribute*
- */NXimage/image_4d/axis_m-field*
- */NXimage/image_4d/axis_m@long_name-attribute*
- */NXimage/image_4d/complex-field*
- */NXimage/image_4d/imag-field*
- */NXimage/image_4d/intensity-field*
- */NXimage/image_4d/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*

It is recommended (effective from 2025) to use the "wavelength_shifter" choice for 3-pole wigglers, while reserving the generic "wiggler" designation for extended multipole wigglers.

Any of these values: `undulator|wiggler|wavelength_shifter`

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

NXlog

Status:

base class, extends *NXobject*

Description:

Information recorded as a function of time.

Description of information that is recorded against time. There are two common use cases for this:

- When logging data such as temperature during a run
- When data is taken in streaming mode data acquisition, i.e. just timestamp, value pairs are stored and correlated later in data reduction with other data,

In both cases, NXlog contains the logged or streamed values and the times at which they were measured as elapsed time since a starting time recorded in ISO8601 format. The time units are specified in the units attribute. An optional scaling attribute can be used to accommodate non standard clocks.

This method of storing logged data helps to distinguish instances in which a variable 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 setpoints.

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:

- `H20`
- `D20`
- `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 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]. 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 scientific 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} 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 identi-

fiers 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:0000000000000

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_lens

Status:

base class, 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 or a custom value (if you use a custom value, also set @custom=True):

- `biconcave`
- `plano-concave`
- `convexo-concave`
- `biconvex`
- `plano-convex`
- `concavo-convex`
- `Fresnel lens`

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_radius_front or curvature_radius_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.

numerical_aperture: (optional) *NX_NUMBER*

The numerical aperture of the lens.

magnification: (optional) *NX_FLOAT*

Magnification 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.

- */NXoptical_lens/Abbe_number-field*
- */NXoptical_lens/chromatic-field*
- */NXoptical_lens/COATING-group*
- */NXoptical_lens/COATING/coating_material-field*
- */NXoptical_lens/COATING/coating_thickness-field*
- */NXoptical_lens/COATING/coating_type-field*
- */NXoptical_lens/COATING/index_of_refraction_coating-field*
- */NXoptical_lens/curvature_radius_FACE-field*
- */NXoptical_lens/focal_length-field*
- */NXoptical_lens/lens_diameter-field*
- */NXoptical_lens/magnification-field*
- */NXoptical_lens/numerical_aperture-field*
- */NXoptical_lens/reflectance-field*
- */NXoptical_lens/substrate-group*
- */NXoptical_lens/substrate/index_of_refraction-field*
- */NXoptical_lens/substrate/substrate_material-field*
- */NXoptical_lens/substrate/substrate_thickness-field*
- */NXoptical_lens/transmission-field*
- */NXoptical_lens/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXoptical_lens.nxdl.xml

NXoptical_window

Status:

base class, extends *NXaperture*

Description:

A window of a cryostat, heater, vacuum chamber or a simple glass slide.

This describes cryostat windows and other possible influences for ellipsometry measurements.

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).

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 measurements should be considered as a separate experiment (with a separate NeXus file), and the reference data shall be *linked* in `reference_data_link`.

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.

Symbols:

No symbol table

Groups cited:
NXprocess

Structure:

window_effects_corrected: (optional) *NX_BOOLEAN*

Was a window correction performed? If so, describe the window correction procedure in `window_correction/procedure`.

window_effects_type: (optional) *NX_CHAR*

Type of effects due to this window on the measurement.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `interference effects`
- `light absorption`
- `light scattering`
- `other`

material: (optional) *NX_CHAR* <=

The material of the window.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `quartz`
- `diamond`
- `calcium fluoride`
- `zinc selenide`
- `thallium bromoiodide`
- `alkali halide compound`
- `Mylar`
- `other`

material_other: (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).

window_correction: (optional) *NXprocess*

Group to describe any window correction - if none performed, then omit this

procedure: (optional) *NX_CHAR*

Describe when (before or after the main measurement + time stamp in 'date') and how the window effects have been corrected, i.e. either mathematically or by performing a reference measurement. In the latter case, provide the link to to the reference data in `reference_data_file`.

reference_data_link: (optional) *NX_NUMBER*

External link to the data field in the NeXus file which describes the reference data if a reference measurement for window correction was performed.

Ideally, the reference measurement was performed on the same sample, using the same conditions as for the actual measurement, with and, if possible, without windows. It should have been conducted as close in time to the actual measurement as possible.

Ideally, the link uses the relative path with respect to the actual NeXus file.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXoptical_window/material-field*
- */NXoptical_window/material_other-field*
- */NXoptical_window/orientation_angle-field*
- */NXoptical_window/thickness-field*
- */NXoptical_window/window_correction-group*
- */NXoptical_window/window_correction/procedure-field*
- */NXoptical_window/window_correction/reference_data_link-field*
- */NXoptical_window/window_effects_corrected-field*
- */NXoptical_window/window_effects_type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXoptical_window.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' dot x, x' dot y, x' dot z, y' dot x, y' dot y, y' dot z] where "dot" is the scalar dot product (cosine of the angle between the unit vectors). The unit vectors in both the local and reference coordinates are right-handed and orthonormal.

The pair of groups NXtranslation and NXorientation together describe the position of a component.

GEOMETRY: (optional) *NXgeometry*

Link to another object if we are using relative positioning, else absent

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXorientation/GEOMETRY-group*
- */NXorientation/value-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXorientation.nxdl.xml

NXparameters

Status:

base class, extends *NXobject*

Description:

Container for parameters used in processing or analysing data.

Typically, this group is stored in a *NXprocess* group in order to contain parameters that are either inputs to or resulting from the process defined by the parent group. However, this base class can also be added to other groups for use in other contexts.

Although this base class can be used to store any kind of parameter, one possible use case is to store parameters that are refined by a fitting function or model. A number of attributes have been defined to store metadata associated with such a refinement.

Symbols:

No symbol table

Groups cited:
none

Structure:

@model: (optional) *NX_CHAR*

The name of the model used in optimizing the parameter values. Fitting packages such as LMFIT (https://lmfit.github.io/lmfit-py/) provide models, which instantiate functions to be fitted to the data. If this attribute is provided, it is assumed that all the parameters in this group are associated with this model.

PARAMETER: (optional) *NX_CHAR_OR_NUMBER*

A parameter that is used in or results from processing.

@units: (optional) *NX_CHAR*

@error: (optional) *NX_NUMBER*

The standard deviation of the parameter after optimizing its value.

@description: (optional) *NX_CHAR*

A description of what this parameter represents.

@expression: (optional) *NX_CHAR*

A string representing an expression that can be used to relate the parameter to another parameter's value. The format of this string is dependent on the program used to optimize the parameters and is not specified by NeXus.

@initial: (optional) *NX_NUMBER*

The initial value of the parameter used in optimization.

@max: (optional) *NX_NUMBER*

The upper bound of the parameter used in optimization.

@min: (optional) *NX_NUMBER*

The lower bound of the parameter used in optimization.

@vary: (optional) *NX_BOOLEAN*

True if the parameter was varied during optimization.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXparameters/PARAMETER-field*
- */NXparameters/PARAMETER@description-attribute*
- */NXparameters/PARAMETER@error-attribute*
- */NXparameters/PARAMETER@expression-attribute*
- */NXparameters/PARAMETER@initial-attribute*
- */NXparameters/PARAMETER@max-attribute*
- */NXparameters/PARAMETER@min-attribute*
- */NXparameters/PARAMETER@units-attribute*
- */NXparameters/PARAMETER@vary-attribute*
- */NXparameters@model-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 attributes 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
#
_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"
```

(continues on next page)

(continued from previous page)

```
        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/NXpdb.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 multivariate scalar-valued fit).

Groups cited:
NXdata, *NXfit_function*

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.

function: (optional) *NXfit_function*

The functional form of the peak. This could be a Gaussian, Lorentzian, Voigt, etc.

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/function-group*
- */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 phase_id 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 phase_id 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 behavior 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 optimization 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 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 behavior 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:

The *NXprocess* class describes an operation used to process data as part of an analysis workflow, providing information such as the software used, the date of the operation, the input parameters, and the resulting data.

Symbols:

No symbol table

Groups cited:
NXdata, *NXnote*, *NXparameters*

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.

PARAMETERS: (optional) *NXparameters* <=

Parameters used in performing the data analysis.

DATA: (optional) *NXdata* <=

The data resulting from the operation.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXprocess/DATA-group*
- */NXprocess/date-field*
- */NXprocess/NOTE-group*
- */NXprocess/PARAMETERS-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:

No symbol table

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 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 or a custom value (if you use a custom value, also set @custom=True):

- `membrane`
- `rotary_vane`
- `roots`
- `turbo_molecular`
- `ion`
- `cryo`
- `diffusion`
- `scroll`

base_pressure: (optional) *NX_FLOAT* {units=*NX_PRESSURE*}

The minimum pressure achievable in a chamber after it has been pumped down for an extended period.

medium: (optional) *NX_CHAR*

The material being moved by the pump.

Pumps intending to create a vacuum should state "vacuum" as the medium, while pumps having the primary purpose of creating a flow or pressure of gas should state "gas" as the medium.

Any of these values: `vacuum` | `liquid` | `gas` | `slurry` | `powder`

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXpump/base_pressure-field*
- */NXpump/design-field*
- */NXpump/medium-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) *NX_CHAR*

Describes the dataset

predicted_y: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The y position at which the bragg peak of the reflection is predicted

@description: (optional) *NX_CHAR*

Describes the dataset

predicted_phi: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANGLE*}

The phi angle at which the bragg peak of the reflection is predicted

@description: (optional) *NX_CHAR*

Describes the dataset

predicted_px_x: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The x pixel position at which the bragg peak of the reflection is predicted

@description: (optional) *NX_CHAR*

Describes the dataset

predicted_px_y: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The y pixel position at which the bragg peak of the reflection is predicted

@description: (optional) *NX_CHAR*

Describes the dataset

observed_frame: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The estimate of the frame at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_frame_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The variance on the estimate of the frame at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_frame_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The standard deviation of the estimate of the frame at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_px_x: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The estimate of the pixel x position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_px_x_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The variance on the estimate of the pixel x position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_px_x_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The standard deviation of the estimate of the pixel x position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_px_y: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The estimate of the pixel y position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_px_y_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The variance on the estimate of the pixel y position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_px_y_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The standard deviation of the estimate of the pixel y position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_phi: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANGLE*}

The estimate of the phi angle at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_phi_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANGLE*}

The variance on the estimate of the phi angle at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_phi_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANGLE*}

The standard deviation of the estimate of the phi angle at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_x: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The estimate of the x position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_x_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The variance on the estimate of the x position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_x_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The standard deviation of the estimate of the x position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_y: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_y_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The variance on the estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

observed_y_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_LENGTH*}

The standard deviation of the estimate of the y position at which the central impact of the reflection was recorded

@description: (optional) *NX_CHAR*

Describes the dataset

bounding_box: (optional) *NX_INT* (Rank: 2, Dimensions: [n, 6]) {units=*NX_UNITLESS*}

The bounding box around the recorded recorded reflection. Should be an integer array of length 6, where the 6 values are pixel positions or frame numbers, as follows:

index	meaning
0	The lower pixel x position
1	The upper pixel x position
2	The lower pixel y position
3	The upper pixel y position
4	The lower frame number
5	The upper frame number

@description: (optional) *NX_CHAR*

Describes the dataset

background_mean: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The mean background under the reflection peak

@description: (optional) *NX_CHAR*

Describes the dataset

int_prf: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The estimate of the reflection intensity by profile fitting

@description: (optional) *NX_CHAR*

Describes the dataset

int_prf_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The variance on the estimate of the reflection intensity by profile fitting

@description: (optional) *NX_CHAR*

Describes the dataset

int_prf_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The standard deviation of the estimate of the reflection intensity by profile fitting

@description: (optional) *NX_CHAR*

Describes the dataset

int_sum: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The estimate of the reflection intensity by summation

@description: (optional) *NX_CHAR*

Describes the dataset

int_sum_var: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The variance on the estimate of the reflection intensity by summation

@description: (optional) *NX_CHAR*

Describes the dataset

int_sum_errors: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The standard deviation of the estimate of the reflection intensity by summation

@description: (optional) *NX_CHAR*

Describes the dataset

lp: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The LP correction factor to be applied to the reflection intensities

@description: (optional) *NX_CHAR*

Describes the dataset

prf_cc: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n])

The correlation of the reflection profile with the reference profile used in profile fitting

@description: (optional) *NX_CHAR*

Describes the dataset

overlaps: (optional) *NX_INT*

An adjacency list specifying the spatial overlaps of reflections. The adjacency list is specified using an array data type where the elements of the array are the indices of the adjacent overlapped reflection

@description: (optional) *NX_CHAR*

Describes the dataset

polar_angle: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANGLE*}

Polar angle of reflection centroid, following the NeXus simple (spherical polar) coordinate system

@description: (optional) *NX_CHAR*

Describes the dataset

azimuthal_angle: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANGLE*}

Azimuthal angle of reflection centroid, following the NeXus simple (spherical polar) coordinate system

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXreflections/azimuthal_angle-field*
- */NXreflections/background_mean-field*
- */NXreflections/background_mean@description-attribute*
- */NXreflections/bounding_box-field*
- */NXreflections/bounding_box@description-attribute*
- */NXreflections/d-field*
- */NXreflections/d@description-attribute*
- */NXreflections/det_module-field*
- */NXreflections/det_module@description-attribute*
- */NXreflections/entering-field*
- */NXreflections/entering@description-attribute*
- */NXreflections/experiments-field*
- */NXreflections/flags-field*
- */NXreflections/flags@description-attribute*
- */NXreflections/h-field*
- */NXreflections/h@description-attribute*
- */NXreflections/id-field*
- */NXreflections/id@description-attribute*
- */NXreflections/int_prf-field*

- */NXreflections/int_prf@description-attribute*
- */NXreflections/int_prf_errors-field*
- */NXreflections/int_prf_errors@description-attribute*
- */NXreflections/int_prf_var-field*
- */NXreflections/int_prf_var@description-attribute*
- */NXreflections/int_sum-field*
- */NXreflections/int_sum@description-attribute*
- */NXreflections/int_sum_errors-field*
- */NXreflections/int_sum_errors@description-attribute*
- */NXreflections/int_sum_var-field*
- */NXreflections/int_sum_var@description-attribute*
- */NXreflections/k-field*
- */NXreflections/k@description-attribute*
- */NXreflections/l-field*
- */NXreflections/l@description-attribute*
- */NXreflections/lp-field*
- */NXreflections/lp@description-attribute*
- */NXreflections/observed_frame-field*
- */NXreflections/observed_frame@description-attribute*
- */NXreflections/observed_frame_errors-field*
- */NXreflections/observed_frame_errors@description-attribute*
- */NXreflections/observed_frame_var-field*
- */NXreflections/observed_frame_var@description-attribute*
- */NXreflections/observed_phi-field*
- */NXreflections/observed_phi@description-attribute*
- */NXreflections/observed_phi_errors-field*
- */NXreflections/observed_phi_errors@description-attribute*
- */NXreflections/observed_phi_var-field*
- */NXreflections/observed_phi_var@description-attribute*
- */NXreflections/observed_px_x-field*
- */NXreflections/observed_px_x@description-attribute*
- */NXreflections/observed_px_x_errors-field*
- */NXreflections/observed_px_x_errors@description-attribute*
- */NXreflections/observed_px_x_var-field*
- */NXreflections/observed_px_x_var@description-attribute*
- */NXreflections/observed_px_y-field*

- */NXreflections/observed_px_y@description-attribute*
- */NXreflections/observed_px_y_errors-field*
- */NXreflections/observed_px_y_errors@description-attribute*
- */NXreflections/observed_px_y_var-field*
- */NXreflections/observed_px_y_var@description-attribute*
- */NXreflections/observed_x-field*
- */NXreflections/observed_x@description-attribute*
- */NXreflections/observed_x_errors-field*
- */NXreflections/observed_x_errors@description-attribute*
- */NXreflections/observed_x_var-field*
- */NXreflections/observed_x_var@description-attribute*
- */NXreflections/observed_y-field*
- */NXreflections/observed_y@description-attribute*
- */NXreflections/observed_y_errors-field*
- */NXreflections/observed_y_errors@description-attribute*
- */NXreflections/observed_y_var-field*
- */NXreflections/observed_y_var@description-attribute*
- */NXreflections/overlaps-field*
- */NXreflections/overlaps@description-attribute*
- */NXreflections/partiality-field*
- */NXreflections/partiality@description-attribute*
- */NXreflections/polar_angle-field*
- */NXreflections/polar_angle@description-attribute*
- */NXreflections/predicted_frame-field*
- */NXreflections/predicted_frame@description-attribute*
- */NXreflections/predicted_phi-field*
- */NXreflections/predicted_phi@description-attribute*
- */NXreflections/predicted_px_x-field*
- */NXreflections/predicted_px_x@description-attribute*
- */NXreflections/predicted_px_y-field*
- */NXreflections/predicted_px_y@description-attribute*
- */NXreflections/predicted_x-field*
- */NXreflections/predicted_x@description-attribute*
- */NXreflections/predicted_y-field*
- */NXreflections/predicted_y@description-attribute*
- */NXreflections/prf_cc-field*

- */NXreflections/prf_cc@description-attribute*
- */NXreflections/reflection_id-field*
- */NXreflections/reflection_id@description-attribute*
- */NXreflections@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:

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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXresolution.nxdl.xml

NXroi_process

Status:

base class, extends *NXprocess*

Description:

Base class to report on the characterization of an area or volume of material.

This area or volume of material is considered a region-of-interest (ROI).

This base class should be used when the characterization was achieved by processing data from experiment or computer simulations into models of the microstructure of the material and the properties of the material or its crystal defects within this ROI. Microstructural features is a narrow synonym for these crystal defects.

This base class can also be used to store data and metadata of the representation of the ROI, i.e. its discretization and shape.

Methods from computational geometry are typically used for defining a discretization of the area and volume.

Do not confuse this base class with *NXregion*. The purpose of the *NXregion* base class is to document data access i.e. I/O pattern on arrays. Therefore, concepts from *NXregion* operate in data space rather than in real or simulated real space.

Symbols:

No symbol table

Groups cited:
NXprocess

Structure:

PROCESS: (optional) *NXprocess*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXroi_process/PROCESS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXroi_process.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*

DEPRECATED: NAPI is frozen.

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_alphabetagamma: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_ANGLE*}

Crystallography unit cell parameters alpha, beta, and gamma

unit_cell: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_comp, 6]) {units=*NX_LENGTH*}

Unit cell parameters (lengths and angles)

unit_cell_volume: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp]) {units=*NX_VOLUME*}

Volume of the unit cell

sample_orientation: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_ANGLE*}

This will follow the Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464

orientation_matrix: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [n_comp, 3, 3])

Orientation matrix of single crystal sample using Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464

ub_matrix: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [n_comp, 3, 3])

UB matrix of single crystal sample using Busing-Levy convention: W. R. Busing and H. A. Levy (1967). Acta Cryst. 22, 457-464. This is the multiplication of the orientation_matrix, given above, with the B matrix which can be derived from the lattice constants.

mass: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp]) {units=*NX_MASS*}

Mass of sample

density: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp]) {units=*NX_MASS_DENSITY*}

Density of sample

relative_molecular_mass: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp]) {units=*NX_MASS*}

Relative Molecular Mass of sample

type: (optional) *NX_CHAR*

Any of these values:

- `sample`
- `sample+can`
- `can`
- `sample+buffer`
- `buffer`
- `calibration sample`
- `normalisation sample`
- `simulated data`
- `none`
- `sample environment`

situation: (optional) *NX_CHAR*

The atmosphere will be one of the components, which is where its details will be stored; the relevant components will be indicated by the entry in the sample_component member.

Any of these values:

- `air`
- `vacuum`
- `inert atmosphere`
- `oxidising atmosphere`

- `reducing atmosphere`
- `sealed can`
- `other`

description: (optional) *NX_CHAR* <=

Description of the sample

preparation_date: (optional) *NX_DATE_TIME*

Date of preparation of the sample

component: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_comp])

Details of the component of the sample and/or can

sample_component: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_comp])

Type of component

Any of these values: `sample` | `can` | `atmosphere` | `kit`

concentration: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp]) {units=*NX_MASS_DENSITY*}

Concentration of each component

volume_fraction: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp])

Volume fraction of each component

scattering_length_density: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_comp]) {units=*NX_SCATTERING_LENGTH_DENSITY*}

Scattering length density of each component

unit_cell_class: (optional) *NX_CHAR*

In case it is all we know and we want to record/document it

Any of these values:

- `triclinic`
- `monoclinic`
- `orthorhombic`
- `tetragonal`
- `rhombohedral`
- `hexagonal`
- `cubic`

space_group: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_comp])

Crystallographic space group

point_group: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_comp])

Crystallographic point group, deprecated if space_group present

path_length: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Path length through sample/can for simple case when it does not vary with scattering direction

path_length_window: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of a beam entry/exit window on the can (mm) - assumed same for entry and exit

thickness: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

sample thickness

external_DAC: (optional) *NX_FLOAT* {units=*NX_ANY*}

value sent to user's sample setup

short_title: (optional) *NX_CHAR*

20 character fixed length sample description for legends

rotation_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Optional rotation angle for the case when the powder diagram has been obtained through an omega-2theta scan like from a traditional single detector powder diffractometer. Note, it is recommended to use NXtransformations instead.

x_translation: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Translation of the sample along the X-direction of the laboratory coordinate system Note, it is recommended to use NXtransformations instead.

distance: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Translation of the sample along the Z-direction of the laboratory coordinate system. Note, it is recommended to use NXtransformations instead.

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_alphabetagamma: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_ANGLE*}

Crystallography unit cell parameters alpha, beta, and gamma

unit_cell_volume: (optional) *NX_FLOAT* {units=*NX_VOLUME*}

Volume of the unit cell

sample_orientation: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_ANGLE*}

This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967)

orientation_matrix: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [3, 3])

Orientation matrix of single crystal sample component. This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967)

mass: (optional) *NX_FLOAT* {units=*NX_MASS*}

Mass of sample component

density: (optional) *NX_FLOAT* {units=*NX_MASS_DENSITY*}

Density of sample component

relative_molecular_mass: (optional) *NX_FLOAT* {units=*NX_MASS*}

Relative Molecular Mass of sample component

description: (optional) *NX_CHAR* <=

Description of the sample component

volume_fraction: (optional) *NX_FLOAT*

Volume fraction of component

scattering_length_density: (optional) *NX_FLOAT* {units=*NX_SCATTERING_LENGTH_DENSITY*}

Scattering length density of component

unit_cell_class: (optional) *NX_CHAR*

In case it is all we know and we want to record/document it

Any of these values:

- `triclinic`
- `monoclinic`
- `orthorhombic`
- `tetragonal`
- `rhombohedral`
- `hexagonal`
- `cubic`

space_group: (optional) *NX_CHAR*

Crystallographic space group

point_group: (optional) *NX_CHAR*

Crystallographic point group, deprecated if space_group present

transmission: (optional) *NXdata* <=

As a function of Wavelength

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

NXscan_controller

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 scan box 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 NXebeam_column base class.

CIRCUIT: (optional) *NXcircuit*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXscan_controller/CIRCUIT-group*
- */NXscan_controller/DEFLECTOR-group*
- */NXscan_controller/dwell_time-field*
- */NXscan_controller/flyback_time-field*
- */NXscan_controller/scan_schema-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXscan_controller.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:
NXgeometry, *NXlog*, *NXoff_geometry*, *NXorientation*

Structure:

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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsensor/attached_to-field*
- */NXsensor/depends_on-field*
- */NXsensor/external_field_brief-field*
- */NXsensor/external_field_full-group*
- */NXsensor/geometry-group*
- */NXsensor/high_trip_value-field*
- */NXsensor/low_trip_value-field*
- */NXsensor/measurement-field*
- */NXsensor/model-field*
- */NXsensor/name-field*

- */NXsensor/OFF_GEOMETRY-group*
- */NXsensor/run_control-field*
- */NXsensor/short_name-field*
- */NXsensor/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*

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*, *NXelectromagnetic_lens*, *NXfabrication*, *NXgeometry*, *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 a 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.

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

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/ELECTROMAGNETIC_LENS-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/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

NXspindispersion

Status:

base class, extends *NXcomponent*

Description:

Class to describe spin filters in photoemission experiments.

Symbols:

No symbol table

Groups cited:
NXactivity, *NXdeflector*, *NXelectromagnetic_lens*, *NXhistory*

Structure:

type: (optional) *NX_CHAR*

Type of spin detector, VLEED, SPLEED, Mott, etc.

figure_of_merit: (optional) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Figure of merit of the spin detector

shermann_function: (optional) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Effective Shermann function, calibrated spin selectivity factor

scattering_energy: (optional) *NX_FLOAT* {units=*NX_ENERGY*}

Energy of the spin-selective scattering

scattering_angle: (optional) *NX_FLOAT* {units=*NX_ANGLE*}

Angle of the spin-selective scattering

scattering_target: (optional) *NX_CHAR*

Name of the target

scattering_target_history: (optional) *NXhistory*

A set of activities that occurred to the `scattering_target` prior to/during the. experiment. For example, this group can be used to describe the preparation of the `scattering_target`.
preparation: (optional) *NXactivity* <=

Preparation procedure of the spin target

DEFLECTOR: (optional) *NXdeflector*

Deflectors in the spin dispersive section

ELECTROMAGNETIC_LENS: (optional) *NXelectromagnetic_lens*

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/ELECTROMAGNETIC_LENS-group*
- */NXspindispersion/figure_of_merit-field*
- */NXspindispersion/scattering_angle-field*
- */NXspindispersion/scattering_energy-field*
- */NXspindispersion/scattering_target-field*
- */NXspindispersion/scattering_target_history-group*
- */NXspindispersion/scattering_target_history/preparation-group*
- */NXspindispersion/shermann_function-field*
- */NXspindispersion/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXspindispersion.nxdl.xml

NXsubentry

Status:

base class, extends *NXobject*

Description:

Group of multiple application definitions for "multi-modal" (e.g. SAXS/WAXS) measurements.

`NXsubentry` is a base class virtually identical to *NXentry* and is used as the (overlay) location for application definitions. Use a separate `NXsubentry` for each application definition.

To use `NXsubentry` with a hypothetical application definition called `NXmyappdef`:

- Create a group with attribute `NX_class="NXsubentry"`
- Within that group, create a field called `definition="NXmyappdef"`.
- There are two optional attributes of definition: `version` and `URL`

The intended use is to define application definitions for a multi-modal (a.k.a. multi-technique) *NXentry*. Previously, an application definition replaced *NXentry* with its own definition. With the increasing popularity of instruments combining multiple techniques for data collection (such as SAXS/WAXS instruments), it was recognized the application definitions must be entered in the NeXus data file tree as children of *NXentry*.

Symbols:

No symbol table

Groups cited:
NXcollection, *NXdata*, *NXinstrument*, *NXmonitor*, *NXnote*, *NXparameters*, *NXprocess*, *NXsample*, *NXuser*

Structure:

@default: (optional) *NX_CHAR* <=

Declares which *NXdata* group contains the data to be shown by default. It is used to resolve ambiguity when one *NXdata* group exists. The value *names* the default *NXentry* group. The value must be the name of a child of the current group. The child must be a NeXus group or a link to a NeXus group.

For more information about how NeXus identifies the default plottable data, see the *Find Plottable Data, v3* section.

@IDF_Version: (optional) *NX_CHAR*

ISIS Muon IDF_Version

title: (optional) *NX_CHAR*

Extended title for entry

experiment_identifier: (optional) *NX_CHAR*

Unique identifier for the experiment, defined by the facility, possibly linked to the proposals

experiment_description: (optional) *NX_CHAR*

Brief summary of the experiment, including key objectives.

collection_identifier: (optional) *NX_CHAR*

User or Data Acquisition defined group of NeXus files or *NXentry*

collection_description: (optional) *NX_CHAR*

Brief summary of the collection, including grouping criteria.

entry_identifier: (optional) *NX_CHAR*

unique identifier for the measurement, defined by the facility.

definition: (optional) *NX_CHAR*

Official NeXus NXDL schema to which this subentry conforms

@version: (optional) *NX_CHAR*

NXDL version number

@URL: (optional) *NX_CHAR*

URL of NXDL file

definition_local: (optional) *NX_CHAR*

Local NXDL schema extended from the subentry specified in the `definition` field. This contains any locally-defined, additional fields in the subentry.

@version: (optional) *NX_CHAR*

NXDL version number

@URL: (optional) *NX_CHAR*

URL of NXDL file

start_time: (optional) *NX_DATE_TIME*

Starting time of measurement

end_time: (optional) *NX_DATE_TIME*

Ending time of measurement

duration: (optional) *NX_INT* {units=*NX_TIME*}

Duration of measurement

collection_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

Time transpired actually collecting data i.e. taking out time when collection was suspended due to e.g. temperature out of range

run_cycle: (optional) *NX_CHAR*

Such as "2007-3". Some user facilities organize their beam time into run cycles.

program_name: (optional) *NX_CHAR*

Name of program used to generate this file

@version: (optional) *NX_CHAR*

Program version number

@configuration: (optional) *NX_CHAR*

configuration of the program

revision: (optional) *NX_CHAR*

Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument definition format, …

@comment: (optional) *NX_CHAR*

pre_sample_flightpath: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

This is the flightpath before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

experiment_documentation: (optional) *NXnote* <=

Description of the full experiment (document in pdf, latex, ...)

notes: (optional) *NXnote* <=

Notes describing entry

thumbnail: (optional) *NXnote* <=

A small image that is representative of the entry. An example of this is a 640x480 jpeg image automatically produced by a low resolution plot of the NXdata.

@mime_type: (optional) *NX_CHAR*

The value should be an `image/*`

Obligatory value: `image/*`

USER: (optional) *NXuser*

SAMPLE: (optional) *NXsample*

INSTRUMENT: (optional) *NXinstrument*

COLLECTION: (optional) *NXcollection* <=

MONITOR: (optional) *NXmonitor*

DATA: (optional) *NXdata* <=

PARAMETERS: (optional) *NXparameters* <=

PROCESS: (optional) *NXprocess*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsubentry/COLLECTION-group*
- */NXsubentry/collection_description-field*
- */NXsubentry/collection_identifier-field*
- */NXsubentry/collection_time-field*
- */NXsubentry/DATA-group*
- */NXsubentry/definition-field*
- */NXsubentry/definition@URL-attribute*
- */NXsubentry/definition@version-attribute*
- */NXsubentry/definition_local-field*
- */NXsubentry/definition_local@URL-attribute*
- */NXsubentry/definition_local@version-attribute*

- */NXsubentry/duration-field*
- */NXsubentry/end_time-field*
- */NXsubentry/entry_identifier-field*
- */NXsubentry/experiment_description-field*
- */NXsubentry/experiment_documentation-group*
- */NXsubentry/experiment_identifier-field*
- */NXsubentry/INSTRUMENT-group*
- */NXsubentry/MONITOR-group*
- */NXsubentry/notes-group*
- */NXsubentry/PARAMETERS-group*
- */NXsubentry/pre_sample_flightpath-field*
- */NXsubentry/PROCESS-group*
- */NXsubentry/program_name-field*
- */NXsubentry/program_name@configuration-attribute*
- */NXsubentry/program_name@version-attribute*
- */NXsubentry/revision-field*
- */NXsubentry/revision@comment-attribute*
- */NXsubentry/run_cycle-field*
- */NXsubentry/SAMPLE-group*
- */NXsubentry/start_time-field*
- */NXsubentry/thumbnail-group*
- */NXsubentry/thumbnail@mime_type-attribute*
- */NXsubentry/title-field*
- */NXsubentry/USER-group*
- */NXsubentry@default-attribute*
- */NXsubentry@IDF_Version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/base_classes/NXsubentry.nxdl.xml

NXtransformations

Status:

base class, extends *NXobject*

Description:

Collection of axis-based translations and rotations to describe a geometry. May also contain axes that do not move and therefore do not have a transformation type specified, but are useful in understanding coordinate frames within which transformations are done, or in documenting important directions, such as the direction of gravity.

A nested sequence of transformations lists the translation and rotation steps needed to describe the position and orientation of any movable or fixed device.

There will be one or more transformations (axes) defined by one or more fields for each transformation. Transformations can also be described by NXlog groups when the values change with time. The all-caps name `AXISNAME` designates the particular axis generating a transformation (e.g. a rotation axis or a translation axis or a general axis). The attribute `units="NX_TRANSFORMATION"` designates the units will be appropriate to the `transformation_type` attribute:

- `NX_LENGTH` for `translation`
- `NX_ANGLE` for `rotation`
- `NX_UNITLESS` for axes for which no transformation type is specified

This class will usually contain all axes of a sample stage or goniometer or a detector. The NeXus default McSTAS coordinate frame is assumed, but additional useful coordinate axes may be defined by using axes for which no transformation type has been specified.

The entry point (`depends_on`) will be outside of this class and point to a field in here (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 NXtransformations 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

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*

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/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

NXwaveplate

Status:

base class, 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:
NXdata, *NXsample*

Structure:

type: (optional) *NX_CHAR*

Type of waveplate (e.g. achromatic or zero-order).

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `zero-order`
- `achromatic`
- `multiple-order`
- `dual-wavelength`

retardance: (optional) *NX_CHAR*

Specify the retardance of the waveplate (e.g. full-wave, half-wave (lambda/2), quarter-wave (lambda/4)).

Any of these values: `full-wave` | `half-wave` | `quarter-wave`

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). In this case, also use type="achromatic".

diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Diameter of the waveplate (if the waveplate is circular).

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.

retardance_distribution: (optional) *NXdata* <=

Wavelength resolved retardance of the waveplate.

substrate: (optional) *NXsample*

Describe the material of the substrate of the waveplate 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 waveplate. 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 waveplate substrate.

index_of_refraction_substrate: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [2, N_spectrum]) {units=*NX_UNITLESS*}

Complex index of refraction of the waveplate substrate. Specify at given wavelength (or energy, wavenumber etc.) values.

coating: (optional) *NXsample*

Is the waveplate 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/reflectance-field*
- */NXwaveplate/retardance-field*
- */NXwaveplate/retardance_distribution-group*
- */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/base_classes/NXwaveplate.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.

Application definitions are grouped together based on the research fields where these are typically used. Definitions that address multiple research fields are listed in each category:

Atom Probe Microscopy

Diffraction & Scattering Techniques

Electron Microscopy

Imaging Techniques

Multi-Dimensional Photoemission Spectroscopy

Optical Spectroscopy

Time-of-Flight Techniques

Complete List

Atom Probe Microscopy / Tomography

Introduction

Set of data schemas to describe the acquisition, i.e. measurement side, the extraction of hits from detector raw data, steps to compute mass-to-charge-state ratios from uncorrected time of flight data, the reconstruction, and the ranging i.e., the identification of peaks in the mass-to-charge-state ratio histogram to detect (molecular) ions. The data schemas are also useful for reporting field-ion microscopy experiments.

Application Definition

Measurements as well as computer simulations of atom probe tomography and field-ion microscopy research are standardized with one application definition:

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 (molecular) iontypes in a process called ranging.

Diffraction & Scattering Techniques

Introduction

Application definitions for different diffraction and (small-angle) scattering techniques

Application Definitions

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.

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*.

NXcanSAS
Implementation of the canSAS standard to store reduced small-angle scattering data of any dimension.

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.

Electron microscopy

Introduction

A set of data schemas is available to describe components of an electron microscope (EM) and its focused-ion beam (FIB) capabilities if available.

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 performing 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., in an interdisciplinary manner.

The application definitions and associated base classes were designed from the perspective of how electron microscopes are used in the materials-science-branch of electron microscopy. Therefore, the focus is on the usage of electron microscopy in condensed-matter physics, chemical physics of solids, and materials engineering applications.

The biology-/bio-materials-/omics-/life-science-community of the electron microscopy community can also take advantage of the definitions and classes. We acknowledge that the Open Microscopy Environment (https://www.openmicroscopy.org) offers a data model and schema set for the life science. Conceptual and semantic overlap between concepts of these two data models exist but should be explored further in the future to improve on the interoperability of data exchange between the materials-science and life science communities.

Design

NXem provides support for documenting research with scanning electron microscopes (SEM), scanning electron microscopes that are equipped with focused-ion beam (FIB) capabilities (SEM/FIB), and transmission electron microscopes (TEM) respectively scanning transmission electron microscopy (STEM). The actual design of the electron-optical beam path and individual components used differ. Therefore, the base classes of and application definition NXem offers modular building blocks from which a virtual electron microscope is instantiated.

The following three figures provide one schematic example for each respective type of instrument (SEM, SEM/FIB, (S)TEM). Specifically, the figure shows how the concepts of NXem can be used to describe each instrument. All figures are meant for illustrative and educational purposes. The figures build up complexity successively starting with the simpler case of an SEM (Fig. 1), moving to an SEM with FIB capabilities (Fig. 2), and arriving at an (S)TEM (Fig. 3) instrument. The capabilities that are illustrated for the SEM in Fig. 1 are also offered for all further cases. For the sake of conciseness and clarity these classes are not repeated though in subsequent figures. The examples take the perspective of typical users who are interested in reporting metadata and data that have been collected during a session with such microscopes:

Fig. 1 - an example for an SEM The instrument is constructed from a so-called column, a housing for all technical components such as the electron source, the lenses, like here shown a condenser and an objective lens, respective apertures, and further components, like a stigmator to correct for axial image distortions. The trajectory of the electron beam along the optical axis is simplified for illustrative purposes. In summary, the sample is illuminated by an electron beam that is guided along the optical axis through and past a set of components. A scan controller is used to deflect this beam to illuminate specific locations on the sample surface. In response to the electron-beam sample interaction an interaction volume is formed. Different types of signals are generated that are picked up by different types of detectors. Three detectors are shown as an example. Apart from the column, an SEM has further components: The base classes that are used for modeling these are listed in the lower part of the figure. These document pumps and other hardware, assumptions made such as frames of references and transformations between these frames, and the computing hardware and software tools that are used for controlling the SEM and all its connected components. Using an electron microscope demands processing of data. These processing steps are modeled with instances and specializations of the NXprocess base class. These specializations are used for documenting the parameterization, the results, and the sequence of such processing steps. Examples of method-specific base classes are NXem_ebsd for electron backscat-

ter diffraction (EBSD), NXem_eds for energy-dispersive X-ray spectroscopy (EDS/EDXS), NXem_img for different imaging modes (secondary electron, backscattered electron), and NXem_eels for electron energy loss spectroscopy (EELS).

Fig. 2 - an example for an SEM with plasma FIB capabilities Adding or flanging another column to an electron microscope equips it with focused-ion beam capabilities. The design of this NXibeam_column follows the design of the NXebeam_column: A housing with technical components, such as the ion source, lenses, apertures, beam distortion and beam shaping components, and an own scan controller for guiding the ion beam towards the sample surface. Like in Fig. 1, the trajectory of the ion beam is simplified.

Fig. 3 an example for a (S)TEM The design principles for the SEM as well as the FIB are used for modeling a transmission electron microscope. Noteworthy is that the figure illustrates an optical setup that is a mixture of a conventional TEM and a STEM (again for illustrative purposes). The presence of a scan controller is one characteristic feature of an STEM. Given that a TEM specimen is typically orders of magnitude thinner than a specimen used in an SEM, the electron beam can penetrate the material. This enables investing additional imaging modes and probing other characteristic electron-matter interactions such as electron energy loss spectroscopy. Consequently, additional lenses and components are introduced into the beam path of the exiting electrons.

Application Definition

An experiment with an electron microscope proceeds as follows: users place a sample into the microscope, calibrate the instrument, take measurements, may prepare their specimens with a focused ion beam, calibrate again, and take further measurements, they process data, until eventually their session on the instrument ends. In between virtually all of these steps, data are collected and stream in from different detectors. Each detector probes different physical mechanisms of the interaction between electrons or other types of radiation with the specimen and its environment. 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. Base classes are provided to describe these steps and events.

A simulation of an electron microscope or of the interaction between an electron beam with matter takes a simpler perspective on many of these practical aspects. Typically, an electron-optical setup and material is defined, assumptions about the properties and trajectory of the electron beam are made or simulated. The simulation analyzes the interaction volume by inspecting e.g. the trajectories of individual electrons or by modeling their collective behavior via computing numerical solutions or approximations for the electromagnetic field.

Measurements as well as computer simulations of electron microscopy research are standardized with one application definition:

NXem
: A general application definition which explores the possibilities of electron microscopes for characterizing electron- and ion-beam matter interactions.

Currently NXem does not provide standardized descriptions for experiments where photons are interacting with the electron beam. The blueprint of NXebeam_column and NXibeam_column surplus the definitions and classes provided by *NXoptical_spectroscopy* provide though a starting point for adding such descriptions in the future via for example an NXpbeam_column.

Imaging Techniques

Introduction

Application definitions for imaging techniques

Application Definitions

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.

Photoemission & core-level spectroscopy

Introduction

These are a set of application definitions to describe multidimensional photoemission (MPES) 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 application definition with minimalistic metadata requirements, apt to describe all photoemission experiments.

NXmpes_arpes
: An application definition for angle-resolved photoemission spectroscopy (ARPES) experiments.

NXxps
: An application definition for X-ray/UV photoelectron spectroscopy (XPS/UPS) experiments.

NXarpes
: An application definition for angular resolved photo emission spectroscopy. Note that this application definition is only kept for legacy reasons and new NeXus ARPES files should use *NXmpes_arpes*.

Base classes

A specific set of base classes which are used in these application definitions can be found *here*.

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, a minimum set of description elements allowing for a reproducible recording of ellipsometry measurements is provided.

Raman spectroscopy

Raman spectroscopy is a characterization method to analyze vibrational properties for liquids, gases, or solids. The measurements are based on the inelastic light scattering due to the material's vibrations. Interpretation can be done based on peaks, which represent the phonon properties (intensity, center, width).

The application definition contains a minimum of descriptive elements required to understand Raman spectroscopy measurements.

Application Definitions

NXoptical_spectroscopy
: A generic application definition for spectroscopy measurements. This includes in particular ellipsometry and Raman spectroscopy measurements, but also other techniques such as photoluminescence, transmission, and reflection measurements. The requirements are: (i) an incident photon beam, (ii) a detector to measure scattered/emitted photons, and (iii) a sample.

NXellipsometry
: An application definition for ellipsometry measurements, including complex systems up to variable angle spectroscopic ellipsometry.

NXraman
: An application definition for Raman spectroscopy measurements.

Base classes

A specific set of base classes which are used in these application definitions can be found *here*.

Time-of-Flight (TOF) Techniques

Introduction

Classical application definitions for time-of-flight (spectroscopy) techniques.

Application Definitions

NXdirecttof
: This is a application definition for raw data from a direct geometry TOF spectrometer.

NXindirecttof
: This is a application definition for raw data from an indirect geometry TOF spectrometer.

NXlauetof
: This is the application definition for a TOF laue diffractometer.

NXreftof
: This is an application definition for raw data from a TOF reflectometer.

NXsastof
: Raw, 2-D SAS data with an area detector with a time-of-flight source.

NXtofnpd
: This is a application definition for raw data from a TOF neutron powder diffractometer.

NXtofraw
: This is an application definition for raw data from a generic TOF instrument.

NXtofsingle
: This is a application definition for raw data from a generic TOF instrument.

Application Definitions

This is the complete list of application definitions:

NXapm
Application definition for real or simulated atom probe and field-ion microscopy experiments.

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.

NXazint1d
Application definition for data from two-dimensional area detectors that has been integrated azimuthally,

NXazint2d
Application definition for data from two-dimensional area detectors that has been integrated azimuthally,

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

NXellipsometry
This is the application definition describing ellipsometry experiments.

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

NXmpes
This is the most general application definition for

NXmpes_arpes
This is a general application definition for angle-resolved (multidimensional)

NXmx
functional application definition for macromolecular crystallography

NXoptical_spectroscopy
A general application definition of optical spectroscopy elements, which may

NXraman
An application definition for Raman spectroscopy experiments.

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.

NXstress
Application definition for stress and strain analysis of crystalline material defined by the EASI-STRESS consortium.

NXstxm
Application definition for a STXM instrument.

NXtas
This is an application definition for a triple axis spectrometer.

NXtofnpd
This is a application definition for raw data from a TOF neutron powder diffractometer

NXtofraw
This is an application definition for raw data from a generic TOF instrument

NXtofsingle
This is a application definition for raw data from a generic TOF instrument

NXtomo
This is the application definition for x-ray or neutron tomography raw data.

NXtomophase
This is the application definition for x-ray or neutron tomography raw data with phase contrast variation at each point.

NXtomoproc
This is an application definition for the final result of a tomography experiment: a 3D construction of some volume of physical properties.

NXxas
This is an application definition for raw data from an X-ray absorption spectroscopy experiment.

NXxasproc
Processed data from XAS. This is energy versus I(incoming)/I(absorbed).

NXxbase
This definition covers the common parts of all monochromatic single crystal raw data application definitions.

NXxeuler
raw data from a four-circle diffractometer with an eulerian cradle, extends *NXxbase*

NXxkappa
raw data from a kappa geometry (CAD4) single crystal diffractometer, extends *NXxbase*

NXxlaue
raw data from a single crystal laue camera, extends *NXxrot*

NXxlaueplate
raw data from a single crystal Laue camera, extends *NXxlaue*

NXxnb
raw data from a single crystal diffractometer, extends *NXxbase*

NXxps
This is the application definition for X-ray photoelectron spectroscopy.

NXxrot
raw data from a rotation camera, extends *NXxbase*

NXapm

Status:

application definition, extends *NXobject*

Description:

Application definition for real or simulated atom probe and field-ion microscopy experiments.

Atom probe tomography and field-ion microscopy are methods for characterizing materials through induced controlled extraction of individual atoms as ions and molecular ions from a sharp needle-shaped specimen.

Offering isotopic and nanometer-scale resolution, atom probe data enable quantification of local chemical composition and computing of volumetric reconstructions which are models for the atomic architecture of the small specimen volume analyzed. These reconstructions provide input for characterization of atomic segregation at crystal defects. The term microstructural features is considered as a narrow synonym for crystal defects.

The aim of the NXapm application definition is to provide a general yet specific enough solution to serialize artifacts for virtually all atom probe and field-ion microscopy experiments.

Before summarizing the design of the base classes and the parts of the NXapm application definition, it is worthwhile to recall and distinguish concepts that are related to atom extraction events and the molecular ions that are frequently generated by the sequence of events:

- An atom probe instrument uses laser or voltage pulsing events to trigger ion extraction events.
- These ions are accelerated in an electric field towards a position-sensitive detector system. Physical events and corresponding signal on this detector is triggered by an ion hitting the detector. Some of these events are not necessarily caused by or directly correlated with an identifiable pulsing event.
- Note that only a part the specimen volume can be measured and finite detection efficiency means that not all atoms in the measured volume will be detected. Neutral atoms can escape detection. Some ions escape detection because they hit into walls of the analysis_chamber.

  Raw data are typically processed as follows:

- Detector pulses and their timing are processed and discriminated into signal events of different quality levels. High quality events might be considered in further processing to identify the corresponding molecular ion and its original position in the reconstructed volume.
- Signal calibration and filtering steps are applied to convert raw time-of-flight data to calibrated mass-to-charge state ratio values and obtain calibrated impact positions on the detector.

- Ranging and identifying an ion that corresponds to each detector event. Isotopic abundance and theoretical models inform these ranging algorithms.
- Finally, such selected ion impact positions and iontypes are used to compute a reconstructed volume of the specimen using backprojection algorithms. In effect, an atom probe measurement is a combination of a data acquisition and a data analysis workflow.

Not only in AMETEK/Cameca's APSuite/IVAS software, which the majority of atom probers use, these concepts are well distinguished. However, the algorithms used to transform correlations between pulses and physical events into actual events, the so-called detector hits of ions, is a proprietary one. This algorithm is also referred to as the hit finding algorithm.

Due to this practical inaccessibility of details, virtually all atom probe studies currently use a reporting schema where the course of the specimen evaporation is documented such that quantities are a function of evaporation_id i.e. actual event/ion, i.e. after having the hit finding algorithm and correlations applied. That is the evaporation_id values take the role of an implicit time and course of the experiment given that ion extraction physically is a sequential process.

This application definition includes fields that the atom probe community has decided to represent best practices for reporting atom probe measurements. Exemplar mapping tables are provided for documents that reported these best practices to translate technical term into concepts of the NXapm application definition.

The NeXus application definition NXapm defines a hierarchical data model with ten building blocks:

The data model represents a tree of concepts. The tree is constructed from groups of concepts representing the branches, together with fields and attributes representing leaves. NXapm is defined by composing and specializing base classes into the following ten categories:

- The field `definition` specifies that the data schema is NXapm. In combination with administrative metadata such as the attribute `NeXus_version` provided by *NXroot* this specifies which version of NXapm the instance data in a NeXus/HDF5 file are compliant with.
- The fields `run_number`, `experiment_alias`, `experiment_description` and the group `userID` provide concepts for storing organizational metadata that contextualize the work within the research workflow and humans involved in this.
- The fields `start_time`, `end_time` provide concepts for framing a temporal context for the research.
- The groups `citeID`, `noteID` provide concepts for adding contextual details such as citations or notes that are associated with the data, i.e. other artifacts that are deemed relevant when reporting about a measurement or simulation. These groups are useful when NXapm is used as a serialization format for technology-partner-agnostic archival of data and metadata that have been collected during a session with an atom probe instrument. The terms run and session are understood as exact synonyms that refer to an uninterrupted period of measurement. Resuming measurement on a specimen after an interruption is viewed as a new run and the new data should not be appended to the previous run, but written to either a new NXentry, or a new file. Removing the specimen from the instrument is an interruption. Changing evaporation conditions while the specimen is remains in the analysis_chamber and resuming thereafter the measurement is not considered as an interruption. It is a common strategy to probe the evaporation process for different instrument parameters. Each individual collection should then though be stored in an own NXapm_event_data group. Parking the specimen to the buffer_chamber and resuming the measurement at a later stage is an interruption. During a run, the microscope is used for a certain amount of time to characterize a single specimen.
- The groups `sample` and `specimen` provide concepts for storing metadata about the sample and the specimen, i.e. the smaller part that was removed from the sample to be measured in the atom probe session. The term "tip" in the context of atom probe research is considered jargon. Specimen is an exact synonym for tip.

- The field `operation_mode` and group `measurement` provides concepts that are useful for describing a measurement during a session with an atom probe or field-ion microscope. This includes the chain of events of data and metadata that were collected during such a session.
- The group `simulation` provides concepts that are useful for describing a simulation of an atom extraction, ionization, and ion trajectory simulation. Combined with `measurement` this provides a data schema for defining a digital twin of the instrument and its setup.
- The groups `consistent_rotations`, and `NAMED_reference_frame` provide concepts for reporting coordinate systems (frames of reference) and rotation conventions that clarify how data should be interpreted specifying the rotation of orientable objects in the microscope, its components, or of crystals and crystal defects in the material analyzed.
- The group `atom_probeID` provides concepts for the computational workflows that were used to convert raw detector data into reconstructed ion positions and documentation of ranging definitions made.
- The group `profiling` provides concepts for reporting computational details such as programs and libraries used, for documenting the libraries of virtual environments such as those used by conda or python virtual environment, including details about the computing hardware used, and documenting capabilities for performance analyses and benchmarking of the software or its parts.

Design choices:

Given that most atom probe instruments across the globe were built by AMETEK/Cameca and are delivered with the AP Suite/IVAS software there is some homogeneity in how a measurement is performed and which data artifacts and algorithms used for data processing. Complementary use of open-source software specifically for the reconstruction, ranging, and later data processing stages contributes to a landscape of multiple tools in use. Therefore, communication of atom probe research differs between user groups. This holds even more so true for the sub community in atom probe which study physical mechanisms involved during ionization to the point that here that almost each research work defines different simulation tools with different types of data artifacts.

NXapm defines constraints on the existence and cardinality of concepts and its concept branches but seeks to offer a compromise. The key design pattern followed is that most branches are made optional or at most recommended but their child concepts are conditionally required. Thereby, NXapm can cover a variety of simple but also complex use cases. An example of this parent-optional-but-children-stronger-restricted design is the combination of the optional group `measurement` with its required child `measurement/instrument`: Users which report simulations are not forced to document the instrument but users which have characterized a specimen are motivated to report about the instrument. They are though not necessarily required to report all the details of the instruments' components because the design pattern is applied recursively.

NXapm distinguishes and stores instance data based on how long they remain unchanged:

`measurement` provides two groups `measurement/instrument` and `measurement/eventID`. The first group is designed for storing metadata about the instrument that do not change over the course of the session. Examples are the name of the technology partner who built the microscope or whether a laser or voltage pulser and reflectron exists or not. The second group is designed for metadata and data that are collected during the session with the instrument. These, are stored as instances of `measurement/eventID`, events that can be time-stamped individually. Each instance of a group `measurement/eventID` contains `measurement/instrument` whose purpose is to store those specific state and settings of the instrument that was present during the collection of the event. Thereby, changing conditions such as campaigns with different target detection rate can be stored.

Noteworthy, such an approach of the atom probe detecting groups of events and storing these as groups has also been in use in the proprietary software via CamecaRoot, a set of customized data structures and file formats that use the CernRoot library. By virtue of design this reduces unnecessary repetition of metadata stored in the first group.

`atom_probeID` offer classes for the each task relevant task in the data processing pipeline that converts raw detector event data to calibrated mass-to-charge-state-ratio values and hit_position on the detector. These include `initial_specimen`, and `final_specimen` locations for storing images of the specimen prior/after the measurement as considered best practice by AMETEK/Cameca, `raw_data` for delay-line timing data, `hit_finding` for details of the hit finding algorithm, `hit_spatial_filtering` a process that filters hits of too low quality and those laying outside the about to be computed reconstruction volume. Furthermore, group `voltage_and_bowl` offers a place for documenting calibrations and processing nonlinearities. Group `mass_to_charge_conversion` is used to document the mass calibration and the conversion from time-of-flight to mass-to-charge-state-ratio values.

Finally, the groups `reconstruction` and `ranging` were designed to match and document the classical approaches how from all the previous sources of input one can compute a reconstructed volume, and apply peak fitting routines on the mass-to-charge-state-ratio histogram to label ions, i.e. range them for their isotopic identity. Group `atom_probeID/reconstruction/naive_discretization` offers a standardized way to report simple three-dimensional histograms. Group `atom_probeID/ranging/peak_identification/ionID` and its complementing group `atom_probeID/ranging/peak_identification/ionID/charge_state_analysis` solves the issue that the ranging definitions in classical file formats are not reported for always for their isotopic identity and charge state. The field `atom_probeID/ranging/peak_identification/iontypes` provides a place for storing a compact representation of the results of each ranging definition made at the level of each ion.

*The compatibility of NXapm and NXem:**

The design of NXapm mirrors that of *NXem*. This was an intentional choice to support the increasingly stronger connection between these two materials characterization methods, especially in light of recent advances in the direct coupling of atom probe and transmission electron microscopes and scanning transmission electron microscopes.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ht: Number of hit qualities, the so-called hit types, distinguished.

n_dld: Number of delay-line-detector (DLD) wires of the detector.

n_bins: Number of bins used in the mass-to-charge-state-ratio spectrum.

p: Number of pulses collected in between start_time and end_time resolved by an instance of *NXapm_event_data*. If this is not defined, p is the number of ions included in the reconstructed volume if the application definition is used to store results of an already reconstructed dataset.

p_out: Number of pulses returned by the hit_finding algorithm. Neither necessarily equal to p nor to n.

n: Number of ions spatially filtered from results of the hit_finding algorithm from which an instance of a reconstructed volume has been generated. These ions get new identifier assigned in the process, the so-called evaporation_id. This identifier must not be confused with the pulse_id. This value is typically smaller than both p and p_out.

m_r: Number of mass resolution values.

Groups cited:

NXapm_charge_state_analysis, *NXapm_event_data*, *NXapm_instrument*, *NXapm_measurement*, *NXapm_ranging*, *NXapm_reconstruction*, *NXapm_simulation*, *NXatom*, *NXchemical_composition*, *NXcite*, *NXcollection*, *NXcomponent*, *NXcoordinate_system*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXdata*, *NXdetector*, *NXelectromagnetic_lens*, *NXentry*, *NXfabrication*, *NXimage*, *NXmanipulator*, *NXnote*, *NXparameters*, *NXpeak*, *NXprocess*, *NXprogram*, *NXpump*, *NXroi_process*, *NXsample*, *NXsensor*, *NXsource*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm`

@version: (optional) *NX_CHAR* <=

run_number: (recommended) *NX_UINT* {units=*NX_UNITLESS*}

The identifier whereby the experiment is referred to in the control software.

It is common practice in atom probe research to refer to a measurement on a single specimen as a run. When working with AMETEK/Cameca instruments it is a common practice also to store all data associated with such a run in files whose name is composed from a prefix that details the type of instrument (e.g. R5076) followed by the run_number. These filenames are often used as the specimen_name or experiment_identifier. The terms run and session are understood as exact synonyms.

For other instruments, such as the one from Stuttgart or Oxcart from Erlangen, or the instruments at GPM in Rouen, use the identifier which matches best conceptually 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; provided these RHIT or HITS files respectively are stored alongside a data artifact. With NXapm the RHIT or HITS can be stored via NXnote in the hit_finding algorithm section.

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. Such interrupted runs should be stored as individual *NXentry* instances in one NeXus file.

experiment_alias: (optional) *NX_CHAR*

Alias or 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 atom probe session started. If the exact duration of the measurement is not relevant, start_time only should be used.

The start_time is required in order to ensure that at least one point in time is provided for full temporal context to a measurement and simulation when writing instance data using NXapm. Otherwise, the instance data can not be sorted in order or even placed in a logical sequence to other steps of the research workflow, which would disallow using functionalities in research data management systems that rely on temporal context.

Specifying start_time and end_time is useful for capturing more detailed bookkeeping 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. For this purpose the *NXapm_event_data* instance should be used.

end_time: (recommended) *NX_DATE_TIME* <=

ISO 8601 time code with local time zone offset to UTC included when the atom probe session ended.

Writing the end_time can be a tricky in practice. If written at the start of the experiment, it can only be an estimate. If written at the end, there is the risk for having the computer crash or lose power. The absence of end_time should not be interpreted as that the experiment was aborted. Only, the field `status` should be used for communicating such abortion.

elapsed_time: (recommended) *NX_FLOAT* {units=*NX_TIME*}

How long did the measurement take e.g. use CRunHeader.CAnalysis.fElapsedTime

operation_mode: (required) *NX_CHAR*

What type of atom probe experiment is performed to inform research data management systems and allow filtering:

- apt are experiments where the analysis_chamber has no imaging gas. Experiments with LEAP instruments are typically with this operation_mode.
- 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. Few experiments of this type have been performed, as it can be detrimental to LEAP systems (see S. Katnagallu et al.).

Any of these values or a custom value (if you use a custom value, also set @custom=True): `apt | fim | apt_fim`

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 which are considered relevant to understand with which software tools this NeXus file instance was generated. Ideally, to 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.

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* <=

citeID: (optional) *NXcite*

author: (optional) *NX_CHAR*

The author(s) of that reference.

doi: (required) *NX_CHAR* <=

noteID: (optional) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

userID: (recommended) *NXuser* <=

identifierNAME: (recommended) *NX_CHAR* <=

@type: (required) *NX_CHAR* <=

name: (optional) *NX_CHAR* <=

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.

In NXapm, a measurement is performed on a specimen. Since APM specimens are very small, they are typically cut from a larger object with some scientific significance, which NXapm refers to as a sample.

identifierNAME: (recommended) *NX_CHAR* <=

is_simulation: (required) *NX_BOOLEAN*

False, if the sample is a real one. True, if the sample is a virtual one.

alias: (required) *NX_CHAR*

Given name/alias for the sample.

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.

If the specimen does not contain many crystals average values might be an unreliable descriptor.

Reporting a grain size may be useful though as it allows judging if specific features are expected to be found in the detector hit map.

grain_diameter_errors: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Magnitude of the standard deviation of the grain_diameter.

heat_treatment_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

An array of elapsed time, the independent axis, of a time-temperature curve.

This field can be used in combination with heat_treatment_temperature and heat_treatment_temperature_errors as well as heat_treatment_quenching_rate and heat_treatment_quenching_rate_errors respectively. In this case, these fields should also be stored as an array with the same dimensions as heat_treatment_time to store the dependant axes of a time-temperature curve as well as its first derivative.

heat_treatment_temperature: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*}

If heat_treatment_time is absent, 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 a documentation of the annealing treatment as a function of time is available one should better rely on this information and have it stored alongside the NeXus file.

If heat_treatment_time is provided, the temperature. Consult the docstring of heat_treatment_time.

heat_treatment_temperature_errors: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*}

Magnitude of the standard deviation of the heat_treatment_temperature.

If heat_treatment_time is provided, the magnitude of the standard derivation of the temperature. Consult the docstring of heat_treatment_time.

heat_treatment_quenching_rate: (optional) *NX_FLOAT* {units=*NX_ANY*}

If heat_treatment_time is absent, the rate of the last quenching step.

Knowledge about this value can give an idea how the sample was heat treated. However, there are many situations where one can imagine that the scalar value for just the quenching rate is insufficient.

If heat_treatment_time is provided, the first derivative of the time-temperature curve. Consult the docstring of heat_treatment_time for further details.

heat_treatment_quenching_rate_errors: (optional) *NX_FLOAT* {units=*NX_ANY*}

Magnitude of the standard deviation of the heat_treatment_quenching_rate.

If heat_treatment_time is provided, the magnitude of the standard deviation of the first derivative of the time-temperature curve. Consult the docstring of heat_treatment_time for further details.

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* <=

ELEMENT: (required) *NXatom* <=

chemical_symbol: (required) *NX_CHAR*

composition: (required) *NX_FLOAT* <=

composition_errors: (recommended) *NX_FLOAT* <=

specimen: (required) *NXsample* <=

Description of the specimen that was cut off from the sample.

In atom probe jargon this is typically referred to as the tip.

identifierNAME: (recommended) *NX_CHAR* <=

is_simulation: (required) *NX_BOOLEAN*

False, if the specimen is a real one. True, if the specimen is a virtual one.

alias: (recommended) *NX_CHAR*

Given name or an alias. Better use identifierNAME and identifier_parent instead.

A single NXentry should be used only for the characterization of a single specimen.

identifier_parent: (recommended) *NX_CHAR* <=

Identifier of the sample from which the specimen was cut or the string "n/a".

The purpose of this field is to support functionalities for tracking sample provenance via a research data management system.

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. 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 environmentally sensitive material such as hydrogen charged specimens or experiments including tracers with a short half time.

atom_types: (required) *NX_CHAR*

List of comma-separated elements from the IUPAC periodic table that are contained in the specimen. If the specimen 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 data instances.

description: (optional) *NX_CHAR* <=

Discouraged free-text field.

is_polycrystalline: (recommended) *NX_BOOLEAN*

True, if the specimen contains a grain or phase boundary. False, if the specimen is a single crystal.

is_amorphous: (recommended) *NX_BOOLEAN*

True, if the specimen is amorphous. False, if the specimen is not.

initial_radius: (recommended) *NX_FLOAT* {units=*NX_LENGTH*}

Ideally measured otherwise best elaborated guess of the initial radius of the specimen.

shank_angle: (recommended) *NX_FLOAT* {units=*NX_ANGLE*}

Ideally measured, otherwise best estimate, of the initial shank angle.

This is a measure of the specimen taper. Define it in such a way that the base of the specimen is modelled as a conical frustrum so that the shank angle is the smallest angle between the specimen space z-axis and a vector on the lateral surface of the cone.

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 Tait-Bryan angles.

Any of these values:

- `zxz`
- `xyx`
- `yzy`
- `zyz`
- `xzx`

- `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`

NAMED_**reference_frame**ID: (required) *NXcoordinate_system*

A coordinate system. Multiple instances require unique names.

Several Euclidean coordinate systems (CS) are used in the field of atom probe:

- 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. Suggested name of the group `laboratory_reference_frame`.
- The specimen space; a CS affixed to either the base or the initial apex of the specimen, whose z axis points towards the detector. Suggested name of the group `specimen_reference_frame`.
- 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. Suggested name of the group `detector_reference_frame`.
- 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. Suggested name of the group `reconstruction_reference_frame`.

To achieve unique names, the prefix "NAMED" should be replaced to with something derived from an alias for the coordinate system, or the value of the "alias" field.

Use the suffix _reference_frame when creating specific instances of NXcoordinate_system e.g. laboratory_reference_frame, reconstruction_reference_frame and so on and so forth.

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 identifier interval. Typically, the set of ion evaporation sequence identifiers 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.

To avoid these ambiguities, instances of *NXtransformations* should be used.

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* <=

measurement: (optional) *NXapm_measurement*

status: (recommended) *NX_CHAR* <=

quality: (recommended) *NX_CHAR* <=

instrument: (required) *NXapm_instrument* <=

type: (recommended) *NX_CHAR* <=

location: (recommended) *NX_CHAR* <=

flight_path: (recommended) *NX_FLOAT* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

reflectron: (optional) *NXcomponent* <=

applied: (required) *NX_BOOLEAN* <=

local_electrode: (recommended) *NXelectromagnetic_lens*

name: (required) *NX_CHAR* <=

aperture_type: (recommended) *NX_CHAR*

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

ion_detector: (recommended) *NXdetector* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

pulser: (recommended) *NXcomponent* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

sourceID: (optional) *NXsource* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

stage: (optional) *NXmanipulator* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

analysis_chamber: (optional) *NXcomponent* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

buffer_chamber: (optional) *NXcomponent* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

load_lock_chamber: (optional) *NXcomponent* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

getter_pump: (optional) *NXpump* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

roughening_pump: (optional) *NXpump* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

turbomolecular_pump: (optional) *NXpump* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

eventID: (optional) *NXapm_event_data* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

instrument: (recommended) *NXapm_instrument* <=

reflectron: (recommended) *NXcomponent* <=

voltage: (required) *NX_FLOAT* <=

local_electrode: (recommended) *NXelectromagnetic_lens*

voltage: (required) *NX_FLOAT*

pulser: (recommended) *NXcomponent* <=

pulse_mode: (required) *NX_CHAR* <=

pulse_frequency: (required) *NX_FLOAT* <=

pulse_fraction: (required) *NX_FLOAT* <=

pulse_voltage: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) <=

pulse_number: (optional) *NX_UINT* (Rank: 1, Dimensions: [n]) <=

standing_voltage: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n]) <=

sourceID: (optional) *NXsource* <=

wavelength: (recommended) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=

power: (required) *NX_FLOAT* {units=*NX_POWER*} <=

pulse_energy: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n]) <=

stage: (required) *NXmanipulator* <=

temperature_sensor: (required) *NXsensor* <=

measurement: (required) *NX_CHAR* <=

value: (required) *NX_FLOAT* <=

analysis_chamber: (required) *NXcomponent* <=

pressure_sensor: (required) *NXsensor* <=

measurement: (required) *NX_CHAR* <=

value: (required) *NX_FLOAT* <=

control: (recommended) *NXparameters* <=

evaporation_control: (required) *NX_CHAR* <=

target_detection_rate: (required) *NX_NUMBER* <=

simulation: (optional) *NXapm_simulation*

atom_probeID: (optional) *NXroi_process*

A region-of-interest analyzed either during or after the session for which specific processed data of the measured or simulated data are available.

If a single instance is required the group should be named atom_probe. If multiple groups are required these should be named atom_probe1, atom_probe2, and so on and so forth. **initial_specimen**: (recommended) *NXimage*

SEM or TEM image of the initial specimen taken before the measurement.

image_2d: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

real: (required) *NX_NUMBER* <=

axis_j: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) <=

@long_name: (required) *NX_CHAR* <=

axis_i: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) <=

@long_name: (required) *NX_CHAR* <=

final_specimen: (recommended) *NXimage*

SEM or TEM image of the final specimen taken after completion of the measurement. **image_2d**: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

real: (required) *NX_NUMBER* <=

axis_j: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_j]) <=

@**long_name**: (required) *NX_CHAR* <=

axis_i: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) <=

@**long_name**: (required) *NX_CHAR* <=

raw_data: (optional) *NXprocess* <=

Document the control software that was used on the instrument with which raw data were collected.

For almost all atom probe instruments, the recorded raw data and metadata follow proprietary semantics. Therefore, this group can currently often not be filled with more than the control software and some pointing to digital artifacts (e.g. proprietary files) that have been collected during the measurement in an effort to document as best as possible all steps of an analysis workflow.

The physical quantities measured in an atom probe experiment are time-of-flight and tuples of arrival_time_pairs as a function of the event chain on the pulser. From these tuples, hits are computed in a process called hit_finding.

sequence_index: (recommended) *NX_POSINT* <=

number_of_dld_wires: (recommended) *NX_UINT* {units=*NX_UNITLESS*}

The number of delay-line-detector (DLD) wires present.

Any of these values: `1 | 2 | 3`

dld_wire_names: (optional) *NX_CHAR* (Rank: 2, Dimensions: [n_dld, 2])

Alias tuple, typical for the begin and the end of each DLD wire of the detector. Order follows arrival_time_pairs.

The order of the first dimension should match that of the second dimension of the arrival_time_pairs field.

arrival_time_pairs: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [p, n_dld, 2]) {units=*NX_TIME*}

Raw readings from the analog-to-digital-converter timing circuits of the detector wires.

programID: (optional) *NXprogram*

The control software that was used for running the measurement.

At least the main software should be reported. If this is the only program to report name the group "program" and use its below fields program and version to detail the version used. E.g. program AP Suite, version 6.3

It is recommended to report multiple programs though, i.e. also the libraries and dependencies of the software. In the case of AP Suite/IVAS this can be used to document the AP Suite GUI, LAS, CamecaRoot, and CernRoot versions. In this case always name the program groups program1, program2, … with program one being AP Suite/IVAS.

In the case of an open-source instrument, like P. Felfer's Oxcart or G. Schmitz's M-TAP instruments, also use program1, program2, … with program1 representing the control software e.g. M. Monajem and P. Felfer PYCCAPT. Further instances (program2, …) can be used to list the dependencies, the python virtual environment.

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

source: (recommended) *NXnote* <=

Possibility to point to files that contain raw data.

Exemplar files that could be pointed to here when working with AMETEK/Cameca instruments are the proprietary STR, RRAW, or HITS files that AP Suite/IVAS generates.

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

hit_finding: (recommended) *NXprocess* <=

The configuration of a hit finding algorithm and its output.

Hit finding is the process of deciding which detector signals are significant and assigning specific ions colliding with the detector to each observed event.

sequence_index: (recommended) *NX_POSINT* <=

hit_positions: (recommended) *NX_NUMBER* (Rank: 2, Dimensions: [p_out, 2]) {units=*NX_LENGTH*}

Evaluated ion impact coordinates on the detector. Use the depends_on field to specify which reference frame the positions are defined in.

@depends_on: (required) *NX_CHAR*

Contains the path to an instance of NX_coordinate_system in which the positions are defined.

total_event_golden: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of events of type "golden" when APSuite/IVAS was used as the software with which the measurement was performed.

The value can be extracted from the CRunHeader.fTotalEventGolden field of a CamecaRoot RHIT/HITS file.

total_event_incomplete: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of events of type "incomplete" when APSuite/IVAS was used as the software with which the measurement was performed.

The value can be extracted from the CRunHeader.fTotalEventIncomplete field of a CamecaRoot RHIT/HITS file.

total_event_multiple: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of events of type "multiple" when APSuite/IVAS was used as the software with which the measurement was performed.

The value can be extracted from the CRunHeader.fTotalEventMultiple field of a CamecaRoot RHIT/HITS file.

total_event_partials: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of events of type "partials" when APSuite/IVAS was used as the software with which the measurement was performed.

The value can be extracted from the CRunHeader.fTotalEventPartials field of a CamecaRoot RHIT/HITS file.

total_event_record: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of events when APSuite/IVAS was used as the software with which the measurement was performed.

The value can be extracted from the CRunHeader.fTotalEventRecords field of a CamecaRoot RHIT/HITS file.

hit_quality_type: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_ht])

Hit quality is an integer that specifies which category/type a hit was assigned to. This field lists the human-readable, possibly though proprietary types distinguished. The indices of this array are used in hit_quality to encode hit_quality for each pulse in a more efficient way than by repeating the string that is used for each type as it is provided in this field.

As an example, assume two types, "golden" and "partial", are distinguished. If hit_quality_type stores the array "golden", "partial", the index 0 in hit_quality identifies all those pulses of category "golden", while the index 1 in hit_quality identifies all of category "partial".

hit_quality: (optional) *NX_UINT* (Rank: 1, Dimensions: [p_out]) {units=*NX_UNITLESS*}

Hit quality identifier for each pulse. Identifier has to be within hit_quality_type.

hit_multiplicity: (optional) *NX_UINT* (Rank: 1, Dimensions: [p_out]) {units=*NX_UNITLESS*}

The number of ions determined to have been collected on the same pulse. These ions may hit different pixels, or even the same detector pixel. The hit_multiplicity is not related to the makeup of the ions and should not be confused with the number of atoms or elements that constitute a molecular ion.

programID: (optional) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

config: (recommended) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

hit_spatial_filtering: (recommended) *NXprocess* <=

sequence_index: (recommended) *NX_POSINT* <=

evaporation_id_offset: (required) *NX_INT* {units=*NX_UNITLESS*}

Integer which defines the first evaporation_id. Typically, this is either zero or one.

evaporation_id: (required) *NX_INT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

There are two possibilities to report evaporation_id values:

If evaporation_id_offset is provided, the evaporation_id values are defined by the sequence $[evaporation_id_offset, evaporation_id_offset + n]$ with n the number of ions in the reconstructed volume.

Alternatively, evaporation_id_offset is not provided but instead a a sequence of n values is defined, these integer values do not need to be sorted.

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

source: (optional) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

hit_filter: (recommended) *NXcs_filter_boolean_mask*

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

voltage_and_bowl: (recommended) *NXprocess* <=

Configuration of and results obtained from a voltage-and-bowl time-of-flight correction algorithm.

The voltage-and-bowl correction is a data post-processing step to correct ion impact positions for flight path differences, detector bias, and nonlinearities.

sequence_index: (recommended) *NX_POSINT* <=

raw_tof: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [n])

Raw time-of-flight data without corrections.

tof_zero_estimate: (optional) *NX_FLOAT* {units=*NX_TIME*}

The parameter t_0, CAnalysis.CCalibMass.fT0Estimate

calibrated_tof: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [n])

Calibrated time-of-flight.

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

source: (optional) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

config: (required) *NXparameters* <=

correction_peak: (recommended) *NX_FLOAT* {units=*NX_ANY*}

Reference mass-to-charge state ratio value

For example 16 Da as mentioned by T. Blum et al. (page 371).

mass_to_charge_conversion: (recommended) *NXprocess* <=

sequence_index: (recommended) *NX_POSINT* <=

mass_to_charge: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n]) {units=*NX_ANY*}

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

source: (recommended) *NXnote* <=

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

config: (recommended) *NXparameters* <=

mass_calibration: (recommended) *NX_FLOAT* {units=*NX_ANY*}

Mass calibration with unit peaks/interp. as mentioned by T. Blum et al. (page 371).

mass_resolution: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [m_r]) {units=*NX_ANY*}

Inverse of the mass resolution $\frac{M}{\Delta M}$ as mentioned by T. Blum et al. (page 371).

Multiple values can be reported but reporting each is only useful when stating also:

- **The full width at which ΔM_{fw} fraction of maximum this value was defined.**
  Examples are at tenth ΔM_{10} or at half maximum (FWHM).

Consequently, mass_resolution_fw should needs to be a vector of the same length and using the same order like used for mass_resolution, i.e. the first mass resolution was defined at the maximum as defined by the first value from mass_resolution_fw.

- **The reference molecular ion e.g.** $^{16}O_2{}^+$
  As many instances of mass_resolutionION should be used with instances numbered starting from 1 up to the length of the mass_resolution vector.

mass_resolution_fw: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [m_r]) {units=*NX_ANY*}

The full width at which ΔM_{fw} fraction of maximum this value was defined. Examples are at tenth ΔM_{10} or at half maximum (FWHM). Consequently, mass_resolution_fw should needs to be a vector of the same length and using the same order like used for mass_resolution, i.e. the first mass resolution was defined at the maximum as defined by the first value from mass_resolution_fw.

mass_resolutionION: (recommended) *NXatom*

The reference molecular ion e.g. $^{16}O_2{}^+$ As many instances of mass_resolutionION should be used with instances numbered starting from 1 up to the length of the mass_resolution vector.

nuclide_hash: (recommended) *NX_UINT* <=

name: (required) *NX_CHAR* <=

reconstruction: (recommended) *NXapm_reconstruction*

sequence_index: (recommended) *NX_POSINT* <=

reconstructed_positions: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n, 3]) <=

volume: (recommended) *NX_FLOAT* <=

field_of_view: (recommended) *NX_FLOAT* <=

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

source: (recommended) *NXnote* <=

For LEAP and APSuite/IVAS-based analyses the root file which stores the settings whereby an RHIT/HITS file can be used to regenerate the reconstructed volume that is here referred to.

The respective RHIT/HITS file should ideally be specified in the serialized group of the hit_finding section of this application definition.

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

results: (recommended) *NXnote* <=

For LEAP and APSuite/IVAS-based analyses the resulting typically file with the reconstructed positions and calibrated mass-to-charge- state ratio values.

For other data collection/analysis software the data artifact which comes closest conceptually to AMETEK/Cameca's typical file formats.

These are typically exported as a POS, ePOS, APT, ATO, ENV, or HDF5 file, which should be stored alongside this record in the research data management system.

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

config: (recommended) *NXparameters* <=

voltage_filter_initial: (recommended) *NX_FLOAT* <=

voltage_filter_final: (recommended) *NX_FLOAT* <=

protocol_name: (recommended) *NX_CHAR* <=

primary_element: (recommended) *NX_CHAR* <=

efficiency: (recommended) *NX_FLOAT* <=

flight_path: (recommended) *NX_FLOAT* <=

evaporation_field: (recommended) *NX_CHAR*

image_compression: (recommended) *NX_FLOAT* <=

kfactor: (recommended) *NX_FLOAT* <=

shank_angle: (recommended) *NX_FLOAT* <=

ion_volume: (recommended) *NX_FLOAT* <=

crystallographic_calibration: (recommended) *NX_CHAR* <=

comment: (recommended) *NX_CHAR* <=

naive_discretization: (required) *NXprocess* <=

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@**version**: (required) *NX_CHAR* <=

DATA: (required) *NXdata* <=

@**signal**: (required) *NX_CHAR* <=

@**axes**: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

intensity: (required) *NX_NUMBER* (Rank: 3, Dimensions: [n_z, n_y, n_x])

axis_z: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_z])

@long_name: (required) *NX_CHAR*

axis_y: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_y])

@long_name: (required) *NX_CHAR*

axis_x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_x])

@long_name: (required) *NX_CHAR*

ranging: (recommended) *NXapm_ranging*

sequence_index: (recommended) *NX_POSINT* <=

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

source: (recommended) *NXnote* <=

The respective ranging definitions file RNG/RRNG/ENV/HDF5.

type: (recommended) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

mass_to_charge_distribution: (recommended) *NXprocess* <=

sequence_index: (recommended) *NX_POSINT* <=

min_mass_to_charge: (required) *NX_FLOAT* <=

max_mass_to_charge: (required) *NX_FLOAT* <=

n_mass_to_charge: (required) *NX_POSINT* <=

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

mass_spectrum: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

intensity: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_bins])

@long_name: (required) *NX_CHAR*

axis_mass_to_charge: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_bins])

@long_name: (required) *NX_CHAR*

background_quantification: (recommended) *NXprocess* <=

sequence_index: (recommended) *NX_POSINT* <=

background: (recommended) *NX_FLOAT* {units=*NX_ANY*}

(Out-of-sync, time-independent) background levels in ppm/ns reported by e.g. APSuite/IVAS for LEAP systems.

mrp_value: (recommended) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

The mass-resolving power (MRP) value

D. Larson et al. report Eq. D.8 in page 282:

$MRP = \frac{1}{2\delta t} \cdot \sqrt{\frac{m}{n}\frac{1}{2eV}L}$,

with δt representing the timing imprecision, $\frac{m}{n}$ the mass-to-charge state ratio, e the elementary charge, V the potential difference, and L the flight path length.

Timing imprecision is caused by variations of flight path length and voltage, the fact that the precision of electronics is finite and a spread of the time-of-departure of individual ions is observed.

mrp_mass_to_charge: (recommended) *NX_FLOAT* {units=*NX_ANY*}

Mass-to-charge state ratio $\frac{m}{n}$ at which mrp_value was specified.

mrp_voltage: (recommended) *NX_FLOAT* {units=*NX_VOLTAGE*}

Potential difference V at which mrp_value was specified.

mrp_flight_path_length: (recommended) *NX_FLOAT* {units=*NX_LENGTH*}

Flight path length L at which mrp_value was specified.

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

peak_search: (recommended) *NXprocess*

sequence_index: (recommended) *NX_POSINT* <=

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

peakID: (optional) *NXpeak*

label: (recommended) *NX_CHAR* <=

description: (recommended) *NX_CHAR*

category: (recommended) *NX_CHAR*

Category for the peak offering a qualitative statement of the location of the peak in light of limited mass-resolving power that is relevant for composition quantification. See D. Larson et al. (p172) for examples of each category:

- 0, well-separated, $^{10}B^{+}$, $^{28}Si^{2+}$
- 1, close, but can be sufficiently separated for quantification in a LEAP system, $^{94}Mo^{3+}$, $^{63}Cu^{2+}$
- 2, closely overlapping, demands better than LEAP4000X MRP can provide $^{14}N^{+}$, $^{28}Si^{2+}$ at different charge states
- 3, overlapped exactly due to multi-charge molecular species, $^{16}{O_2}^{2+}$, $^{16}O^{+}$
- 4, overlapped, same charge state, cannot as of 2013 be discriminated with a LEAP4000X, $^{14}{N_2}^{+}$, $^{28}Si^{+}$
- 5, overlapped, same charge state, any expectation of resolvability, $^{54}Cr^{2+}$, $^{54}Fe^{2+}$

Any of these values: `0 | 1 | 2 | 3 | 4 | 5`

position: (required) *NX_NUMBER*

peak_identification: (recommended) *NXprocess* <=

sequence_index: (recommended) *NX_POSINT* <=

number_of_ion_types: (required) *NX_UINT* <=

maximum_number_of_atoms_per_molecular_ion: (required) *NX_UINT* <=

iontypes: (recommended) *NX_UINT* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

The iontype identifier for each ion that was best matching; stored in the order of the evaporation_id.

The value zero is reserved for documenting that an ion was unranged. Identifier for ranged ions need to start at 1 up to number_of_ion_types.

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

ionID: (required) *NXatom* <=

Ions that were ranged.

The value zero is reserved for documenting that an ion was unranged. Identifier for ranged ions need to start at 1 up to number_of_ion_types.

nuclide_hash: (required) *NX_UINT* <=

charge_state: (required) *NX_INT*

mass_to_charge_range: (required) *NX_FLOAT*

nuclide_list: (recommended) *NX_UINT* <=

name: (recommended) *NX_CHAR* <=

charge_state_analysis: (optional) *NXapm_charge_state_analysis*

charge_state: (required) *NX_INT* <=

nuclide_hash: (required) *NX_UINT* <=

mass: (required) *NX_FLOAT* <=

natural_abundance_product: (required) *NX_FLOAT* <=

shortest_half_life: (required) *NX_FLOAT* <=

config: (required) *NXparameters* <=

nuclides: (required) *NX_UINT* <=

mass_to_charge_range: (required) *NX_FLOAT* <=

min_half_life: (required) *NX_FLOAT* <=

min_abundance: (required) *NX_FLOAT* <=

sacrifice_isotopic_uniqueness: (required) *NX_BOOLEAN* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm/ENTRY-group*
- */NXapm/ENTRY/atom_probeID-group*
- */NXapm/ENTRY/atom_probeID/final_specimen-group*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d-group*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d/axis_i-field*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d/axis_i@long_name-attribute*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d/axis_j-field*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d/axis_j@long_name-attribute*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d/real-field*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d@axes-attribute*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d@AXISNAME_indices-attribute*
- */NXapm/ENTRY/atom_probeID/final_specimen/image_2d@signal-attribute*
- */NXapm/ENTRY/atom_probeID/hit_finding-group*
- */NXapm/ENTRY/atom_probeID/hit_finding/config-group*
- */NXapm/ENTRY/atom_probeID/hit_finding/config/algorithm-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/config/checksum-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/config/file_name-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/config/type-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/hit_multiplicity-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/hit_positions-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/hit_positions@depends_on-attribute*

- */NXapm/ENTRY/atom_probeID/hit_finding/hit_quality-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/hit_quality_type-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/programID-group*
- */NXapm/ENTRY/atom_probeID/hit_finding/programID/program-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/hit_finding/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/total_event_golden-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/total_event_incomplete-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/total_event_multiple-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/total_event_partials-field*
- */NXapm/ENTRY/atom_probeID/hit_finding/total_event_record-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering-group*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/evaporation_id-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/evaporation_id_offset-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/hit_filter-group*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/hit_filter/bitdepth-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/hit_filter/mask-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/hit_filter/number_of_objects-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/programID-group*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/programID/program-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/source-group*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/source/algorithm-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/source/checksum-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/source/file_name-field*
- */NXapm/ENTRY/atom_probeID/hit_spatial_filtering/source/type-field*
- */NXapm/ENTRY/atom_probeID/initial_specimen-group*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d-group*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d/axis_i-field*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d/axis_i@long_name-attribute*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d/axis_j-field*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d/axis_j@long_name-attribute*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d/real-field*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d@axes-attribute*
- */NXapm/ENTRY/atom_probeID/initial_specimen/image_2d@AXISNAME_indices-attribute*

- /NXapm/ENTRY/atom_probeID/initial_specimen/image_2d@signal-attribute
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion-group
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config-group
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config/mass_calibration-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config/mass_resolution-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config/mass_resolution_fw-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config/mass_resolutionION-group
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config/mass_resolutionION/name-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/config/mass_resolutionION/nuclide_hash-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/mass_to_charge-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/programID-group
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/programID/program-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/programID/program@version-attribute
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/sequence_index-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/source-group
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/source/algorithm-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/source/checksum-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/source/file_name-field
- /NXapm/ENTRY/atom_probeID/mass_to_charge_conversion/source/type-field
- /NXapm/ENTRY/atom_probeID/ranging-group
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification-group
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/background-field
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/mrp_flight_path_length-field
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/mrp_mass_to_charge-field
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/mrp_value-field
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/mrp_voltage-field
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/programID-group
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/programID/program-field
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/programID/program@version-attribute
- /NXapm/ENTRY/atom_probeID/ranging/background_quantification/sequence_index-field
- /NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution-group
- /NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum-group
- /NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum/axis_mass_to_charge-field
- /NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum/axis_mass_to_charge@long_name-attribute
- /NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum/intensity-field

- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum/intensity@long_name-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum/title-field*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum@axes-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum@AXISNAME_indices-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/mass_spectrum@signal-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/max_mass_to_charge-field*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/min_mass_to_charge-field*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/n_mass_to_charge-field*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/programID-group*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/programID/program-field*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/mass_to_charge_distribution/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/charge_state-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/config-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/config/mass_to_charge_range-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/config/min_abundance-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/config/min_half_life-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/config/nuclides-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/config/sacrifice_isotopic_uniqueness-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/mass-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/natural_abundance_product-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/nuclide_hash-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/charge_state_analysis/shortest_half_life-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/mass_to_charge_range-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/name-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/nuclide_hash-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/ionID/nuclide_list-field*

- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/iontypes-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/maximum_number_of_atoms_per_molecular_ion-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/number_of_ion_types-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/programID-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/programID/program-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/peak_identification/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/peakID-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/peakID/category-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/peakID/description-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/peakID/label-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/peakID/position-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/programID-group*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/programID/program-field*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/peak_search/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/ranging/programID-group*
- */NXapm/ENTRY/atom_probeID/ranging/programID/program-field*
- */NXapm/ENTRY/atom_probeID/ranging/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/ranging/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/ranging/source-group*
- */NXapm/ENTRY/atom_probeID/ranging/source/algorithm-field*
- */NXapm/ENTRY/atom_probeID/ranging/source/checksum-field*
- */NXapm/ENTRY/atom_probeID/ranging/source/file_name-field*
- */NXapm/ENTRY/atom_probeID/ranging/source/type-field*
- */NXapm/ENTRY/atom_probeID/raw_data-group*
- */NXapm/ENTRY/atom_probeID/raw_data/arrival_time_pairs-field*
- */NXapm/ENTRY/atom_probeID/raw_data/dld_wire_names-field*
- */NXapm/ENTRY/atom_probeID/raw_data/number_of_dld_wires-field*
- */NXapm/ENTRY/atom_probeID/raw_data/programID-group*
- */NXapm/ENTRY/atom_probeID/raw_data/programID/program-field*
- */NXapm/ENTRY/atom_probeID/raw_data/programID/program@version-attribute*
- */NXapm/ENTRY/atom_probeID/raw_data/sequence_index-field*
- */NXapm/ENTRY/atom_probeID/raw_data/source-group*

- /NXapm/ENTRY/atom_probeID/raw_data/source/algorithm-field
- /NXapm/ENTRY/atom_probeID/raw_data/source/checksum-field
- /NXapm/ENTRY/atom_probeID/raw_data/source/file_name-field
- /NXapm/ENTRY/atom_probeID/raw_data/source/type-field
- /NXapm/ENTRY/atom_probeID/reconstruction-group
- /NXapm/ENTRY/atom_probeID/reconstruction/config-group
- /NXapm/ENTRY/atom_probeID/reconstruction/config/comment-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/crystallographic_calibration-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/efficiency-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/evaporation_field-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/flight_path-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/image_compression-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/ion_volume-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/kfactor-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/primary_element-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/protocol_name-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/shank_angle-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/voltage_filter_final-field
- /NXapm/ENTRY/atom_probeID/reconstruction/config/voltage_filter_initial-field
- /NXapm/ENTRY/atom_probeID/reconstruction/field_of_view-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization-group
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA-group
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/axis_x-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/axis_x@long_name-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/axis_y-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/axis_y@long_name-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/axis_z-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/axis_z@long_name-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/intensity-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA/title-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA@axes-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA@AXISNAME_indices-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/DATA@signal-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/programID-group
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/programID/program-field
- /NXapm/ENTRY/atom_probeID/reconstruction/naive_discretization/programID/program@version-attribute

- /NXapm/ENTRY/atom_probeID/reconstruction/programID-group
- /NXapm/ENTRY/atom_probeID/reconstruction/programID/program-field
- /NXapm/ENTRY/atom_probeID/reconstruction/programID/program@version-attribute
- /NXapm/ENTRY/atom_probeID/reconstruction/reconstructed_positions-field
- /NXapm/ENTRY/atom_probeID/reconstruction/results-group
- /NXapm/ENTRY/atom_probeID/reconstruction/results/algorithm-field
- /NXapm/ENTRY/atom_probeID/reconstruction/results/checksum-field
- /NXapm/ENTRY/atom_probeID/reconstruction/results/file_name-field
- /NXapm/ENTRY/atom_probeID/reconstruction/results/type-field
- /NXapm/ENTRY/atom_probeID/reconstruction/sequence_index-field
- /NXapm/ENTRY/atom_probeID/reconstruction/source-group
- /NXapm/ENTRY/atom_probeID/reconstruction/source/algorithm-field
- /NXapm/ENTRY/atom_probeID/reconstruction/source/checksum-field
- /NXapm/ENTRY/atom_probeID/reconstruction/source/file_name-field
- /NXapm/ENTRY/atom_probeID/reconstruction/source/type-field
- /NXapm/ENTRY/atom_probeID/reconstruction/volume-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl-group
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/calibrated_tof-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/config-group
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/config/correction_peak-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/programID-group
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/programID/program-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/programID/program@version-attribute
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/raw_tof-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/sequence_index-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/source-group
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/source/algorithm-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/source/checksum-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/source/file_name-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/source/type-field
- /NXapm/ENTRY/atom_probeID/voltage_and_bowl/tof_zero_estimate-field
- /NXapm/ENTRY/citeID-group
- /NXapm/ENTRY/citeID/author-field
- /NXapm/ENTRY/citeID/doi-field
- /NXapm/ENTRY/consistent_rotations-group
- /NXapm/ENTRY/consistent_rotations/axis_angle_convention-field

- /NXapm/ENTRY/consistent_rotations/euler_angle_convention-field
- /NXapm/ENTRY/consistent_rotations/rotation_convention-field
- /NXapm/ENTRY/consistent_rotations/rotation_handedness-field
- /NXapm/ENTRY/consistent_rotations/sign_convention-field
- /NXapm/ENTRY/definition-field
- /NXapm/ENTRY/definition@version-attribute
- /NXapm/ENTRY/elapsed_time-field
- /NXapm/ENTRY/end_time-field
- /NXapm/ENTRY/experiment_alias-field
- /NXapm/ENTRY/experiment_description-field
- /NXapm/ENTRY/measurement-group
- /NXapm/ENTRY/measurement/eventID-group
- /NXapm/ENTRY/measurement/eventID/end_time-field
- /NXapm/ENTRY/measurement/eventID/instrument-group
- /NXapm/ENTRY/measurement/eventID/instrument/analysis_chamber-group
- /NXapm/ENTRY/measurement/eventID/instrument/analysis_chamber/pressure_sensor-group
- /NXapm/ENTRY/measurement/eventID/instrument/analysis_chamber/pressure_sensor/measurement-field
- /NXapm/ENTRY/measurement/eventID/instrument/analysis_chamber/pressure_sensor/value-field
- /NXapm/ENTRY/measurement/eventID/instrument/control-group
- /NXapm/ENTRY/measurement/eventID/instrument/control/evaporation_control-field
- /NXapm/ENTRY/measurement/eventID/instrument/control/target_detection_rate-field
- /NXapm/ENTRY/measurement/eventID/instrument/local_electrode-group
- /NXapm/ENTRY/measurement/eventID/instrument/local_electrode/voltage-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser-group
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/pulse_fraction-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/pulse_frequency-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/pulse_mode-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/pulse_number-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/pulse_voltage-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/sourceID-group
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/sourceID/power-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/sourceID/pulse_energy-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/sourceID/wavelength-field
- /NXapm/ENTRY/measurement/eventID/instrument/pulser/standing_voltage-field
- /NXapm/ENTRY/measurement/eventID/instrument/reflectron-group
- /NXapm/ENTRY/measurement/eventID/instrument/reflectron/voltage-field

- /NXapm/ENTRY/measurement/eventID/instrument/stage-group
- /NXapm/ENTRY/measurement/eventID/instrument/stage/temperature_sensor-group
- /NXapm/ENTRY/measurement/eventID/instrument/stage/temperature_sensor/measurement-field
- /NXapm/ENTRY/measurement/eventID/instrument/stage/temperature_sensor/value-field
- /NXapm/ENTRY/measurement/eventID/start_time-field
- /NXapm/ENTRY/measurement/instrument-group
- /NXapm/ENTRY/measurement/instrument/analysis_chamber-group
- /NXapm/ENTRY/measurement/instrument/analysis_chamber/fabrication-group
- /NXapm/ENTRY/measurement/instrument/analysis_chamber/fabrication/model-field
- /NXapm/ENTRY/measurement/instrument/analysis_chamber/fabrication/serial_number-field
- /NXapm/ENTRY/measurement/instrument/analysis_chamber/fabrication/vendor-field
- /NXapm/ENTRY/measurement/instrument/buffer_chamber-group
- /NXapm/ENTRY/measurement/instrument/buffer_chamber/fabrication-group
- /NXapm/ENTRY/measurement/instrument/buffer_chamber/fabrication/model-field
- /NXapm/ENTRY/measurement/instrument/buffer_chamber/fabrication/serial_number-field
- /NXapm/ENTRY/measurement/instrument/buffer_chamber/fabrication/vendor-field
- /NXapm/ENTRY/measurement/instrument/fabrication-group
- /NXapm/ENTRY/measurement/instrument/fabrication/model-field
- /NXapm/ENTRY/measurement/instrument/fabrication/serial_number-field
- /NXapm/ENTRY/measurement/instrument/fabrication/vendor-field
- /NXapm/ENTRY/measurement/instrument/flight_path-field
- /NXapm/ENTRY/measurement/instrument/getter_pump-group
- /NXapm/ENTRY/measurement/instrument/getter_pump/fabrication-group
- /NXapm/ENTRY/measurement/instrument/getter_pump/fabrication/model-field
- /NXapm/ENTRY/measurement/instrument/getter_pump/fabrication/serial_number-field
- /NXapm/ENTRY/measurement/instrument/getter_pump/fabrication/vendor-field
- /NXapm/ENTRY/measurement/instrument/ion_detector-group
- /NXapm/ENTRY/measurement/instrument/ion_detector/fabrication-group
- /NXapm/ENTRY/measurement/instrument/ion_detector/fabrication/model-field
- /NXapm/ENTRY/measurement/instrument/ion_detector/fabrication/serial_number-field
- /NXapm/ENTRY/measurement/instrument/ion_detector/fabrication/vendor-field
- /NXapm/ENTRY/measurement/instrument/load_lock_chamber-group
- /NXapm/ENTRY/measurement/instrument/load_lock_chamber/fabrication-group
- /NXapm/ENTRY/measurement/instrument/load_lock_chamber/fabrication/model-field
- /NXapm/ENTRY/measurement/instrument/load_lock_chamber/fabrication/serial_number-field
- /NXapm/ENTRY/measurement/instrument/load_lock_chamber/fabrication/vendor-field

- */NXapm/ENTRY/measurement/instrument/local_electrode-group*
- */NXapm/ENTRY/measurement/instrument/local_electrode/aperture_type-field*
- */NXapm/ENTRY/measurement/instrument/local_electrode/fabrication-group*
- */NXapm/ENTRY/measurement/instrument/local_electrode/fabrication/model-field*
- */NXapm/ENTRY/measurement/instrument/local_electrode/fabrication/serial_number-field*
- */NXapm/ENTRY/measurement/instrument/local_electrode/fabrication/vendor-field*
- */NXapm/ENTRY/measurement/instrument/local_electrode/name-field*
- */NXapm/ENTRY/measurement/instrument/location-field*
- */NXapm/ENTRY/measurement/instrument/pulser-group*
- */NXapm/ENTRY/measurement/instrument/pulser/fabrication-group*
- */NXapm/ENTRY/measurement/instrument/pulser/fabrication/model-field*
- */NXapm/ENTRY/measurement/instrument/pulser/fabrication/serial_number-field*
- */NXapm/ENTRY/measurement/instrument/pulser/fabrication/vendor-field*
- */NXapm/ENTRY/measurement/instrument/pulser/sourceID-group*
- */NXapm/ENTRY/measurement/instrument/pulser/sourceID/fabrication-group*
- */NXapm/ENTRY/measurement/instrument/pulser/sourceID/fabrication/model-field*
- */NXapm/ENTRY/measurement/instrument/pulser/sourceID/fabrication/serial_number-field*
- */NXapm/ENTRY/measurement/instrument/pulser/sourceID/fabrication/vendor-field*
- */NXapm/ENTRY/measurement/instrument/reflectron-group*
- */NXapm/ENTRY/measurement/instrument/reflectron/applied-field*
- */NXapm/ENTRY/measurement/instrument/roughening_pump-group*
- */NXapm/ENTRY/measurement/instrument/roughening_pump/fabrication-group*
- */NXapm/ENTRY/measurement/instrument/roughening_pump/fabrication/model-field*
- */NXapm/ENTRY/measurement/instrument/roughening_pump/fabrication/serial_number-field*
- */NXapm/ENTRY/measurement/instrument/roughening_pump/fabrication/vendor-field*
- */NXapm/ENTRY/measurement/instrument/stage-group*
- */NXapm/ENTRY/measurement/instrument/stage/fabrication-group*
- */NXapm/ENTRY/measurement/instrument/stage/fabrication/model-field*
- */NXapm/ENTRY/measurement/instrument/stage/fabrication/serial_number-field*
- */NXapm/ENTRY/measurement/instrument/stage/fabrication/vendor-field*
- */NXapm/ENTRY/measurement/instrument/turbomolecular_pump-group*
- */NXapm/ENTRY/measurement/instrument/turbomolecular_pump/fabrication-group*
- */NXapm/ENTRY/measurement/instrument/turbomolecular_pump/fabrication/model-field*
- */NXapm/ENTRY/measurement/instrument/turbomolecular_pump/fabrication/serial_number-field*
- */NXapm/ENTRY/measurement/instrument/turbomolecular_pump/fabrication/vendor-field*
- */NXapm/ENTRY/measurement/instrument/type-field*

- */NXapm/ENTRY/measurement/quality-field*
- */NXapm/ENTRY/measurement/status-field*
- */NXapm/ENTRY/NAMED_reference_frameID-group*
- */NXapm/ENTRY/NAMED_reference_frameID/alias-field*
- */NXapm/ENTRY/NAMED_reference_frameID/origin-field*
- */NXapm/ENTRY/NAMED_reference_frameID/type-field*
- */NXapm/ENTRY/NAMED_reference_frameID/x-field*
- */NXapm/ENTRY/NAMED_reference_frameID/x_direction-field*
- */NXapm/ENTRY/NAMED_reference_frameID/y-field*
- */NXapm/ENTRY/NAMED_reference_frameID/y_direction-field*
- */NXapm/ENTRY/NAMED_reference_frameID/z-field*
- */NXapm/ENTRY/NAMED_reference_frameID/z_direction-field*
- */NXapm/ENTRY/noteID-group*
- */NXapm/ENTRY/noteID/algorithm-field*
- */NXapm/ENTRY/noteID/checksum-field*
- */NXapm/ENTRY/noteID/file_name-field*
- */NXapm/ENTRY/noteID/type-field*
- */NXapm/ENTRY/operation_mode-field*
- */NXapm/ENTRY/profiling-group*
- */NXapm/ENTRY/profiling/environment-group*
- */NXapm/ENTRY/profiling/environment/PROGRAM-group*
- */NXapm/ENTRY/profiling/environment/PROGRAM/program-field*
- */NXapm/ENTRY/profiling/environment/PROGRAM/program@version-attribute*
- */NXapm/ENTRY/profiling/programID-group*
- */NXapm/ENTRY/profiling/programID/program-field*
- */NXapm/ENTRY/profiling/programID/program@version-attribute*
- */NXapm/ENTRY/run_number-field*
- */NXapm/ENTRY/sample-group*
- */NXapm/ENTRY/sample/alias-field*
- */NXapm/ENTRY/sample/chemical_composition-group*
- */NXapm/ENTRY/sample/chemical_composition/ELEMENT-group*
- */NXapm/ENTRY/sample/chemical_composition/ELEMENT/chemical_symbol-field*
- */NXapm/ENTRY/sample/chemical_composition/ELEMENT/composition-field*
- */NXapm/ENTRY/sample/chemical_composition/ELEMENT/composition_errors-field*
- */NXapm/ENTRY/sample/chemical_composition/normalization-field*
- */NXapm/ENTRY/sample/description-field*

- */NXapm/ENTRY/sample/grain_diameter-field*
- */NXapm/ENTRY/sample/grain_diameter_errors-field*
- */NXapm/ENTRY/sample/heat_treatment_quenching_rate-field*
- */NXapm/ENTRY/sample/heat_treatment_quenching_rate_errors-field*
- */NXapm/ENTRY/sample/heat_treatment_temperature-field*
- */NXapm/ENTRY/sample/heat_treatment_temperature_errors-field*
- */NXapm/ENTRY/sample/heat_treatment_time-field*
- */NXapm/ENTRY/sample/identifierNAME-field*
- */NXapm/ENTRY/sample/is_simulation-field*
- */NXapm/ENTRY/simulation-group*
- */NXapm/ENTRY/specimen-group*
- */NXapm/ENTRY/specimen/alias-field*
- */NXapm/ENTRY/specimen/atom_types-field*
- */NXapm/ENTRY/specimen/description-field*
- */NXapm/ENTRY/specimen/identifier_parent-field*
- */NXapm/ENTRY/specimen/identifierNAME-field*
- */NXapm/ENTRY/specimen/initial_radius-field*
- */NXapm/ENTRY/specimen/is_amorphous-field*
- */NXapm/ENTRY/specimen/is_polycrystalline-field*
- */NXapm/ENTRY/specimen/is_simulation-field*
- */NXapm/ENTRY/specimen/preparation_date-field*
- */NXapm/ENTRY/specimen/shank_angle-field*
- */NXapm/ENTRY/start_time-field*
- */NXapm/ENTRY/userID-group*
- */NXapm/ENTRY/userID/identifierNAME-field*
- */NXapm/ENTRY/userID/identifierNAME@type-attribute*
- */NXapm/ENTRY/userID/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXapm.nxdl.xml

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.

This definition is a legacy support for older NXarpes experiments. There is, however, a newer definition to collect data & metadata for general photoemission experiments, called *NXmpes*, as well as a specialization for ARPES experiments, called *NXmpes_arpes*."

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

NXazint1d

Status:

application definition, extends *NXobject*

Description:

Application definition for data from two-dimensional area detectors that has been integrated azimuthally, with a certain radial binning in units of q or 2theta.

An example application that creates these files is documented here: https://maxiv-science.github.io/azint_writer/

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nImg: Number of integrated images

nRad: Number of radial bins

nRadEdge: Number of radial bin edges (nRad+1)

Groups cited:
NXdata, *NXentry*, *NXinstrument*, *NXmonitor*, *NXmonochromator*, *NXparameters*, *NXprocess*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

The `NXsubentry` or Multi-Method Data convention described here: https://manual.nexusformat.org/rules.html#table-nxsubentry should be used when different method (e.g. `NXcanSAS` or `NXmonopd`), `NXazint2d` or other `NXazint1d` data, integrated with different options, should be stored under the same `NXentry`.

In case of a single `NXazint1d` data processing the standard convention with the application definition directly under `NXentry` should be used.

@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.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms.

Obligatory value: `NXazint1d`

solid_angle_applied: (required) *NX_BOOLEAN*

is solid angle correction applied or not.

polarization_applied: (required) *NX_BOOLEAN*

is polarization correction applied or not.

normalization_applied: (required) *NX_BOOLEAN*

is a normalization correction applied or not.

It indicates that integrated intensities and their errors were already divided by the appropriate normalization factors accounting for the effective number or weighted contribution of detector pixels to each integration bin.

monitor_applied: (optional) *NX_BOOLEAN*

is a monitor correction applied or not.

The monitor correction accounts for external factors that are independent of the azimuthal integration process. Most commonly, this involves normalizing for fluctuations in the incident beam intensity or, where applicable, variations in exposure time.

INSTRUMENT: (required) *NXinstrument* <=

name: (required) *NX_CHAR* <=

Name of instrument (beamline) where data was collected.

MONOCHROMATOR: (required) *NXmonochromator* <=

wavelength: (required) *NX_FLOAT* {units=angstrom} <=

Wavelength in angstrom.

energy: (required) *NX_FLOAT* {units=keV} <=

Energy in keV.

SOURCE: (required) *NXsource* <=

name: (required) *NX_CHAR* <=

Name of laboratory where data was collected.

type: (required) *NX_CHAR* <=

Type of laboratory where data was collected.

probe: (required) *NX_CHAR* <=

Type of probe.

reduction: (required) *NXprocess* <=

program: (required) *NX_CHAR* <=

Name of the program that made this file.

version: (required) *NX_CHAR* <=

Version of the progam that made this file.

date: (required) *NX_DATE_TIME* <=

Date the file was created

reference: (required) *NX_CHAR*

Citation or other references for the algorithm used in the processing.

note: (optional) *NX_CHAR*

Notes required to help interpret the data, e.g. on coordinate systems.

input: (required) *NXparameters* <=

Parameters should exactly match those required by the algorithm used in the processing. For example, *azint* requires *error_model*, *mask*, *n_splitting*, *poni*, etc.

MONITOR: (optional) *NXmonitor* <=

Monitor data for example *I_zero*.

data: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nImg]) <=

DATA: (required) *NXdata* <=

Azimuthally integrated data with radial binning in q or 2theta.

@axes: (required) *NX_CHAR* <=

Obligatory value: `['.', 'radial_axis']`

@interpretation: (required) *NX_CHAR*

Obligatory value: `spectrum`

@signal: (required) *NX_CHAR* <=

Obligatory value: `I`

I: (required) *NX_NUMBER* (Rank: 2, Dimensions: [nImg, nRad])

@long_name: (required) *NX_CHAR*

Obligatory value: `intensity`

@units: (required) *NX_CHAR*

Obligatory value: `arbitrary units`

I_errors: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nImg, nRad]) <=

@long_name: (optional) *NX_CHAR*

Obligatory value: `estimated intensity error`

@units: (optional) *NX_CHAR*

Obligatory value: `arbitrary units`

radial_axis: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nRad]) <=

@long_name: (required) *NX_CHAR* <=

Any of these values: `q` | `2theta`

@units: (required) *NX_CHAR*

Any of these values: `NX_PER_LENGTH` | `NX_WAVENUMBER` | `NX_ANGLE`

radial_axis_edges: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nRadEdge]) <=

@long_name: (required) *NX_CHAR* <=

Any of these values: `q bin edges` | `2theta bin edges`

@units: (required) *NX_CHAR*

Any of these values: `NX_PER_LENGTH` | `NX_WAVENUMBER` | `NX_ANGLE`

norm: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nRad])

Values of the normalization correction.

The normalization correction accounts for the effective number or weighted contribution of detector pixels to each integration bin.

Note: An important aspect of the normalization strategy is how polarization and solid angle corrections are incorporated, which can vary depending on the specific application, software, and its configuration options (see, for example, PyFAI documentation). Additionally, the normalization strategy may include a relative or absolute calibration factor. Two common normalization approaches are: "Relative normalization" to the PONI (Point Of Normal Incidence) pixel, and "Absolute calibration", which yields the number of photons scattered by the sample in a given direction per unit solid angle. The type of the normalization strategy is not indicated on this level. It must be concluded from the software used or its parameters.

The monitor correction is not included in the normalization correction and it is specified separately.

@long_name: (required) *NX_CHAR*

Obligatory value:

- `effective number of pixels contributing to the corresponding bin`

@units: (required) *NX_CHAR*

Obligatory value: `arbitrary units`

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXazint1d/ENTRY-group*
- */NXazint1d/ENTRY/DATA-group*
- */NXazint1d/ENTRY/DATA/I-field*
- */NXazint1d/ENTRY/DATA/I@long_name-attribute*
- */NXazint1d/ENTRY/DATA/I@units-attribute*
- */NXazint1d/ENTRY/DATA/I_errors-field*
- */NXazint1d/ENTRY/DATA/I_errors@long_name-attribute*
- */NXazint1d/ENTRY/DATA/I_errors@units-attribute*
- */NXazint1d/ENTRY/DATA/norm-field*
- */NXazint1d/ENTRY/DATA/norm@long_name-attribute*
- */NXazint1d/ENTRY/DATA/norm@units-attribute*
- */NXazint1d/ENTRY/DATA/radial_axis-field*
- */NXazint1d/ENTRY/DATA/radial_axis@long_name-attribute*
- */NXazint1d/ENTRY/DATA/radial_axis@units-attribute*
- */NXazint1d/ENTRY/DATA/radial_axis_edges-field*
- */NXazint1d/ENTRY/DATA/radial_axis_edges@long_name-attribute*
- */NXazint1d/ENTRY/DATA/radial_axis_edges@units-attribute*
- */NXazint1d/ENTRY/DATA@axes-attribute*
- */NXazint1d/ENTRY/DATA@interpretation-attribute*
- */NXazint1d/ENTRY/DATA@signal-attribute*
- */NXazint1d/ENTRY/definition-field*
- */NXazint1d/ENTRY/INSTRUMENT-group*
- */NXazint1d/ENTRY/INSTRUMENT/MONOCHROMATOR-group*
- */NXazint1d/ENTRY/INSTRUMENT/MONOCHROMATOR/energy-field*
- */NXazint1d/ENTRY/INSTRUMENT/MONOCHROMATOR/wavelength-field*
- */NXazint1d/ENTRY/INSTRUMENT/name-field*
- */NXazint1d/ENTRY/INSTRUMENT/SOURCE-group*
- */NXazint1d/ENTRY/INSTRUMENT/SOURCE/name-field*
- */NXazint1d/ENTRY/INSTRUMENT/SOURCE/probe-field*
- */NXazint1d/ENTRY/INSTRUMENT/SOURCE/type-field*
- */NXazint1d/ENTRY/MONITOR-group*
- */NXazint1d/ENTRY/MONITOR/data-field*
- */NXazint1d/ENTRY/monitor_applied-field*
- */NXazint1d/ENTRY/normalization_applied-field*

- */NXazint1d/ENTRY/polarization_applied-field*
- */NXazint1d/ENTRY/reduction-group*
- */NXazint1d/ENTRY/reduction/date-field*
- */NXazint1d/ENTRY/reduction/input-group*
- */NXazint1d/ENTRY/reduction/note-field*
- */NXazint1d/ENTRY/reduction/program-field*
- */NXazint1d/ENTRY/reduction/reference-field*
- */NXazint1d/ENTRY/reduction/version-field*
- */NXazint1d/ENTRY/solid_angle_applied-field*
- */NXazint1d/ENTRY@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXazint1d.nxdl.xml

NXazint2d

Status:

application definition, extends *NXobject*

Description:

Application definition for data from two-dimensional area detectors that has been integrated azimuthally, with a certain radial binning in units of q or 2theta and with a binning around the azimuthal angle eta.

An example application that creates these files is documented here: https://maxiv-science.github.io/azint_writer/

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nImg: Number of integrated images

nRad: Number of radial bins

nRadEdge: Number of radial bin edges (nRad+1)

nEta: Number of azimuthal bins

nEtaEdge: Number of azimuthal bin edges (nEta+1)

Groups cited:
NXdata, *NXentry*, *NXinstrument*, *NXmonitor*, *NXmonochromator*, *NXparameters*, *NXprocess*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

The `NXsubentry` or Multi-Method Data convention described here: https://manual.nexusformat.org/rules.html#table-nxsubentry should be used when different method (e.g. `NXcanSAS` or `NXmonopd`), `NXazint1d` or other `NXazint2d` data, integrated with different options, should be stored under the same `NXentry`.

In case of a single `NXazint2d` data processing the standard convention with the application definition directly under `NXentry` should be used.

@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.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms.

Obligatory value: `NXazint2d`

solid_angle_applied: (required) *NX_BOOLEAN*

is solid angle correction applied or not.

polarization_applied: (required) *NX_BOOLEAN*

is polarization correction applied or not.

normalization_applied: (required) *NX_BOOLEAN*

is a normalization correction applied or not.

It indicates that integrated intensities and their errors were already divided by the appropriate normalization factors accounting for the effective number or weighted contribution of detector pixels to each integration bin.

monitor_applied: (optional) *NX_BOOLEAN*

is a monitor correction applied or not.

The monitor correction accounts for external factors that are independent of the azimuthal integration process. Most commonly, this involves normalizing for fluctuations in the incident beam intensity or, where applicable, variations in exposure time.

INSTRUMENT: (required) *NXinstrument* <=

name: (required) *NX_CHAR* <=

Name of instrument (beamline) where data was collected.

MONOCHROMATOR: (required) *NXmonochromator* <=

wavelength: (required) *NX_FLOAT* {units=angstrom} <=

Wavelength in angstrom.

energy: (required) *NX_FLOAT* {units=keV} <=

Energy in keV.

SOURCE: (required) *NXsource* <=

name: (required) *NX_CHAR* <=

Name of laboratory where data was collected.

type: (required) *NX_CHAR* <=

Type of laboratory where data was collected.

probe: (required) *NX_CHAR* <=

Type of probe.

reduction: (required) *NXprocess* <=

program: (required) *NX_CHAR* <=

Name of the program that made this file.

version: (required) *NX_CHAR* <=

Version of the progam that made this file.

date: (required) *NX_DATE_TIME* <=

Date the file was created.

reference: (required) *NX_CHAR*

Citation or other references for the algorithm used in the processing.

note: (optional) *NX_CHAR*

Notes required to help interpret the data, e.g. on coordinate systems.

input: (required) *NXparameters* <=

Parameters should exactly match those required by the algorithm used in the processing. For example, *azint* requires *error_model*, *mask*, *n_splitting*, *poni*, etc.

MONITOR: (optional) *NXmonitor* <=

Monitor data for example *I_zero*.

data: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nImg]) <=

DATA: (required) *NXdata* <=

Azimuthally integrated data with radial binning in q or 2theta and with azimuthal binning.

@axes: (required) *NX_CHAR* <=

Obligatory value: `['.', 'azimuthal_axis', 'radial_axis']`

@interpretation: (required) *NX_CHAR*

Obligatory value: `image`

@signal: (required) *NX_CHAR* <=

Obligatory value: `I`

I: (required) *NX_NUMBER* (Rank: 3, Dimensions: [nImg, nEta, nRad])

@long_name: (required) *NX_CHAR*

Obligatory value: `intensity`

@units: (required) *NX_CHAR*

Obligatory value: `arbitrary units`

I_errors: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [nImg, nEta, nRad]) <=

@long_name: (optional) *NX_CHAR*

Obligatory value: `estimated intensity error`

@units: (optional) *NX_CHAR*

Obligatory value: `arbitrary units`

radial_axis: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nRad]) <=

@long_name: (required) *NX_CHAR* <=

Any of these values: `q | 2theta`

@units: (required) *NX_CHAR*

Any of these values: `NX_PER_LENGHT` | `NX_WAVENUMBER` | `NX_ANGLE`

radial_axis_edges: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nRadEdge]) <=

@long_name: (required) *NX_CHAR* <=

Any of these values: `q bin edges` | `2theta bin edges`

@units: (required) *NX_CHAR*

Any of these values: `NX_PER_LENGTH` | `NX_WAVENUMBER` | `NX_ANGLE`

norm: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nEta, nRad])

Values of the normalization correction.

The normalization correction accounts for the effective number or weighted contribution of detector pixels to each integration bin.

Note: An important aspect of the normalization strategy is how polarization and solid angle corrections are incorporated, which can vary depending on the specific application, software, and its configuration options (see, for example, PyFAI documentation). Additionally, the normalization strategy may include a relative or absolute calibration factor. Two common normalization approaches are: "Relative normalization" to the PONI (Point Of Normal Incidence) pixel, and "Absolute calibration", which yields the number of photons scattered by the sample in a given direction per unit solid angle. The type of the normalization strategy is not indicated on this level. It must be concluded from the software used or its parameters.

The monitor correction is not included in the normalization correction and it is specified separately.

@long_name: (required) *NX_CHAR*

Obligatory value:

- `effective number of pixels contributing to the corresponding bin`

@units: (required) *NX_CHAR*

Obligatory value: `arbitrary units`

azimuthal_axis: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nEta])

@long_name: (required) *NX_CHAR*

Obligatory value: `azimuthal bin center`

@units: (required) *NX_CHAR*

Obligatory value: `NX_ANGLE`

azimuthal_axis_edges: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nEtaEdge])

@long_name: (required) *NX_CHAR*

Obligatory value: `azimuthal bin edges`

@units: (required) *NX_CHAR*

Obligatory value: `NX_ANGLE`

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXazint2d/ENTRY-group*
- */NXazint2d/ENTRY/DATA-group*
- */NXazint2d/ENTRY/DATA/azimuthal_axis-field*
- */NXazint2d/ENTRY/DATA/azimuthal_axis@long_name-attribute*
- */NXazint2d/ENTRY/DATA/azimuthal_axis@units-attribute*
- */NXazint2d/ENTRY/DATA/azimuthal_axis_edges-field*
- */NXazint2d/ENTRY/DATA/azimuthal_axis_edges@long_name-attribute*
- */NXazint2d/ENTRY/DATA/azimuthal_axis_edges@units-attribute*
- */NXazint2d/ENTRY/DATA/I-field*
- */NXazint2d/ENTRY/DATA/I@long_name-attribute*
- */NXazint2d/ENTRY/DATA/I@units-attribute*
- */NXazint2d/ENTRY/DATA/I_errors-field*
- */NXazint2d/ENTRY/DATA/I_errors@long_name-attribute*
- */NXazint2d/ENTRY/DATA/I_errors@units-attribute*
- */NXazint2d/ENTRY/DATA/norm-field*
- */NXazint2d/ENTRY/DATA/norm@long_name-attribute*
- */NXazint2d/ENTRY/DATA/norm@units-attribute*
- */NXazint2d/ENTRY/DATA/radial_axis-field*
- */NXazint2d/ENTRY/DATA/radial_axis@long_name-attribute*
- */NXazint2d/ENTRY/DATA/radial_axis@units-attribute*
- */NXazint2d/ENTRY/DATA/radial_axis_edges-field*
- */NXazint2d/ENTRY/DATA/radial_axis_edges@long_name-attribute*
- */NXazint2d/ENTRY/DATA/radial_axis_edges@units-attribute*
- */NXazint2d/ENTRY/DATA@axes-attribute*
- */NXazint2d/ENTRY/DATA@interpretation-attribute*
- */NXazint2d/ENTRY/DATA@signal-attribute*
- */NXazint2d/ENTRY/definition-field*
- */NXazint2d/ENTRY/INSTRUMENT-group*
- */NXazint2d/ENTRY/INSTRUMENT/MONOCHROMATOR-group*
- */NXazint2d/ENTRY/INSTRUMENT/MONOCHROMATOR/energy-field*
- */NXazint2d/ENTRY/INSTRUMENT/MONOCHROMATOR/wavelength-field*
- */NXazint2d/ENTRY/INSTRUMENT/name-field*
- */NXazint2d/ENTRY/INSTRUMENT/SOURCE-group*
- */NXazint2d/ENTRY/INSTRUMENT/SOURCE/name-field*

- */NXazint2d/ENTRY/INSTRUMENT/SOURCE/probe-field*
- */NXazint2d/ENTRY/INSTRUMENT/SOURCE/type-field*
- */NXazint2d/ENTRY/MONITOR-group*
- */NXazint2d/ENTRY/MONITOR/data-field*
- */NXazint2d/ENTRY/monitor_applied-field*
- */NXazint2d/ENTRY/normalization_applied-field*
- */NXazint2d/ENTRY/polarization_applied-field*
- */NXazint2d/ENTRY/reduction-group*
- */NXazint2d/ENTRY/reduction/date-field*
- */NXazint2d/ENTRY/reduction/input-group*
- */NXazint2d/ENTRY/reduction/note-field*
- */NXazint2d/ENTRY/reduction/program-field*
- */NXazint2d/ENTRY/reduction/reference-field*
- */NXazint2d/ENTRY/reduction/version-field*
- */NXazint2d/ENTRY/solid_angle_applied-field*
- */NXazint2d/ENTRY@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXazint2d.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 is, at least, to include a reference to how the data was acquired and processed. Yet this is not a requirement.
- There is no need for NXcanSAS to refer to any raw data.

The canSAS data format has a structure similar to NeXus, not identical. To allow canSAS data to be expressed in NeXus, yet identifiable by the canSAS standard, an additional group attribute `canSAS_class` was introduced. Here is the mapping of some common groups.

group (*)	NX_class	canSAS_class
sasentry	NXentry	SASentry
sasdata	NXdata	SASdata
sasdetector	NXdetector	SASdetector
sasinstrument	NXinstrument	SASinstrument
sasnote	NXnote	SASnote
sasprocess	NXprocess	SASprocess
sasprocessnote	NXcollection	SASprocessnote
sastransmission	NXdata	SAStransmission_spectrum
sassample	NXsample	SASsample
sassource	NXsource	SASsource

(*) The name of each group is a suggestion, not a fixed requirement and is chosen as fits each data file. See the section on defining *NXDL group and field names*.

Refer to the NeXus Coordinate System drawing (*The NeXus Coordinate System*) for choice and direction of x, y, and z axes.

The minimum required information for a NeXus data file written to the NXcanSAS specification.

```
NXcanSAS HDF5 data file
  entry : NXentry
    @NX_class = "NXentry"
    @canSAS_class = "SASentry"
    @version  = "1.0"
    definition = "NXcanSAS"
    run = "<see the documentation>"
    title = "something descriptive yet short"
    data : NXdata
      @NX_class = "NXdata"
      @canSAS_class = "SASdata"
      @signal = "I"
      @I_axes = "<see the documentation>"
      @Q_indices : NX_INT = <see the documentation>
      I : NX_NUMBER
        @units = <see the documentation>
      Q : NX_NUMBER
        @units = NX_PER_LENGTH
```

Symbols:

No symbol table

Groups cited:

NXaperture, *NXcollection*, *NXcollimator*, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXnote*, *NXprocess*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

Place the canSAS `SASentry` group as a child of a NeXus `NXentry` group (when data from multiple techniques are being stored) or as a replacement for the `NXentry` group.

Note: It is required for all numerical objects to provide a *units* attribute that describes the engineering units. Use the Unidata UDunits[1] specification as this is compatible with various community standards.

@default: (optional) *NX_CHAR <=*

Declares which *NXdata* group contains the data to be shown by default. It is needed to resolve ambiguity when more than one *NXdata* group exists. The value is the name of the default *NXdata* group. Usually, this will be the name of the first *SASdata* group.

@canSAS_class: (required) *NX_CHAR*

Official canSAS group: **SASentry**

Obligatory value: `SASentry`

@version: (required) *NX_CHAR*

Describes the version of the canSAS standard used to write this data. This must be a text (not numerical) representation. Such as:

[1] The UDunits specification also includes instructions for derived units.

```
@version="1.1"
```

Obligatory value: `1.1`

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this subentry conforms.

Obligatory value: `NXcanSAS`

title: (required) *NX_CHAR* <=

Title of this *SASentry*. Make it so that you can recognize the data by its title. Could be the name of the sample, the name for the measured data, or something else representative.

run: (required) *NX_CHAR*

Run identification for this *SASentry*. For many facilities, this is an integer, such as en experiment number. Use multiple instances of `run` as needed, keeping in mind that HDF5 requires unique names for all entities in a group.

@**name**: (optional) *NX_CHAR*

Optional string attribute to identify this particular *run*. Could use this to associate (correlate) multiple *SASdata* elements with *run* elements.

DATA: (required) *NXdata* <=

A *SASData* group contains a single reduced small-angle scattering data set that can be represented as $I(\vec{Q})$ or $I(|\vec{Q}|)$.

Q can be either a vector ($\vec{Q}$) or a vector magnitude ($|\vec{Q}|$)

The name of each *SASdata* group must be unique within a SASentry group. Suggest using names such as `sasdata01`.

NOTE: For the first *SASdata* group, be sure to write the chosen name into the *SASentry/@default* attribute, as in:

```
SASentry/@default="sasdata01"
```

A *SASdata* group has several attributes:

- I_axes
- Q_indices
- Mask_indices

To indicate the dependency relationships of other varied parameters, use attributes similar to `@Mask_indices` (such as `@Temperature_indices` or `@Pressure_indices`).

@**canSAS_class**: (required) *NX_CHAR*

Official canSAS group: NXcanSAS (applications); SASdata

Obligatory value: `SASdata`

@**signal**: (required) *NX_CHAR* <=

Name of the default data field.

Obligatory value:

- `I`: For canSAS **SASdata**, this is always "I".

@I_axes: (required) *NX_CHAR*

String array that defines the independent data fields used in the default plot for all of the dimensions of the *signal* field (the *signal* field is the field in this group that is named by the `signal` attribute of this group). One entry is provided for every dimension of the `I` data object. Such as:

```
@I_axes="Temperature", "Time", "Pressure", "Q", "Q"
```

Since there are five items in the list, the intensity field of this example `I` must be a five-dimensional array (rank=5).

@Q_indices: (required) *NX_INT* <=

Integer or integer array that describes which indices (of the I data object) are used to reference the `Q` data object. The items in this array use zero-based indexing. Such as:

```
@Q_indices=1,3,4
```

which indicates that `Q` requires three indices from the I data object: one for time and two for Q position. Thus, in this example, the `Q` data is time-dependent: $\vec{Q}(t)$.

@mask: (required) *NX_CHAR*

Name of the data mask field.

The data *mask* must have the same shape as the *data* field. Positions in the mask correspond to positions in the *data* field. The value of the mask field may be either a boolean array where `false` means *no mask* and `true` means *mask* or a more descriptive array as as defined in *NXdetector*.

@Mask_indices: (optional) *NX_CHAR*

Integer or integer array that describes which indices (of the I data object) are used to reference the `Mask` data object. The items in this array use zero-based indexing. Such as:

```
@Mask_indices=3,4
```

which indicates that Q requires two indices from the I data object for Q position.

@timestamp: (optional) *NX_DATE_TIME*

ISO-8601 time[2]

Q: (required) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Array of Q data to accompany I.

Q may be represented as either the three-dimensional scattering vector $\vec{Q}$ or the magnitude of the scattering vector, $|\vec{Q}|$.

$$|\vec{Q}| = (4\pi/\lambda)sin(\theta)$$

[2] ISO-8601 standard time representation.
NeXus dates and times are reported in ISO-8601 (e.g., `yyyy-mm-ddThh:mm:ss`) or modified ISO-8601 (e.g., `yyyy-mm-dd hh:mm:ss`).
See: http://www.w3.org/TR/NOTE-datetime or http://en.wikipedia.org/wiki/ISO_8601 for more details.

Fig. 11: The $\vec{Q}$ geometry. (`full image`)

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`

@**uncertainties**: (optional) *NX_CHAR*

(optional: for numerical arrays)

Names the dataset (in this SASdata group) that provides the uncertainty to be used for data analysis. The name of the dataset containing the Q uncertainty is flexible. The name must be unique in the *SASdata* group.

Such as:

```
@uncertainties="Q_uncertainties"
```

The *uncertainties* field will have the same *shape* (dimensions) as the Q field.

These values are the estimates of uncertainty of each Q. By default, this will be interpreted to be the estimated standard deviation. In special cases, when a standard deviation cannot possibly be used, its value can specify another measure of distribution width.

There may also be a subdirectory (optional) with constituent components.

Note

To report distribution in reported Q values, use the `@resolutions` attribute. It is possible for both `@resolutions` and `uncertainties` to be reported.

@resolutions: (optional) *NX_CHAR*

(optional: for numerical arrays)

Names the dataset (in this SASdata group) containing the Q resolution. The name of the dataset containing the Q resolution is flexible. The name must be unique in the *SASdata* group.

The *resolutions* field will have the same *shape* (dimensions) as the Q field.

Generally, this is the principal resolution of each Q. Names the data object (in this SASdata group) that provides the Q resolution to be used for data analysis. Such as:

```
@resolutions="Qdev"
```

To specify two-dimensional resolution for slit-smearing geometry, such as (*dQw*, *dQl*), use a string array, such as:

```
@resolutions="dQw", "dQl"
```

There may also be a subdirectory (optional) with constituent components.

This pattern will demonstrate how to introduce further as-yet unanticipated terms related to the data.

By default, the values of the resolutions data object are assumed to be one standard deviation of any function used to approximate the resolution function. This equates to the width of the gaussian distribution if a Gaussian is chosen. See the `@resolutions_description` attribute.

Note

To report uncertainty in reported Q values, use the `@uncertainties` attribute. It is possible for both `@resolutions` and `uncertainties` to be reported.

@resolutions_description: (optional) *NX_CHAR*

(optional) Generally, this describes the Q `@resolutions` data object. By default, the value is assumed to be "Gaussian". These are suggestions:

- Gaussian
- Lorentzian
- Square : note that the full width of the square would be ~2.9 times the standard deviation specified in the vector

- Triangular
- Sawtooth-outward : vertical edge pointing to larger Q
- Sawtooth-inward vertical edge pointing to smaller Q
- Bin : range of values contributing (for example, when 2-D detector data have been reduced to a 1-D $I(|Q|)$ dataset)

For other meanings, it may be necessary to provide further details such as the function used to assess the resolution. In such cases, use additional datasets or a *NXnote* subgroup to include that detail.

I: (required) *NX_NUMBER*

Array of intensity (I) data.

The intensity may be represented in one of these forms:

absolute units: $d\Sigma/d\Omega(Q)$ differential cross-section per unit volume per unit solid angle (such as: 1/cm/sr or 1/m/sr)

absolute units: $d\sigma/d\Omega(Q)$ differential cross-section per unit atom per unit solid angle (such as: cm^2 or m^2)

arbitrary units: $I(Q)$ usually a ratio of two detectors but units are meaningless (such as: a.u. or counts)

This presents a few problems for analysis software to sort out when reading the data. Fortunately, it is possible to analyze the *units* to determine which type of intensity is being reported and make choices at the time the file is read. But this is an area for consideration and possible improvement.

One problem arises with software that automatically converts data into some canonical units used by that software. The software should not convert units between these different types of intensity indiscriminately.

A second problem is that when arbitrary units are used, then the set of possible analytical results is restricted. With such units, no meaningful volume fraction or number density can be determined directly from $I(Q)$.

In some cases, it is possible to apply a factor to convert the arbitrary units to an absolute scale. This should be considered as a possibility of the analysis process.

Where this documentation says *typical units*, it is possible that small-angle data may be presented in other units and still be consistent with NeXus. See the *NeXus Data Units* section.

`@`**units**: (required) *NX_CHAR*

Engineering units to use when expressing I and intensity-related terms.

Data expressed in other units (or missing a `@units` attribute) will be treated as `arbitrary` by some software packages.

For software using the UDUNITS-2 library, `arbitrary` will be changed to `unknown` for handling with that library.

Any of these values:

- `1/m`: includes m2/m3 and 1/m/sr
- `1/cm`: includes cm2/cm3 and 1/cm/sr

- `m2/g`
- `cm2/g`
- `arbitrary`

@**uncertainties**: (optional) *NX_CHAR*

(optional: for numerical arrays)

Names the dataset (in this SASdata group) that provides the uncertainty of I to be used for data analysis. The name of the dataset containing the I uncertainty is flexible. The name must be unique in the *SASdata* group.

Generally, this is the estimate of the uncertainty of each I. Typically the estimated standard deviation.

Idev is the canonical name from the 1D standard. The NXcanSAS standard allows for the name to be described using this attribute. Such as:

```
@uncertainties="Idev"
```

@**scaling_factor**: (optional) *NX_CHAR*

(optional) Names the field (a.k.a. dataset) that contains a factor to multiply `I`. By default, this value is unity. Should an uncertainty be associated with the scaling factor field, the field containing that uncertainty would be designated via the `uncertainties` attribute. Such as:

```
I : NX_NUMBER
    @uncertainties="Idev" : NX_CHAR
    @scaling_factor="I_scaling" : NX_CHAR
Idev : NX_NUMBER
I_scaling : NX_NUMBER
    @uncertainties="I_scaling_dev" : NX_CHAR
I_scaling_dev : NX_NUMBER
```

The exact names for `I_scaling` and `I_scaling_dev` are not defined by NXcanSAS. The user has the flexibility to use names different than those shown in this example.

Idev: (optional) *NX_NUMBER*

Estimated **uncertainty** (usually standard deviation) in I. Must have the same units as I.

When present, the name of this field is also recorded in the *uncertainties* attribute of *I*, as in:

```
I/@uncertainties="Idev"
```

@**units**: (required) *NX_CHAR*

Engineering units to use when expressing I and intensity-related terms.

Data expressed in other units (or missing a `@units` attribute) will generate a warning from any validation process and will be treated as `arbitrary` by some analysis software packages.

For software using the UDUNITS-2 library, `arbitrary` will be changed to `unknown` for handling with that library.

Any of these values:

- `1/m`: includes m2/m3 and 1/m/sr
- `1/cm`: includes cm2/cm3 and 1/cm/sr
- `m2/g`
- `cm2/g`
- `arbitrary`

Qdev: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Estimated Q **resolution** (usually standard deviation). Must have the same units as Q.

When present, the name of this field is also recorded in the *resolutions* attribute of Q, as in:

```
Q/@resolutions="Qdev"
```

or:

```
Q/@resolutions="dQw", "dQl"
```

@units: (required) *NX_CHAR*

Engineering units to use when expressing Q and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

- `1/m`
- `1/nm`: preferred
- `1/angstrom`

dQw: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*} <=

Q **resolution** along the axis of scanning (the high-resolution *slit width* direction). Useful for defining resolution data from slit-smearing instruments such as Bonse-Hart geometry. Must have the same units as Q.

When present, the name of this field is also recorded in the *resolutions* attribute of Q, as in:

```
Q/@resolutions="dQw", "dQl"
```

@units: (required) *NX_CHAR*

Engineering units to use when expressing Q and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

- `1/m`
- `1/nm`: preferred
- `1/angstrom`

dQl: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*} <=

Q **resolution** perpendicular to the axis of scanning (the low-resolution *slit length* direction). Useful for defining resolution data from slit-smearing instruments such as Bonse-Hart geometry. Must have the same units as Q.

When present, the name of this field is also recorded in the *resolutions* attribute of Q, as in:

```
Q/@resolutions="dQw", "dQl"
```

@units: (required) *NX_CHAR*

Engineering units to use when expressing Q and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

- `1/m`
- `1/nm`: preferred
- `1/angstrom`

Qmean: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Mean value of Q for this data point. Useful when describing data that has been binned from higher-resolution data.

It is expected that `Q` is provided and that both `Q` and `Qmean` will have the same units.

@units: (required) *NX_CHAR*

Engineering units to use when expressing Q and related terms.

Data expressed in other units may not be processed by some software packages.

Any of these values:

- `1/m`
- `1/nm`: preferred
- `1/angstrom`

ShadowFactor: (optional) *NX_CHAR* {units=*NX_DIMENSIONLESS*}

A numerical factor applied to pixels affected by the beam stop penumbra. Used in data files from NIST/NCNR instruments.

See: J.G. Barker and J.S. Pedersen (1995) *J. Appl. Cryst.* **28**, 105-114.

INSTRUMENT: (optional) *NXinstrument* <=

Description of the small-angle scattering instrument.

Consider, carefully, the relevance to the SAS data analysis process when adding subgroups in this **NXinstrument** group. Additional information can be added but will likely be ignored by standardized data 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 604, 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 604, 2]

lambda: (required) *NX_NUMBER* {units=*NX_WAVELENGTH*}

Wavelength of the radiation.

This array is of the same shape as `T` and `Tdev`.

T: (required) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Transmission values (I/I_0) as a function of wavelength.

This array is of the same shape as `lambda` and `Tdev`.

@uncertainties: (required) *NX_CHAR*

Names the dataset (in this SASdata group) that provides the uncertainty of each transmission T to be used for data analysis. The name of the dataset containing the T uncertainty is expected to be `Tdev`.

Typically:

@uncertainties="Tdev"

Tdev: (required) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Estimated uncertainty (usually standard deviation) in T. Must have the same units as T.

This is the field is named in the *uncertainties* attribute of *T*, as in:

```
T/@uncertainties="Tdev"
```

This array is of the same shape as `lambda` and `T`.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcanSAS/ENTRY-group*
- */NXcanSAS/ENTRY/COLLECTION-group*
- */NXcanSAS/ENTRY/COLLECTION@canSAS_class-attribute*
- */NXcanSAS/ENTRY/DATA-group*
- */NXcanSAS/ENTRY/DATA/dQl-field*
- */NXcanSAS/ENTRY/DATA/dQl@units-attribute*
- */NXcanSAS/ENTRY/DATA/dQw-field*
- */NXcanSAS/ENTRY/DATA/dQw@units-attribute*
- */NXcanSAS/ENTRY/DATA/I-field*
- */NXcanSAS/ENTRY/DATA/I@scaling_factor-attribute*
- */NXcanSAS/ENTRY/DATA/I@uncertainties-attribute*
- */NXcanSAS/ENTRY/DATA/I@units-attribute*
- */NXcanSAS/ENTRY/DATA/Idev-field*
- */NXcanSAS/ENTRY/DATA/Idev@units-attribute*
- */NXcanSAS/ENTRY/DATA/Q-field*
- */NXcanSAS/ENTRY/DATA/Q@resolutions-attribute*
- */NXcanSAS/ENTRY/DATA/Q@resolutions_description-attribute*
- */NXcanSAS/ENTRY/DATA/Q@uncertainties-attribute*
- */NXcanSAS/ENTRY/DATA/Q@units-attribute*
- */NXcanSAS/ENTRY/DATA/Qdev-field*
- */NXcanSAS/ENTRY/DATA/Qdev@units-attribute*
- */NXcanSAS/ENTRY/DATA/Qmean-field*
- */NXcanSAS/ENTRY/DATA/Qmean@units-attribute*
- */NXcanSAS/ENTRY/DATA/ShadowFactor-field*
- */NXcanSAS/ENTRY/DATA@canSAS_class-attribute*
- */NXcanSAS/ENTRY/DATA@I_axes-attribute*

- */NXcanSAS/ENTRY/DATA@mask-attribute*
- */NXcanSAS/ENTRY/DATA@Mask_indices-attribute*
- */NXcanSAS/ENTRY/DATA@Q_indices-attribute*
- */NXcanSAS/ENTRY/DATA@signal-attribute*
- */NXcanSAS/ENTRY/DATA@timestamp-attribute*
- */NXcanSAS/ENTRY/definition-field*
- */NXcanSAS/ENTRY/INSTRUMENT-group*
- */NXcanSAS/ENTRY/INSTRUMENT/APERTURE-group*
- */NXcanSAS/ENTRY/INSTRUMENT/APERTURE/shape-field*
- */NXcanSAS/ENTRY/INSTRUMENT/APERTURE/x_gap-field*
- */NXcanSAS/ENTRY/INSTRUMENT/APERTURE/y_gap-field*
- */NXcanSAS/ENTRY/INSTRUMENT/APERTURE@canSAS_class-attribute*
- */NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR-group*
- */NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR/distance-field*
- */NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR/length-field*
- */NXcanSAS/ENTRY/INSTRUMENT/COLLIMATOR@canSAS_class-attribute*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR-group*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/beam_center_x-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/beam_center_y-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/name-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/pitch-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/roll-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/SDD-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/slit_length-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/x_pixel_size-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/x_position-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/y_pixel_size-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/y_position-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR/yaw-field*
- */NXcanSAS/ENTRY/INSTRUMENT/DETECTOR@canSAS_class-attribute*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE-group*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/beam_shape-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/beam_size_x-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/beam_size_y-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/incident_wavelength-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/incident_wavelength_spread-field*

- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/radiation-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/wavelength_max-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE/wavelength_min-field*
- */NXcanSAS/ENTRY/INSTRUMENT/SOURCE@canSAS_class-attribute*
- */NXcanSAS/ENTRY/INSTRUMENT@canSAS_class-attribute*
- */NXcanSAS/ENTRY/PROCESS-group*
- */NXcanSAS/ENTRY/PROCESS/COLLECTION-group*
- */NXcanSAS/ENTRY/PROCESS/COLLECTION@canSAS_class-attribute*
- */NXcanSAS/ENTRY/PROCESS/date-field*
- */NXcanSAS/ENTRY/PROCESS/description-field*
- */NXcanSAS/ENTRY/PROCESS/name-field*
- */NXcanSAS/ENTRY/PROCESS/NOTE-group*
- */NXcanSAS/ENTRY/PROCESS/term-field*
- */NXcanSAS/ENTRY/PROCESS@canSAS_class-attribute*
- */NXcanSAS/ENTRY/run-field*
- */NXcanSAS/ENTRY/run@name-attribute*
- */NXcanSAS/ENTRY/SAMPLE-group*
- */NXcanSAS/ENTRY/SAMPLE/details-field*
- */NXcanSAS/ENTRY/SAMPLE/name-field*
- */NXcanSAS/ENTRY/SAMPLE/pitch-field*
- */NXcanSAS/ENTRY/SAMPLE/roll-field*
- */NXcanSAS/ENTRY/SAMPLE/temperature-field*
- */NXcanSAS/ENTRY/SAMPLE/thickness-field*
- */NXcanSAS/ENTRY/SAMPLE/transmission-field*
- */NXcanSAS/ENTRY/SAMPLE/x_position-field*
- */NXcanSAS/ENTRY/SAMPLE/y_position-field*
- */NXcanSAS/ENTRY/SAMPLE/yaw-field*
- */NXcanSAS/ENTRY/SAMPLE@canSAS_class-attribute*
- */NXcanSAS/ENTRY/title-field*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM-group*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/lambda-field*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/T-field*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/T@uncertainties-attribute*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM/Tdev-field*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@canSAS_class-attribute*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@name-attribute*

- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@signal-attribute*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@T_axes-attribute*
- */NXcanSAS/ENTRY/TRANSMISSION_SPECTRUM@timestamp-attribute*
- */NXcanSAS/ENTRY@canSAS_class-attribute*
- */NXcanSAS/ENTRY@default-attribute*
- */NXcanSAS/ENTRY@version-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXcanSAS.nxdl.xml

NXdirecttof

Status:

application definition, extends *NXtofraw*

Description:

This is a application definition for raw data from a direct geometry TOF spectrometer

Symbols:

No symbol table

Groups cited:
NXdisk_chopper, *NXentry*, *NXfermi_chopper*, *NXinstrument*

Structure:

ENTRY: (required) *NXentry* <=

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXdirecttof`

INSTRUMENT: (required) *NXinstrument* <=

We 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.

- */NXdirecttof/ENTRY-group*
- */NXdirecttof/ENTRY/definition-field*
- */NXdirecttof/ENTRY/INSTRUMENT-group*
- */NXdirecttof/ENTRY/INSTRUMENT/disk_chopper-group*
- */NXdirecttof/ENTRY/INSTRUMENT/disk_chopper/energy-field*
- */NXdirecttof/ENTRY/INSTRUMENT/disk_chopper/rotation_speed-field*
- */NXdirecttof/ENTRY/INSTRUMENT/fermi_chopper-group*
- */NXdirecttof/ENTRY/INSTRUMENT/fermi_chopper/energy-field*
- */NXdirecttof/ENTRY/INSTRUMENT/fermi_chopper/rotation_speed-field*
- */NXdirecttof/ENTRY/start_time-field*
- */NXdirecttof/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXdirecttof.nxdl.xml

NXellipsometry

Status:

application definition, extends *NXoptical_spectroscopy*

Description:

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 specializes *NXoptical_spectroscopy* by extending the terms and setting specific requirements.

Information on ellipsometry is provided, e.g. in:

- H. Fujiwara, Spectroscopic ellipsometry: principles and applications, John Wiley & Sons, 2007.
- R. M. A. Azzam and N. M. Bashara, Ellipsometry and Polarized Light, North-Holland Publishing Company, 1977.
- H. G. Tompkins and E. A. Irene, Handbook of Ellipsometry, William Andrew, 2005.

Open access sources:

- https://www.angstromadvanced.com/resource.asp
- https://pypolar.readthedocs.io/en/latest/

Review articles:

- T. E. Jenkins, "Multiple-angle-of-incidence ellipsometry", J. Phys. D: Appl. Phys. 32, R45 (1999), https://doi.org/10.1088/0022-3727/32/9/201
- D. E. Aspnes, "Spectroscopic ellipsometry - Past, present, and future", Thin Solid Films 571, 334-344 (2014), https://doi.org/10.1016/j.tsf.2014.03.056
- R. M. A. Azzam, "Mueller-matrix ellipsometry: a review", Proc. SPIE 3121, Polarization: Measurement, Analysis, and Remote Sensing, (3 October 1997), https://doi.org/10.1117/12.283870
- E. A. Irene, "Applications of spectroscopic ellipsometry to microelectronics", Thin Solid Films 233, 96-111 (1993), https://doi.org/10.1016/0040-6090(93)90069-2
- S. Zollner et al., "Spectroscopic ellipsometry from 10 to 700 K", Adv. Opt. Techn., (2022), https://doi.org/10.1515/aot-2022-0016

Symbols:

Variables used throughout the document, e.g. dimensions or parameters.

N_spectrum: Length of the spectrum array (e.g. wavelength or energy) of the measured data.

N_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.

Groups cited:
NXdata, *NXentry*, *NXfabrication*, *NXinstrument*, *NXoptical_lens*, *NXprogram*, *NXsample*, *NXwaveplate*

Structure:

ENTRY: (required) *NXentry* <=

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.

title: (recommended) *NX_CHAR* <=

experiment_type: (required) *NX_CHAR* <=

Specify the type of the optical experiment.

You may specify fundamental characteristics or properties in the experimental subtype.

Obligatory value: `ellipsometry`

ellipsometry_experiment_type: (required) *NX_CHAR*

Specify the type of ellipsometry.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `in situ spectroscopic ellipsometry`
- `THz spectroscopic ellipsometry`
- `infrared spectroscopic ellipsometry`
- `ultraviolet spectroscopic ellipsometry`
- `uv-vis spectroscopic ellipsometry`
- `NIR-Vis-UV spectroscopic ellipsometry`

INSTRUMENT: (required) *NXinstrument* <=

Properties of the ellipsometry equipment.

ellipsometer_type: (required) *NX_CHAR*

What type of ellipsometry was used? See Fujiwara Table 4.2.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `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`

focusing_probes: (optional) *NXoptical_lens* <=

If focusing probes (lenses) were used, please state if the data were corrected for the window effects.

type: (required) *NX_CHAR* <=

Any of these values or a custom value (if you use a custom value, also set @custom=True): `objective` | `lens` | `glass fiber` | `none`

data_correction: (recommended) *NX_BOOLEAN*

Were the recorded data corrected by the window effects of the focusing probes (lenses)?

angular_spread: (recommended) *NX_NUMBER* {units=*NX_ANGLE*}

Specify the angular spread caused by the focusing probes.

device_information: (optional) *NXfabrication* <=

rotating_element: (required) *NXwaveplate* <=

Properties of the rotating element defined in 'instrument/rotating_element_type'.

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)`

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.

SAMPLE: (required) *NXsample* <=

backside_roughness: (optional) *NX_BOOLEAN*

Was the backside of the sample roughened? Relevant for infrared ellipsometry.

data_collection: (optional) *NXdata* <=

Measured data, data errors, and varied parameters. This may be used to describe indirectly derived data or data transformed between different descriptions, such as: Raw Data –> Psi Delta Psi, Delta –> N,C,S Mueller matrix –> N,C,S Mueller matrix –> Psi, Delta etc.

Other types of data, such as temperature or sample location, may be saved in a generic (NXdata) concept from *NXoptical_spectroscopy*, or better directly in the location of the sample positioner or temperature sensor.

data_identifier: (recommended) *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: (recommended) *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 'NAME' 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*

External link to the data field in 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.

Ideally, the link uses the relative path with respect to the actual NeXus file.

data_software: (optional) *NXprogram*

@URL: (optional) *NX_CHAR*

Website of the software.

program: (recommended) *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: (recommended) *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.

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_identifier-field*
- */NXellipsometry/ENTRY/data_collection/data_software-group*
- */NXellipsometry/ENTRY/data_collection/data_software/program-field*
- */NXellipsometry/ENTRY/data_collection/data_software/version-field*
- */NXellipsometry/ENTRY/data_collection/data_software@URL-attribute*
- */NXellipsometry/ENTRY/data_collection/data_type-field*
- */NXellipsometry/ENTRY/data_collection/measured_data-field*
- */NXellipsometry/ENTRY/data_collection/measured_data@units-attribute*
- */NXellipsometry/ENTRY/data_collection/measured_data_errors-field*
- */NXellipsometry/ENTRY/data_collection/measured_data_errors@units-attribute*

- */NXellipsometry/ENTRY/data_collection/NAME_spectrum-field*
- */NXellipsometry/ENTRY/data_collection/NAME_spectrum@units-attribute*
- */NXellipsometry/ENTRY/data_collection/reference_data_link-field*
- */NXellipsometry/ENTRY/data_collection/varied_parameter_link-field*
- */NXellipsometry/ENTRY/definition-field*
- */NXellipsometry/ENTRY/definition@URL-attribute*
- */NXellipsometry/ENTRY/definition@version-attribute*
- */NXellipsometry/ENTRY/ellipsometry_experiment_type-field*
- */NXellipsometry/ENTRY/experiment_type-field*
- */NXellipsometry/ENTRY/INSTRUMENT-group*
- */NXellipsometry/ENTRY/INSTRUMENT/ellipsometer_type-field*
- */NXellipsometry/ENTRY/INSTRUMENT/focusing_probes-group*
- */NXellipsometry/ENTRY/INSTRUMENT/focusing_probes/angular_spread-field*
- */NXellipsometry/ENTRY/INSTRUMENT/focusing_probes/data_correction-field*
- */NXellipsometry/ENTRY/INSTRUMENT/focusing_probes/device_information-group*
- */NXellipsometry/ENTRY/INSTRUMENT/focusing_probes/type-field*
- */NXellipsometry/ENTRY/INSTRUMENT/rotating_element-group*
- */NXellipsometry/ENTRY/INSTRUMENT/rotating_element/fixed_revolutions-field*
- */NXellipsometry/ENTRY/INSTRUMENT/rotating_element/max_revolutions-field*
- */NXellipsometry/ENTRY/INSTRUMENT/rotating_element/revolutions-field*
- */NXellipsometry/ENTRY/INSTRUMENT/rotating_element/rotating_element_type-field*
- */NXellipsometry/ENTRY/SAMPLE-group*
- */NXellipsometry/ENTRY/SAMPLE/backside_roughness-field*
- */NXellipsometry/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXellipsometry.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, to simulate this, to explore physical mechanisms and phenomena, or to characterize materials.

The NeXus application definition NXem defines a hierarchical data model with ten building blocks:

The data model represents a tree of concepts. The tree is constructed from groups of concepts representing the branches surplus fields and attributes representing leafs.

NXem an introduction for typical use cases in material characterization and simulation:

Transmission electron microscopy (TEM) and Scanning Transmission Electron Microscopy (STEM) Scanning Electron Microscopy (SEM) Scanning Electron Microscopy combined a Focused-Ion Beam (SEM/FIB)

A deeper dive into the branches of NXem:

NXem is constructed from composing and specializing base classes into the following ten categories:

- The field `definition` specifies that the data schema is NXem. In combination with administrative metadata such as the `NeXus_version` provided by *NXroot* this specifies which version of NXem the instance data in a NeXus/HDF5 file are compliant with.
- The fields `identifier_experiment`, `experiment_alias`, `experiment_description` and the group `userID` provide concepts for storing organizational metadata that contextualize the work within the research workflow and humans involved in this.
- The fields `start_time`, `end_time` provide concepts for framing a temporal context for the research.
- The groups `citeID`, `noteID` provide concepts for adding contextual details such as citations that are associated with or notes, i.e. other artifacts that 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 when a simulation was performed.
- The group `sampleID` provides concepts for storing metadata about the sample that was characterized or simulated during the session.
- The group `measurement` provides concepts that are useful for describing a measurement during a session with an electron microscope. This includes the chain of events of data and metadata that were collected during such a session.
- The group``simulation`` provides concepts that are useful for describing a simulation of an electron beam that interacts with matter. Combined with `measurement` this provides a data schema for defining a digital twin of the microscope and its optical setup.
- The groups `consistent_rotations`, `NAMED_reference_frame` provide concepts for reporting coordinate systems (frames of reference) and rotation conventions that clarify how data should be interpreted specifying the rotation of orientable objects in the microscope, its components, or of crystals and crystal defects in the material analyzed. These metadata support interpretation for downstream or on-the-fly data analyses which electron microscopes typically nowadays perform during a session. Examples are the indexing of diffraction patterns, image analysis in general, or analyses of the chemical composition.
- The group `roiID` provides concepts for reporting several domain- and technique-specific configuration parameter and results of data processing steps that were applied.
- The group `profiling` provides concepts for reporting computational details such as programs and libraries used, for documenting the libraries of virtual environments such as those used by conda or python virtual environment, including details about the computing hardware used, and documenting capabilities for performance analyses and benchmarking of the software or its parts.

Design choices:

Specific details about how an electron microscope was used and eventually its configuration modified differ between user groups. This holds also true for computer simulations of electron-beam matter interaction.

Different peer groups in different sub-domains in electron microscopy consider different data and metadata relevant. NXem defines constraints on the existence and cardinality of concepts and its concept branches but seeks to offer a compromise. The key design pattern followed is that most branches are made optional or at most recommended but their child concepts conditional required. Thereby, NXem can cover a variety of simple but also complex use cases. An example of this parent-optional-but-children-stronger-restricted design is the combination of the optional group `measurement` with its required child `measurement/instrument`: Users which report simulations are not forced to document the instrument but users which have characterized a sample are motivated to report about the instrument. They are though not necessarily required to report all the details of the instruments' components because the design pattern is-used applied recursively.

Inclusive design, one schema for scanning, focused-ion beam, and transmission electron microscopes:

Contrary to many other proposals of a data schema for electron microscopy, NXem seeks to highlight the similarity of the three fundamental types of electron microscopes that are nowadays used most routinely in academia and industry: An electron microscope is a beamline that provides a controlled beam of electrons combined with eventually beams of other particles (ions) to investigate electron/ion(-beam) matter interaction. This design of per-particle-type concept branches is realized in the base classes `NXebeam_column` and `NXibeam_column`. These provide concepts for reporting the technical components that are typically used for generating a controllable (and typically scanning) beam of particles such as electrons or ions.

Focused-ion beam capabilities are modelled by adding an optional group `measurement/instrument/ibeam_column`. We foresee that this design is beneficial also in the future when research should be documented where photon-electron interactions via an electron microscope are combined. The current proposal though does not include such a `NXpbeam_column` base class that could be used for 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 the electron-optical setup and the components used. What remains the same from the perspective of an observer who monitors the experiment inside the electron-matter interaction volume, i.e. in, on, or close to the surface of the specimen is the imaginary split into an upper and a lower half-space. In the upper half-space a specific but eventually differently shaped electron beam illuminates the specimen when comparing a scanning with a transmission electron microscope. In the lower half-space the beam or particles exit the specimen or end up thermalized in thick specimens.

NXem distinguishes and stores instance data based on how long they remain unchanged:

`measurement` provides two groups `measurement/instrument` and `measurement/eventID`. The first group is designed for storing metadata about the instrument which do not change over the course of the session. Examples are the name of the technology partner who built the microscope, the microscope's serial number, or the type of lenses mounted, etc. The second group is designed for metadata and data that are collected during the microscope session. For these, specializations of `NXdata` specifically `NXimage` and `NXspectrum` are provided. Each `measurement/eventID` event can be time-stamped individually. Each instance of a group `measurement/eventID` contains `measurement/instrument` whose purpose is to store those specific state and settings of the microscope that was present during the collection of the event. This includes lens settings, apertures used, aberrations, and other components, etc. By virtue of design this reduces unnecessary repetition of metadata stored in the first group like is often observed in image-based 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 provide 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 and breath how standardization can be useful even to report specifically formatted data representations like multi-dimensional plotting. Thereby, downstream process-

ing using software for data analyses or research data management can take advantage of a standardized reporting rather than demanding for a zoo of parsers that interconvert between many representations.

*NXem within the ecosystem of data schemata for electron microscopy:**

We support the statement that substantially fewer standardized rather than many ad hoc schemata are required to facilitate the documentation and exchange of knowledge within electron microscopy. We tailored NXem to serve the materials science and materials engineering usage of electron microscopy to provide a complementary coverage to what OMERO has established for the bio- and life science usage of electron microscopy.

Symbols:

No symbol table

Groups cited:

NXaberration, *NXactuator*, *NXaperture*, *NXatom*, *NXbeam*, *NXcite*, *NXcollection*, *NXcomponent*, *NXcoordinate_system*, *NXcorrector_cs*, *NXcs_profiling*, *NXdata*, *NXdeflector*, *NXdetector*, *NXebeam_column*, *NXelectromagnetic_lens*, *NXem_ebsd*, *NXem_eds*, *NXem_eels*, *NXem_event_data*, *NXem_img*, *NXem_instrument*, *NXem_interaction_volume*, *NXem_measurement*, *NXem_optical_system*, *NXem_simulation*, *NXentry*, *NXfabrication*, *NXibeam_column*, *NXimage*, *NXmanipulator*, *NXmonochromator*, *NXnote*, *NXparameters*, *NXphase*, *NXprocess*, *NXprogram*, *NXpump*, *NXroi_process*, *NXsample*, *NXscan_controller*, *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 *NXem_event_data* 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*

author: (optional) *NX_CHAR*

The author(s) of that reference.

doi: (required) *NX_CHAR* <=

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 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 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 illuminable 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 *NXem_event_data* 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`
- `xzx`
- `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`

NAMED_**reference_frame**ID: (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) *NXem_instrument* <=

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) *NXelectromagnetic_lens* <=

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* <=

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) *NXscan_controller* <=

fabrication: (recommended) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

ibeam_column: (optional) *NXibeam_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) *NXatom* <=

fabrication: (optional) *NXfabrication* <=

vendor: (required) *NX_CHAR* <=

model: (required) *NX_CHAR* <=

serial_number: (recommended) *NX_CHAR* <=

lensID: (optional) *NXelectromagnetic_lens* <=

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) *NXscan_controller* <=

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: (recommended) *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) *NXem_event_data* <=

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*

intensity: (optional) *NX_NUMBER* <=

@**long_name**: (required) *NX_CHAR*

complex: (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*

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

magnitude: (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*

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

complex: (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* <=

image_4d: (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*

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

complex: (optional) *NX_COMPLEX* <=

@long_name: (required) *NX_CHAR*

axis_m: (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* <=

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*

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

complex: (optional) *NX_COMPLEX* <=

@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*

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

complex: (optional) *NX_COMPLEX* <=

@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*

intensity: (optional) *NX_NUMBER* <=

@long_name: (required) *NX_CHAR*

complex: (optional) *NX_COMPLEX* <=

@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) *NXem_instrument* <=

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) *NXelectromagnetic_lens* <=

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* <=

scan_controller: (recommended) *NXscan_controller* <=

scan_schema: (required) *NX_CHAR* <=

dwell_time: (required) *NX_NUMBER* <=

ibeam_column: (optional) *NXibeam_column* <=

operation_mode: (recommended) *NX_CHAR* <=

ion_source: (required) *NXsource* <=

voltage: (required) *NX_NUMBER* <=

flux: (required) *NX_NUMBER* <=

probe: (required) *NXatom* <=

lensID: (optional) *NXelectromagnetic_lens* <=

power_setting: (required) *NX_CHAR_OR_NUMBER* <=

apertureID: (optional) *NXaperture* <=

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* <=

blankerID: (optional) *NXdeflector* <=

monochromatorID: (optional) *NXmonochromator* <=

applied: (required) *NX_BOOLEAN* <=

sensorID: (optional) *NXsensor* <=

actuatorID: (optional) *NXactuator* <=

beamID: (optional) *NXbeam* <=

scan_controller: (recommended) *NXscan_controller* <=

scan_schema: (required) *NX_CHAR* <=

dwell_time: (required) *NX_NUMBER* <=

optics: (recommended) *NXem_optical_system* <=

detectorID: (optional) *NXdetector* <=

operation_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) *NXem_interaction_volume*

DATA: (recommended) *NXdata* <=

PROCESS: (recommended) *NXprocess* <=

roiID: (optional) *NXroi_process*

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* <=

@**long_name**: (required) *NX_CHAR*

axis_i: (required) *NX_NUMBER* <=

@**long_name**: (required) *NX_CHAR* <=

axis_j: (required) *NX_NUMBER* <=

@**long_name**: (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/citeID/author-field*
- */NXem/ENTRY/citeID/doi-field*
- */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/complex-field*
- */NXem/ENTRY/measurement/eventID/imageID/image_1d/complex@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/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/complex-field
- /NXem/ENTRY/measurement/eventID/imageID/image_3d/complex@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/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/image_4d-group
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_i-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_i@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_j-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_j@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_k-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_k@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_m-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/axis_m@long_name-attribute

- /NXem/ENTRY/measurement/eventID/imageID/image_4d/complex-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/complex@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/imag-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/imag@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/intensity-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/intensity@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/real-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/real@long_name-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d/title-field
- /NXem/ENTRY/measurement/eventID/imageID/image_4d@axes-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d@AXISNAME_indices-attribute
- /NXem/ENTRY/measurement/eventID/imageID/image_4d@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/complex-field
- /NXem/ENTRY/measurement/eventID/imageID/stack_1d/complex@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/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/complex-field
- /NXem/ENTRY/measurement/eventID/imageID/stack_2d/complex@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/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/complex-field
- /NXem/ENTRY/measurement/eventID/imageID/stack_3d/complex@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/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/operation_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/NAMED_reference_frameID-group
- /NXem/ENTRY/NAMED_reference_frameID/alias-field
- /NXem/ENTRY/NAMED_reference_frameID/origin-field
- /NXem/ENTRY/NAMED_reference_frameID/type-field
- /NXem/ENTRY/NAMED_reference_frameID/x-field
- /NXem/ENTRY/NAMED_reference_frameID/x_direction-field

- */NXem/ENTRY/NAMED_reference_frameID/y-field*
- */NXem/ENTRY/NAMED_reference_frameID/y_direction-field*
- */NXem/ENTRY/NAMED_reference_frameID/z-field*
- */NXem/ENTRY/NAMED_reference_frameID/z_direction-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_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/intensity-field
- /NXem/ENTRY/roiID/eds/indexing/ELEMENT_SPECIFIC_MAP/image_2d/intensity@long_name-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*

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

NXmpes

Status:

application definition, extends *NXobject*

Description:

This is the most general application definition for photoemission experiments.

Groups and fields are named according to the ISO 18115-1:2023 specification as well as the IUPAC Recommendations 2020.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays

n_transmission_function: Number of data points in the transmission function.

Groups cited:

NXactivity, *NXactuator*, *NXaperture*, *NXbeam*, *NXcalibration*, *NXcollectioncolumn*, *NXcoordinate_system*, *NXdata*, *NXdistortion*, *NXelectron_detector*, *NXelectronanalyzer*, *NXenergydispersion*, *NXentry*, *NXenvironment*, *NXfabrication*, *NXfit*, *NXhistory*, *NXinsertion_device*, *NXinstrument*, *NXlog*, *NXmanipulator*, *NXmonochromator*, *NXpid_controller*, *NXregistration*, *NXresolution*, *NXsample*, *NXsensor*, *NXsource*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXmpes`

@version: (required) *NX_CHAR* <=

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

Datetime of the start of the measurement. Should be an ISO8601 date/time stamp. It is recommended to add an explicit time zone, otherwise the local time zone is assumed per ISO8601.

end_time: (recommended) *NX_DATE_TIME* <=

Datetime of the end of the measurement. Should be a ISO8601 date/time stamp. It is recommended to add an explicit time zone, otherwise the local time zone is assumed per ISO8601.

method: (recommended) *NX_CHAR*

Name of the experimental method.

If applicable, this name should match the terms given by Clause 11 of the ISO 18115-1:2023 specification.

Examples include:

- X-ray photoelectron spectroscopy (XPS)
- angle-resolved X-ray photoelectron spectroscopy (ARXPS)
- ultraviolet photoelectron spectroscopy (UPS)
- angle-resolved photoelectron spectroscopy (ARPES)
- hard X-ray photoemission spectroscopy (HAXPES)

- near ambient pressure X-ray photoelectron spectroscopy (NAPXPS)
- photoelectron emission microscopy (PEEM)
- electron spectroscopy for chemical analysis (ESCA)
- time-resolved angle-resolved X-ray photoelectron spectroscopy (trARPES)
- spin-resolved angle-resolved X-ray photoelectron spectroscopy (spin-ARPES)
- momentum microscopy

transitions: (optional) *NX_CHAR*

Array of strings representing the electronic core levels and Auger transitions probed in this MPES experiment.

In order for experiments to be comparable, the notation must follow a strict convention.

For core levels:

- The element symbol (chemical symbol) is written first.
- It is followed by a whitespace and then the electronic level (e.g., "1s", "2p", "3d", etc.)
- Fine-structure splitting levels must include the total angular momentum quantum number **J**, written as a fraction after the orbital label (e.g., "3d5/2", "4f7/2").
- When relevant, fine-structure levels should be specified. If multiple fine-structure levels are probed, they should either be given explicitly or the generic level (e.g., "3d", "4f") can be used.

Examples of correct core level notation:

- "C 1s"
- "O 1s"
- "Fe 2p"
- "Fe 2p3/2"
- "Fe 2p1/2"
- "Au 4f"
- "Au 4f5/2"
- "Au 4f7/2"

For Auger transitions:

- The element symbol (chemical symbol) is written first.
- It is followed by a whitespace and the Auger transitions, which can include:
  - Explicit transitions (e.g., "KLL", "LMM") without fine-structure splitting
  - Explicit transitions (e.g., "KL1L2", "LM1M2") with fine-structure splitting
  - Simplified valence notation (e.g., "KVV", "KLV").
  - Combinations of the above (e.g. "KL1V").

Examples of correct Auger transition notation:

- "C KLL"
- "O KLL"

- "O KVV"
- "O KL1L2"

Additional Allowed Entries:

Besides specific core levels and Auger transitions, the following broader spectral regions can also be listed:

- "Fermi Edge"
- "Valence Band"
- "Survey"

Incorrect Notation Examples (Do Not Use):

- "C1s" (missing space)
- "O-1s" (incorrect separator)
- "Fe2p" (missing space)
- "Au4f7/2" (missing space)
- "O-KVV" (incorrect separator)
- "Fe 2p_3/2" (incorrect underscore)
- "Fe 2p 3/2" (extra space between "p" and "3/2")

COORDINATE_SYSTEM: (optional) *NXcoordinate_system*

Description of one coordinate systems that are specific to the setup and the measurement geometry.

Multiple coordinate systems can be used if necessary.

USER: (recommended) *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 time when the experiment was performed.

INSTRUMENT: (required) *NXinstrument* <=

Description of the photoemission spectrometer and its individual parts.

This concept is related to term 12.58 of the ISO 18115-1:2023 standard. **energy_resolution**: (optional) *NXresolution*

Overall energy resolution of the photoemission instrument.

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `energy`

type: (recommended) *NX_CHAR* <=

resolution: (required) *NX_FLOAT* {units=*NX_ENERGY*} <=

Minimum distinguishable energy separation in the energy spectra.

This concept is related to term 10.24 of the ISO 18115-1:2023 standard.

relative_resolution: (optional) *NX_FLOAT* <=

Ratio of the energy resolution of the photoemission spectrometer at a specified energy value to that energy value.

This concept is related to term 10.7 ff. of the ISO 18115-1:2023 standard.

RESOLUTION: (optional) *NXresolution*

Any further resolution information about the instrument. For example, the angular resolution of the instrument if the spectrometer is angle-resolving.

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

source_probe: (recommended) *NXsource* <=

The source used to generate the *beam_probe*.

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 or similar.

type: (required) *NX_CHAR* <=

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `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`
- `UV lamp`
- `Monochromatized electron source`

name: (recommended) *NX_CHAR* <=

probe: (optional) *NX_CHAR* <=

associated_beam: (required) *NX_CHAR*

A reference to a beam emitted by this source. Should be named with the same suffix, e.g., for `source_probe` it should refer to `beam_probe`.

Example:

- /entry/instrument/source_probe/associated_beam = /entry/instrument/beam_probe

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

source_pump: (optional) *NXsource* <=

The source used to generate the *beam_pump* in pump-probe experiments.

Properties refer strictly to parameters of the source, not of the output beam.

type: (required) *NX_CHAR* <=

name: (recommended) *NX_CHAR* <=

probe: (optional) *NX_CHAR* <=

associated_beam: (required) *NX_CHAR*

A reference to a beam emitted by this source. Should be named with the same suffix, e.g., for `source_pump` it should refer to `beam_pump`.

Example:

- /entry/instrument/source_pump/associated_beam = /entry/instrument/beam_pump

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

source_TYPE: (optional) *NXsource* <=

Any other source used to generate a beam.

This group is to be used for any additional beams that are not described by *source_probe* or *source_pump*.

Examples could be a low energy electron source for charge neutralization (see also *flood_gun*) or an additional laser source.

Properties refer strictly to parameters of the source, not of the output beam.

Note that the uppercase notation in `source_TYPE` means that multiple sources can be provided. The uppercase part can be substituted with any string that consists of alphanumeric characters, including both uppercase and lowercase letters from A to Z and numerical digits from 0 to 9. For example, in pump-probe experiments, it is possible to have both a `source_laser` and a `source_electron`.

type: (required) *NX_CHAR* <=

name: (recommended) *NX_CHAR* <=

probe: (optional) *NX_CHAR* <=

associated_beam: (required) *NX_CHAR*

A reference to a beam emitted by this source. Should be named with the same suffix, e.g., for `source_laser` it should refer to `beam_laser`.

Example:

- /entry/instrument/source_laser/associated_beam = /entry/instrument/beam_laser

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

beam_probe: (required) *NXbeam* <=

Properties of the probe beam at a given location.

This is the beam that is used to facilitate the photoemission during MPES experiments.

distance: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

Distance between the point where the current NXbeam instance is evaluating the beam properties and the point where the beam interacts with the sample. For photoemission, the latter is the point where the the centre of the beam touches 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*} <=

extent: (recommended) *NX_FLOAT* <=

associated_source: (recommended) *NX_CHAR*

The source that emitted this beam. Should be named with the same suffix, e.g., for `beam_probe` it should refer to `source_probe`. This should be specified if an associated source exists.

Example:

- /entry/instrument/beam_probe/associated_source = /entry/instrument/source_probe

beam_pump: (optional) *NXbeam* <=

Properties of the pump beam at a given location.

In pump-probe experiments, this is the beam that excites the system, initiating a change in its state. It sets the timing for the experiment by defining time zero in a pump-probe setup.

distance: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

Distance between the point where the current NXbeam instance is evaluating the beam properties and the point where the beam interacts with

the sample. For photoemission, the latter is the point where the the centre of the beam touches 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*} <=

extent: (recommended) *NX_FLOAT* <=

associated_source: (recommended) *NX_CHAR*

The source that emitted this beam. Should be named with the same suffix, e.g., for `beam_pump` it should refer to `source_pump`. This should be specified if an associated source exists.

Example:

- /entry/instrument/beam_pump/associated_source = /entry/instrument/source_pump

beam_TYPE: (optional) *NXbeam* <=

Properties of any other beam at a given location.

This group is to be used for any additional beams that are not described by *beam_probe* or *beam_pump*.

Should be named with the same suffix as `source_TYPE`, e.g., for `source_laser` it should refer to `beam_laser`.

distance: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

Distance between the point where the current NXbeam instance is evaluating the beam properties and the point where the beam interacts with the sample. For photoemission, the latter is the point where the the centre of the beam touches 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*} <=

extent: (recommended) *NX_FLOAT* <=

associated_source: (recommended) *NX_CHAR*

The source that emitted this beam. Should be named with the same suffix, e.g., for `beam_laser` it should refer to `source_laser`. This should be specified if an associated source exists.

Example:

- /entry/instrument/beam_laser/associated_source = /entry/instrument/source_laser

ELECTRONANALYZER: (required) *NXelectronanalyzer*

description: (recommended) *NX_CHAR* <=

work_function: (recommended) *NX_FLOAT* {units=*NX_ENERGY*} <=

fast_axes: (recommended) *NX_CHAR* <=

slow_axes: (recommended) *NX_CHAR* <=

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

energy_resolution: (optional) *NXresolution* <=

type: (recommended) *NX_CHAR* <=

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `energy`

resolution: (required) *NX_FLOAT* <=

transmission_function: (optional) *NXdata* <=

COLLECTIONCOLUMN: (required) *NXcollectioncolumn* <=

scheme: (required) *NX_CHAR* <=

Scheme of the electron collection column.

Any of these values:

- `angular dispersive`
- `spatial dispersive`
- `momentum dispersive`
- `non-dispersive`

lens_mode: (recommended) *NX_CHAR* <=

projection: (recommended) *NX_CHAR* <=

angular_acceptance: (optional) *NX_FLOAT* <=

spatial_acceptance: (optional) *NX_FLOAT* <=

field_aperture: (optional) *NXaperture* <=

The size and position of the field aperture inserted in the column. To add additional or other apertures use the APERTURE group of NXcollectioncolumn.

contrast_aperture: (optional) *NXaperture* <=

The size and position of the contrast aperture inserted in the column. To add additional or other apertures use the APERTURE group of NXcollectioncolumn.

iris: (optional) *NXaperture* <=

Size, position and shape of the iris inserted in the column.

The iris is an aperture in the lens with a variable diameter which can reduce the number of electrons entering the analyzer.

To add additional or other slits use the APERTURE group of NXcollectioncolumn.

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

ENERGYDISPERSION: (required) *NXenergydispersion* <=

scheme: (required) *NX_CHAR* <=

Any of these values:

- `tof`
- `hemispherical`
- `double hemispherical`
- `cylindrical mirror`
- `display mirror`
- `retarding grid`

pass_energy: (recommended) *NX_FLOAT* {units=*NX_ENERGY*} <=

Only one of `pass_energy` or `drift_energy` should be supplied. `pass_energy` should be used when working with hemispherical analyzers.

drift_energy: (recommended) *NX_FLOAT* {units=*NX_ENERGY*} <=

Only one of `pass_energy` or `drift_energy` should be supplied. `drift_energy` should be used if a TOF is used in the energy dispersive part of the electron analyzer.

energy_scan_mode: (recommended) *NX_CHAR* <=

entrance_slit: (optional) *NXaperture* <=

Size, position and shape of the entrance slit in dispersive analyzers.

To add additional or other slits use the APERTURE group of NXenergydispersion.

exit_slit: (optional) *NXaperture* <=

Size, position and shape of the exit slit in dispersive analyzers.

To add additional or other slits use the APERTURE group of NXenergydispersion.

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

ELECTRON_DETECTOR: (required) *NXelectron_detector*

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`

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

raw_data: (recommended) *NXdata* <=

Contains the raw data collected by the detector before calibration. The data which is considered raw might change from experiment to experiment due to hardware pre-processing of the data. This group ideally collects the data with the lowest level of processing possible.

Axes should be named according to the conventions defined below. Note that this list is a glossary with explicitly named axis names, which is only intended to cover the most common measurement axes and is therefore not complete. It is possible to add axes with other names at any time.

@signal: (required) *NX_CHAR* <=

Obligatory value: `raw`

raw: (required) *NX_NUMBER* <=

Raw data before calibration.

pixel_x: (optional) *NX_POSINT*

Detector pixel number in x direction.

pixel_y: (optional) *NX_POSINT*

Detector pixel number in y direction.

energy: (recommended) *NX_NUMBER* {units=*NX_ENERGY*}

(Un)calibrated energy axis.

@type: (required) *NX_CHAR*

The energy can be either stored as kinetic or as binding energy.

Any of these values:

- `kinetic`: (Un)calibrated kinetic energy axis. This concept is related to term 3.35 of the ISO 18115-1:2023 standard.
- `binding`: (Un)calibrated binding energy axis. This concept is related to term 12.16 of the ISO 18115-1:2023 standard.

photon_energy: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

(Un)calibrated photon energy of the incoming probe beam.

Could be a link to /entry/instrument/beam_probe/incident_energy.

kx: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

(Un)calibrated k-space coordinate in x direction. It is envisioned that the axes in momentum space are named `kx`, `ky`, and `kz`. Typically, the vectors in momentum space are defined such that `kx` and `ky` comprise the parallel component, while `kz` is the perpendicular component.

It is also possible to define `k_parallel` and `k_perpendicular` for the parallel and perpendicular momenta, respectively.

Units are typically 1/angstrom.

ky: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

(Un)calibrated k-space coordinate in y direction. For more information, see the definition of the *kx* axis.

kz: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

(Un)calibrated k-space coordinate in z direction. For more information, see the definition of the *kx* axis.

k_parallel: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

(Un)calibrated parallel component in k-space.

`k_parallel` and *k_perpendicular* describe how the electron's wave vector `k` is split into components relative to the surface.

`k_parallel` is the component of the electron's wave vector that is parallel to the surface. It is conserved during the photoemission process. This means that the electron's momentum along the surface inside the material is directly related to its measured momentum outside the material.

Units are typically 1/angstrom.

k_perpendicular: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

(Un)calibrated perpendicular component in k-space.

`k_perpendicular` is the component that is normal (perpendicular) to the surface. It is not conserved during photoemission because the electron experiences a potential change

when it exits the material into vacuum. To determine `k_perpendicular` inside the material, one typically needs to estimate the inner potential ϕ_0, which accounts for the energy shift due to the material's work function and electronic structure.

Units are typically 1/angstrom.

angular0: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

First (un)calibrated angular coordinate. It is envisioned that the axes in angular space are named `angular0` and `angular1`.

The angular axes should be named in order of decreasing speed, i.e., `angular0` should be the fastest scan axis and `angular1` should be the slow-axis angular coordinate. However, `angular0` may also be second slowest axis if the measurement is angularly integrated and `angular1` could also be the second fastest axis in the case of simultaneous dispersion in two angular dimensions.

angular1: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Second (un)calibrated angular coordinate.

For more information, see the definition of the *angular0* axis.

This is typically the slower scan axis compared to `angular0`.

spatial0: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

First (un)calibrated spatial coordinate. It is envisioned that the axes in regular space are named `spatial0` and `spatial1`.

The spatial axes should be named in order of decreasing speed, i.e., `spatial0` should be the fastest scan axis and *spatial1* ` should be the slow-axis spatial coordinate. However, `spatial` may also be second slow axis if the measurement is spatially integrated and `spatial1` could also be the second fast axis in the case of simultaneous dispersion in two spatial dimensions.

spatial1: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Second (un)calibrated spatial coordinate.

For more information, see the definition of the *spatial0* axis.

This is typically the slower scan axis compared to `spatial0`.

delay: (optional) *NX_NUMBER* {units=*NX_TIME*} <=

(Un)calibrated delay time. This is to be used for time-resolved pump-probe experiments and describes the delay between *beam_pump* and *beam_probe*.

temperature: (optional) *NX_NUMBER* {units=*NX_TIME*}

(Un)calibrated temperature axis in case of experiments where the temperature was scanned. This is typically the sample temperature and could be linked from /entry/sample/temperature_env/temperature_sensor/value.

MANIPULATOR: (optional) *NXmanipulator*

Manipulator for positioning of the sample.

temperature_sensor: (recommended) *NXsensor* <=

name: (recommended) *NX_CHAR* <=

measurement: (required) *NX_CHAR* <=

Obligatory value: `temperature`

type: (optional) *NX_CHAR* <=

value: (required) *NX_FLOAT* <=

sample_heater: (optional) *NXactuator* <=

name: (recommended) *NX_CHAR* <=

actuation_target: (required) *NX_CHAR* <=

Obligatory value: `temperature`

type: (optional) *NX_CHAR* <=

output_heater_power: (recommended) *NX_FLOAT* <=

PID_CONTROLLER: (recommended) *NXpid_controller* <=

setpoint: (recommended) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

cryostat: (optional) *NXactuator* <=

name: (recommended) *NX_CHAR* <=

actuation_target: (required) *NX_CHAR* <=

Obligatory value: `temperature`

type: (optional) *NX_CHAR* <=

PID_CONTROLLER: (required) *NXpid_controller* <=

setpoint: (recommended) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

drain_current_ammeter: (optional) *NXsensor* <=

name: (recommended) *NX_CHAR* <=

measurement: (required) *NX_CHAR* <=

Obligatory value: `current`

type: (optional) *NX_CHAR* <=

value: (required) *NX_FLOAT* <=

sample_bias_voltmeter: (recommended) *NXsensor* <=

name: (recommended) *NX_CHAR* <=

measurement: (required) *NX_CHAR* <=

Obligatory value: `voltage`

type: (optional) *NX_CHAR* <=

value: (required) *NX_FLOAT* <=

sample_bias_potentiostat: (recommended) *NXactuator* <=

name: (recommended) *NX_CHAR* <=

actuation_target: (required) *NX_CHAR* <=

Obligatory value: `voltage`

type: (optional) *NX_CHAR* <=

PID_CONTROLLER: (recommended) *NXpid_controller* <=

setpoint: (recommended) *NX_FLOAT* {units=*NX_VOLTAGE*} <=

device_information: (recommended) *NXfabrication* <=

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

pressure_gauge: (recommended) *NXsensor* <=

Device to measure the gas pressure in the instrument.

name: (recommended) *NX_CHAR* <=

measurement: (required) *NX_CHAR* <=

Obligatory value: `pressure`

type: (optional) *NX_CHAR* <=

value: (required) *NX_FLOAT* {units=*NX_PRESSURE*} <=

In case of a single or averaged gas pressure measurement, this is the scalar gas pressure. It can also be an 1D array of measured pressures (without time stamps).

value_log: (optional) *NXlog* <=

value: (required) *NX_NUMBER* {units=*NX_PRESSURE*} <=

In the case of an experiment in which the gas pressure changes and is recorded, this is an array of length m of gas pressures.

flood_gun: (optional) *NXactuator* <=

Device to bring low-energy electrons to the sample for charge neutralization

name: (recommended) *NX_CHAR* <=

actuation_target: (required) *NX_CHAR* <=

Obligatory value: `current`

type: (optional) *NX_CHAR* <=

current: (recommended) *NX_FLOAT* {units=*NX_CURRENT*}

In case of a fixed or averaged electron current, this is the scalar current. It can also be an 1D array of output current (without time stamps).

current_log: (optional) *NXlog* <=

value: (required) *NX_NUMBER* {units=*NX_CURRENT*} <=

In the case of an experiment in which the electron current is changed and recorded with time stamps, this is an array of length m of current setpoints.

monochromator_TYPE: (optional) *NXmonochromator* <=

If any of the beams is monochromatized, an `NXmonochromator` can be used to describe the properties of the monochromator.

energy: (recommended) *NX_FLOAT* {units=*NX_ENERGY*} <=

associated_beam: (required) *NX_CHAR*

A reference to a beam emitted by this source. Should be named with the same suffix, e.g., for `monochromator_probe` it should refer to `beam_probe`.

Example:

- /entry/instrument/monochromator_probe/associated_beam = /entry/instrument/beam_probe

INSERTION_DEVICE: (optional) *NXinsertion_device* <=

Insertion device if synchrotron radiation is used for the MPES experiment.

history: (optional) *NXhistory* <=

A set of activities that occurred to the instrument prior to/during the photoemission experiment, including any activities performed on the individual instrument parts. This group can be used to describe the preparation of the instrument prior to the experiment, e.g. the cleaning procedure for a spin filter crystal.

energy_axis_calibration: (recommended) *NXcalibration*

Calibration event on the energy axis.

For XPS, the calibration should ideally be performed according to ISO 15472:2010 specification.

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `energy`

calibrated_axis: (required) *NX_FLOAT* {units=*NX_ENERGY*} <=

This is the calibrated energy axis to be used for data plotting.

AXIS_**axis_calibration**: (optional) *NXcalibration*

Calibration event for one of the axes in the *NXdata*.

The naming of these calibrations should follow those in the *NXdata*. For example, for the momentum axis `kx`, the corresponding calibration should be called `kx_axis_calibration`.

calibrated_axis: (recommended) *NX_FLOAT* <=

energy_referencing: (optional) *NXcalibration*

For energy referencing, the measured energies are corrected for the charging potential (i.e., the electrical potential of the surface region of an insulating sample, caused by irradiation) such that those energies correspond to a sample with no surface charge. Usually, the energy axis is adjusted by shifting all energies uniformly until one well-defined emission line peak (or the Fermi edge) is located at a known _correct_ energy.

This concept is related to term 12.74 ff. of the ISO 18115-1:2023 standard.

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `energy`

level: (recommended) *NX_CHAR*

Electronic core or valence level that was used for the calibration.

This should be single string defining the core or valence level that was used for energy referencing.

The notation should be the same as the one described in the *NXmpes/ENTRY/transitions* field.

reference_peak: (recommended) *NX_CHAR*

Reference peak that was used for the calibration.

For example: adventitious carbon | C-C | metallic Au | elemental Si | Fermi edge | vacuum level

binding_energy: (recommended) *NX_FLOAT* {units=*NX_ENERGY*}

The binding energy (in units of eV) that the specified emission line appeared at, after adjusting the binding energy scale.

This concept is related to term 12.16 of the ISO 18115-1:2023 standard.

offset: (recommended) *NX_FLOAT* {units=*NX_ENERGY*}

Offset between measured binding energy and calibrated binding energy of the emission line.

calibrated_axis: (recommended) *NX_FLOAT* {units=*NX_ENERGY*} <=

This is the calibrated energy axis to be used for data plotting.

This could be a link to /entry/data/energy.

transmission_correction: (optional) *NXcalibration*

In the transmission correction, each intensity measurement for electrons of a given kinetic energy is multiplied by the corresponding value in the relative_intensity field of the transmission_function. This calibration procedure is used to account for energy-dependent transmission efficiencies in certain lens modes. **transmission_function**: (recommended) *NXdata* <=

Transmission function of the electron analyzer.

The transmission function (TF) specifies the detection efficiency for electrons of different kinetic energy passing through the electron analyzer.

This can be a link to /entry/instrument/electronanalyzer/transmission_function.

@**signal**: (required) *NX_CHAR* <=

Obligatory value: `relative_intensity`

@**axes**: (required) *NX_CHAR* <=

Obligatory value: `['kinetic_energy']`

kinetic_energy: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_transmission_function]) {units=*NX_ENERGY*}

Kinetic energy values

relative_intensity: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_transmission_function]) {units=*NX_UNITLESS*}

Relative transmission efficiency for the given kinetic energies

REGISTRATION: (optional) *NXregistration*

Describes the operations of image registration (i.e. affine transformations like rotations or translations).

DISTORTION: (optional) *NXdistortion*

Describes the operations of image distortion correction.

CALIBRATION: (optional) *NXcalibration*

Any further calibration procedures. For example, a calibration event for the photoemission counts (e.g., by dividing by some base line intensity I_0.).

FIT: (optional) *NXfit*

Any fit procedures.

SAMPLE: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

chemical_formula: (recommended) *NX_CHAR* <=

atom_types: (recommended) *NX_CHAR*

Array 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*.

physical_form: (recommended) *NX_CHAR* <=

situation: (recommended) *NX_CHAR* <=

Any of these values:

- `vacuum`
- `inert atmosphere`
- `oxidizing atmosphere`
- `reducing atmosphere`

history: (recommended) *NXhistory* <=

A set of activities that occurred to the sample prior to/during photoemission experiment. **sample_preparation**: (recommended) *NXactivity* <=

Details about the sample preparation for the photoemission experiment (e.g. UHV cleaving, in-situ growth, sputtering/annealing, etc.).

start_time: (required) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

method: (recommended) *NX_CHAR*

Details about the method of sample preparation before the photoemission experiment.

temperature_env: (recommended) *NXenvironment* <=

Sample temperature (either controlled or just measured) and actuators/sensors controlling/measuring it.

value: (optional) *NX_FLOAT* {units=*NX_TEMPERATURE*} <=

This is to be used if there is no actuator/sensor that controls/measures the temperature.

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 temperature is not advised, but using NXsensor and NXactuator is strongly recommended instead.

temperature_sensor: (recommended) *NXsensor* <=

Temperature sensor measuring the sample temperature.

In most cases, this can be a link to /entry/instrument/manipulator/temperature_sensor if a manipulator is present in the instrument.

sample_heater: (optional) *NXactuator* <=

Device to heat the sample.

In most cases, this can be a link to /entry/instrument/manipulator/sample_heater if a manipulator is present in the instrument.

cryostat: (optional) *NXactuator* <=

Cryostat for cooling the sample.

In most cases, this can be a link to /entry/instrument/manipulator/cryostat if a manipulator is present in the instrument.

gas_pressure_env: (recommended) *NXenvironment* <=

Gas pressure surrounding the sample and actuators/sensors controlling/measuring it.

value: (optional) *NX_FLOAT* {units=*NX_PRESSURE*} <=

This is to be used if there is no actuator/sensor that controls/measures the gas pressure around the sample. An example would be a UHV experiment where the gas pressure is not monitored.

Note that this method for recording the gas pressure is not advised, but using NXsensor and NXactuator is strongly recommended instead.

pressure_gauge: (recommended) *NXsensor* <=

Gauge measuring the gas pressure.

In most cases, this can be a link to /entry/instrument/pressure_gauge or to another NXsensor measuring gas pressure (typically, the gauge in closest proximity to the sample) if such a pressure gauge is present in the instrument.

bias_env: (recommended) *NXenvironment* <=

Bias of the sample with respect to analyzer ground and actuators/sensors controlling/measuring it.

This concept is related to term 8.41 of the ISO 18115-1:2023 standard.

value: (optional) *NX_FLOAT* {units=*NX_VOLTAGE*} <=

This is to be used if there is no actuator/sensor that controls/measures the bias.

Note that this method for recording the bias is not advised, but using NXsensor and NXactuator is strongly recommended instead.

voltmeter: (recommended) *NXsensor* <=

Sensor measuring the applied voltage.

In most cases, this can be a link to /entry/instrument/manipulator/sample_bias_voltmeter if a manipulator is present in the instrument.

potentiostat: (optional) *NXactuator* <=

Actuator applying a voltage to sample and sample holder.

In most cases, this can be a link to /entry/instrument/manipulator/sample_bias_potentiostat if a manipulator is present in the instrument.

drain_current_env: (optional) *NXenvironment* <=

Drain current of the sample and sample holder.

value: (optional) *NX_FLOAT* {units=*NX_CURRENT*} <=

This is to be used if there is no actuator/sensor that controls/measures the drain current.

Note that this method for recording the drain current is not advised, but using NXsensor and NXactuator is strongly recommended instead.

ammeter: (recommended) *NXsensor* <=

Ammeter measuring the drain current of the sample and sample holder.

In most cases, this can be a link to /entry/instrument/manipulator/drain_current_ammeter if a manipulator is present in the instrument.

flood_gun_current_env: (optional) *NXenvironment* <=

Current of low-energy electrons to the sample (for charge neutralization) and actuators/sensors controlling/measuring it.

value: (optional) *NX_FLOAT* {units=*NX_CURRENT*} <=

This is to be used if there is no actuator/sensor that controls/measures the flood_gun_current.

Note that this method for recording the flood gun current is not advised, but using NXsensor and NXactuator is strongly recommended instead.

flood_gun: (recommended) *NXactuator* <=

Flood gun creating a current of low-energy electrons.

In most cases this can be a link to /entry/instrument/flood_gun if a flood_gun is present in the instrument.

DATA: (required) *NXdata* <=

The NXdata group containing a view on the measured data.

This NXdata group contains a collection of the main relevant fields (axes). Axes should be named according to the conventions defined below. Note that this list is a glossary with explicitly named axis names, which is only intended to cover the most common measurement axes and is therefore not complete. It is possible to add axes with other names at any time.

In NXmpes, it is recommended to provide an energy axis.

@signal: (required) *NX_CHAR* <=

Obligatory value: `data`

@energy_indices: (recommended) *NX_INT* <=

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 (or classes inheriting from NXprocess).

energy: (recommended) *NX_NUMBER* {units=*NX_ENERGY*}

Calibrated axis for the energy of the measured electrons.

@type: (required) *NX_CHAR*

The energy can be either stored as kinetic or as binding energy.

Any of these values:

- `kinetic`: Calibrated kinetic energy axis. In case the kinetic energy axis is referenced to the Fermi level E_F (e.g., in entry/process/energy_referencing), kinetic energies E are provided as $E - E_F$. This concept is related to term 3.35 of the ISO 18115-1:2023 standard.
- `binding`: Calibrated binding energy axis. This concept is related to term 12.16 of the ISO 18115-1:2023 standard.

photon_energy: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

Calibrated photon energy of the incoming probe beam.

Could be a link to /entry/instrument/beam_probe/incident_energy.

kx: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Calibrated k-space coordinate in x direction. It is envisioned that the axes in momentum space are named `kx`, `ky`, and `kz`. Typically, the vectors in momentum space are defined such that `kx` and `ky` comprise the parallel component, while `kz` is the perpendicular component.

It is also possible to define `k_parallel` and `k_perp` for the parallel and perpendicular momenta, respectively.

Units are typically 1/angstrom.

ky: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Calibrated k-space coordinate in y direction. For more information, see the definition of the *kx* axis.

kz: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Calibrated k-space coordinate in z direction. For more information, see the definition of the *kx* axis.

k_parallel: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Calibrated parallel component in k-space.

`k_parallel` and *k_perpendicular* describe how the electron's wave vector `k` is split into components relative to the surface.

`k_parallel` is the component of the electron's wave vector that is parallel to the surface. It is conserved during the photoemission process. This means that the electron's momentum along the surface inside the material is directly related to its measured momentum outside the material.

Units are typically 1/angstrom.

k_perpendicular: (optional) *NX_NUMBER* {units=*NX_WAVENUMBER*}

Calibrated perpendicular component in k-space.

`k_perpendicular` is the component that is normal (perpendicular) to the surface. It is not conserved during photoemission because the electron experiences a potential change when it exits the material into vacuum. To determine `k_perpendicular` inside the material, one typically needs to estimate the inner potential V_0, which accounts for the energy shift due to the material's work function and electronic structure.

Units are typically 1/angstrom.

angular0: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

First calibrated angular coordinate. It is envisioned that the axes in angular space are named `angular0` and `angular1`.

The angular axes should be named in order of decreasing speed, i.e., `angular0` should be the fastest scan axis and `angular1` should be the slow-axis angular coordinate. However, `angular0` may also be second slow axis if the measurement is angularly integrated and `angular1` could also be the second fast axis in the case of simultaneous dispersion in two angular dimensions.

angular1: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Second calibrated angular coordinate.

For more information, see the definition of the *angular0* axis.

This is typically the slower scan axis compared to `angular0`.

spatial0: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

First calibrated spatial coordinate. It is envisioned that the axes in angular space are named `spatial0` and `spatial1`.

The spatial axes should be named in order of decreasing speed, i.e., `spatial0` should be the fastest scan axis and *spatial1* ` should be the slow-axis spatial coordinate. However, `spatial` may also be second slow axis if the measurement is spatially integrated and `spatial1` could also be the second fast axis in the case of simultaneous dispersion in two spatial dimensions.

spatial1: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Second calibrated spatial coordinate.

For more information, see the definition of the *spatial0* axis.

This is typically the slower scan axis compared to `spatial0`.

delay: (optional) *NX_NUMBER* {units=*NX_TIME*} <=

Calibrated pump-probe delay time. Could be a link to /entry/instrument/beam_pump/pulse_delay.

temperature: (optional) *NX_NUMBER* {units=*NX_TIME*}

Calibrated temperature axis in case of experiments where the temperature was scanned. This is typically the sample temperature and could be linked from /entry/sample/temperature_env/temperature_sensor/value.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmpes/ENTRY-group*
- */NXmpes/ENTRY/AXIS_axis_calibration-group*
- */NXmpes/ENTRY/AXIS_axis_calibration/calibrated_axis-field*
- */NXmpes/ENTRY/CALIBRATION-group*
- */NXmpes/ENTRY/COORDINATE_SYSTEM-group*
- */NXmpes/ENTRY/DATA-group*
- */NXmpes/ENTRY/DATA/angular0-field*
- */NXmpes/ENTRY/DATA/angular1-field*
- */NXmpes/ENTRY/DATA/data-field*
- */NXmpes/ENTRY/DATA/delay-field*
- */NXmpes/ENTRY/DATA/energy-field*
- */NXmpes/ENTRY/DATA/energy@type-attribute*
- */NXmpes/ENTRY/DATA/k_parallel-field*
- */NXmpes/ENTRY/DATA/k_perpendicular-field*
- */NXmpes/ENTRY/DATA/kx-field*
- */NXmpes/ENTRY/DATA/ky-field*
- */NXmpes/ENTRY/DATA/kz-field*
- */NXmpes/ENTRY/DATA/photon_energy-field*
- */NXmpes/ENTRY/DATA/spatial0-field*
- */NXmpes/ENTRY/DATA/spatial1-field*
- */NXmpes/ENTRY/DATA/temperature-field*
- */NXmpes/ENTRY/DATA@energy_indices-attribute*
- */NXmpes/ENTRY/DATA@signal-attribute*
- */NXmpes/ENTRY/definition-field*

- */NXmpes/ENTRY/definition@version-attribute*
- */NXmpes/ENTRY/DISTORTION-group*
- */NXmpes/ENTRY/end_time-field*
- */NXmpes/ENTRY/energy_axis_calibration-group*
- */NXmpes/ENTRY/energy_axis_calibration/calibrated_axis-field*
- */NXmpes/ENTRY/energy_axis_calibration/physical_quantity-field*
- */NXmpes/ENTRY/energy_referencing-group*
- */NXmpes/ENTRY/energy_referencing/binding_energy-field*
- */NXmpes/ENTRY/energy_referencing/calibrated_axis-field*
- */NXmpes/ENTRY/energy_referencing/level-field*
- */NXmpes/ENTRY/energy_referencing/offset-field*
- */NXmpes/ENTRY/energy_referencing/physical_quantity-field*
- */NXmpes/ENTRY/energy_referencing/reference_peak-field*
- */NXmpes/ENTRY/FIT-group*
- */NXmpes/ENTRY/INSTRUMENT-group*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe-group*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe/associated_source-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe/distance-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe/extent-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe/incident_energy-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe/incident_energy_spread-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_probe/incident_polarization-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump-group*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump/associated_source-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump/distance-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump/extent-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump/incident_energy-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump/incident_energy_spread-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_pump/incident_polarization-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE-group*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE/associated_source-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE/distance-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE/extent-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE/incident_energy-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE/incident_energy_spread-field*
- */NXmpes/ENTRY/INSTRUMENT/beam_TYPE/incident_polarization-field*

- */NXmpes/ENTRY/INSTRUMENT/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/angular_acceptance-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/contrast_aperture-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/field_aperture-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/iris-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/lens_mode-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/projection-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/scheme-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/spatial_acceptance-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/description-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/amplifier_type-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/detector_type-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data-group*

- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/angular0-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/angular1-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/delay-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/energy-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/energy@type-attribute
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/k_parallel-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/k_perpendicular-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/kx-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/ky-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/kz-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/photon_energy-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/pixel_x-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/pixel_y-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/raw-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/spatial0-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/spatial1-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data/temperature-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ELECTRON_DETECTOR/raw_data@signal-attribute
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/energy_resolution-group
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/energy_resolution/physical_quantity-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/energy_resolution/resolution-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/energy_resolution/type-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION-group
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/device_information-group
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/device_information/identifier-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/device_information/model-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/device_information/vendor-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/drift_energy-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/energy_scan_mode-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/entrance_slit-group

- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/exit_slit-group
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/pass_energy-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/scheme-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/fast_axes-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/slow_axes-field
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transmission_function-group
- /NXmpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/work_function-field
- /NXmpes/ENTRY/INSTRUMENT/energy_resolution-group
- /NXmpes/ENTRY/INSTRUMENT/energy_resolution/physical_quantity-field
- /NXmpes/ENTRY/INSTRUMENT/energy_resolution/relative_resolution-field
- /NXmpes/ENTRY/INSTRUMENT/energy_resolution/resolution-field
- /NXmpes/ENTRY/INSTRUMENT/energy_resolution/type-field
- /NXmpes/ENTRY/INSTRUMENT/flood_gun-group
- /NXmpes/ENTRY/INSTRUMENT/flood_gun/actuation_target-field
- /NXmpes/ENTRY/INSTRUMENT/flood_gun/current-field
- /NXmpes/ENTRY/INSTRUMENT/flood_gun/current_log-group
- /NXmpes/ENTRY/INSTRUMENT/flood_gun/current_log/value-field
- /NXmpes/ENTRY/INSTRUMENT/flood_gun/name-field
- /NXmpes/ENTRY/INSTRUMENT/flood_gun/type-field
- /NXmpes/ENTRY/INSTRUMENT/history-group
- /NXmpes/ENTRY/INSTRUMENT/INSERTION_DEVICE-group
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR-group
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/cryostat-group
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/cryostat/actuation_target-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/cryostat/name-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/cryostat/PID_CONTROLLER-group
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/cryostat/PID_CONTROLLER/setpoint-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/cryostat/type-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/device_information-group
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/device_information/identifier-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/device_information/model-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/device_information/vendor-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/drain_current_ammeter-group
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/drain_current_ammeter/measurement-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/drain_current_ammeter/name-field
- /NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/drain_current_ammeter/type-field

- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/drain_current_ammeter/value-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_potentiostat-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_potentiostat/actuation_target-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_potentiostat/name-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_potentiostat/PID_CONTROLLER-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_potentiostat/PID_CONTROLLER/setpoint-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_potentiostat/type-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_voltmeter-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_voltmeter/measurement-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_voltmeter/name-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_voltmeter/type-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_bias_voltmeter/value-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater/actuation_target-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater/name-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater/output_heater_power-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater/PID_CONTROLLER-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater/PID_CONTROLLER/setpoint-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/sample_heater/type-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/temperature_sensor-group*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/temperature_sensor/measurement-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/temperature_sensor/name-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/temperature_sensor/type-field*
- */NXmpes/ENTRY/INSTRUMENT/MANIPULATOR/temperature_sensor/value-field*
- */NXmpes/ENTRY/INSTRUMENT/monochromator_TYPE-group*
- */NXmpes/ENTRY/INSTRUMENT/monochromator_TYPE/associated_beam-field*
- */NXmpes/ENTRY/INSTRUMENT/monochromator_TYPE/energy-field*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge-group*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge/measurement-field*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge/name-field*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge/type-field*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge/value-field*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge/value_log-group*
- */NXmpes/ENTRY/INSTRUMENT/pressure_gauge/value_log/value-field*
- */NXmpes/ENTRY/INSTRUMENT/RESOLUTION-group*

- */NXmpes/ENTRY/INSTRUMENT/source_probe-group*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/associated_beam-field*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/name-field*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/probe-field*
- */NXmpes/ENTRY/INSTRUMENT/source_probe/type-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump-group*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/associated_beam-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/name-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/probe-field*
- */NXmpes/ENTRY/INSTRUMENT/source_pump/type-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE-group*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/associated_beam-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/device_information-group*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/device_information/identifier-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/device_information/model-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/device_information/vendor-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/name-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/probe-field*
- */NXmpes/ENTRY/INSTRUMENT/source_TYPE/type-field*
- */NXmpes/ENTRY/method-field*
- */NXmpes/ENTRY/REGISTRATION-group*
- */NXmpes/ENTRY/SAMPLE-group*
- */NXmpes/ENTRY/SAMPLE/atom_types-field*
- */NXmpes/ENTRY/SAMPLE/bias_env-group*
- */NXmpes/ENTRY/SAMPLE/bias_env/potentiostat-group*
- */NXmpes/ENTRY/SAMPLE/bias_env/value-field*
- */NXmpes/ENTRY/SAMPLE/bias_env/voltmeter-group*
- */NXmpes/ENTRY/SAMPLE/chemical_formula-field*

- */NXmpes/ENTRY/SAMPLE/drain_current_env-group*
- */NXmpes/ENTRY/SAMPLE/drain_current_env/ammeter-group*
- */NXmpes/ENTRY/SAMPLE/drain_current_env/value-field*
- */NXmpes/ENTRY/SAMPLE/flood_gun_current_env-group*
- */NXmpes/ENTRY/SAMPLE/flood_gun_current_env/flood_gun-group*
- */NXmpes/ENTRY/SAMPLE/flood_gun_current_env/value-field*
- */NXmpes/ENTRY/SAMPLE/gas_pressure_env-group*
- */NXmpes/ENTRY/SAMPLE/gas_pressure_env/pressure_gauge-group*
- */NXmpes/ENTRY/SAMPLE/gas_pressure_env/value-field*
- */NXmpes/ENTRY/SAMPLE/history-group*
- */NXmpes/ENTRY/SAMPLE/history/sample_preparation-group*
- */NXmpes/ENTRY/SAMPLE/history/sample_preparation/end_time-field*
- */NXmpes/ENTRY/SAMPLE/history/sample_preparation/method-field*
- */NXmpes/ENTRY/SAMPLE/history/sample_preparation/start_time-field*
- */NXmpes/ENTRY/SAMPLE/identifier-field*
- */NXmpes/ENTRY/SAMPLE/name-field*
- */NXmpes/ENTRY/SAMPLE/physical_form-field*
- */NXmpes/ENTRY/SAMPLE/situation-field*
- */NXmpes/ENTRY/SAMPLE/temperature_env-group*
- */NXmpes/ENTRY/SAMPLE/temperature_env/cryostat-group*
- */NXmpes/ENTRY/SAMPLE/temperature_env/sample_heater-group*
- */NXmpes/ENTRY/SAMPLE/temperature_env/temperature_sensor-group*
- */NXmpes/ENTRY/SAMPLE/temperature_env/value-field*
- */NXmpes/ENTRY/start_time-field*
- */NXmpes/ENTRY/title-field*
- */NXmpes/ENTRY/transitions-field*
- */NXmpes/ENTRY/transmission_correction-group*
- */NXmpes/ENTRY/transmission_correction/transmission_function-group*
- */NXmpes/ENTRY/transmission_correction/transmission_function/kinetic_energy-field*
- */NXmpes/ENTRY/transmission_correction/transmission_function/relative_intensity-field*
- */NXmpes/ENTRY/transmission_correction/transmission_function@axes-attribute*
- */NXmpes/ENTRY/transmission_correction/transmission_function@signal-attribute*
- */NXmpes/ENTRY/USER-group*
- */NXmpes/ENTRY/USER/affiliation-field*
- */NXmpes/ENTRY/USER/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXmpes.nxdl.xml

NXmpes_arpes

Status:

application definition, extends *NXmpes*

Description:

This is a general application definition for angle-resolved (multidimensional) photoelectron spectroscopy (ARPES).

Symbols:

No symbol table

Groups cited:
NXaperture, *NXcollectioncolumn*, *NXcoordinate_system*, *NXdata*, *NXelectronanalyzer*, *NXenergydispersion*, *NXentry*, *NXinstrument*, *NXresolution*, *NXsample*, *NXtransformations*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXmpes_arpes`

@version: (required) *NX_CHAR* <=

method: (recommended) *NX_CHAR* <=

Name of the experimental method.

If applicable, this name should match the terms given by Clause 11 of the ISO 18115-1:2023 specification.

Examples include:

- angle-resolved photoelectron spectroscopy (ARPES)
- time-resolved angle-resolved X-ray photoelectron spectroscopy (trARPES)
- spin-resolved angle-resolved X-ray photoelectron spectroscopy (spin-ARPES)

arpes_geometry: (required) *NXcoordinate_system* <=

depends_on: (required) *NX_CHAR* <=

Link to transformations defining an ARPES base coordinate system, which is defined such that the positive z-axis points towards the analyzer entry, and the x-axis lies within the beam/analyzer plane.

TRANSFORMATIONS: (required) *NXtransformations* <=

Set of transformations, describing the orientation of the ARPES coordinate system with respect to the beam coordinate system (.).

INSTRUMENT: (required) *NXinstrument* <=

angularN**_resolution**: (recommended) *NXresolution* <=

Overall angular resolution along the Nth angular axis. Create one such entry per relevant angular axis, corresponding to the angular axes in NXdata. For hemispherical analyzers, angular0_resolution corresponds to the direction along the analyzer slit, and angular1_resolution to the one perpendicular to it.

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `angle`

type: (recommended) *NX_CHAR* <=

resolution: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

ELECTRONANALYZER: (required) *NXelectronanalyzer* <=

depends_on: (required) *NX_CHAR* <=

Reference to the last transformation describing the orientation of the analyzer relative to the beam, e.g. transformations/analyzer_elevation.

angularN**_resolution**: (recommended) *NXresolution* <=

Analyzer angular resolution along the Nth angular axis. Create one such entry per relevant angular axis, corresponding to the angular axes in NXdata. For hemispherical analyzers, angular0_resolution corresponds to the direction along the analyzer slit, and angular1_resolution to the one perpendicular to it.

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `angle`

type: (recommended) *NX_CHAR* <=

resolution: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

transformations: (required) *NXtransformations* <=

Set of transformations, describing the relative orientation of the analyzer with respect to the beam coordinate system (.).

analyzer_rotation: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Rotation about the analyzer lens axis. Its zero reference is defined such that the angular0 axis is increasing towards the positive y axis (analyzer slit vertical).

@**transformation_type**: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@**vector**: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@**depends_on**: (required) *NX_CHAR* <=

Path to a transformation that places the analyzer origin system into the arpes_geometry coordinate system.

analyzer_elevation: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Elevation of the effective analyzer acceptance area, e.g. realized by deflectors, or as one angle in a TOF detector. If a resolved angle, place the calibrated axis coordinates here.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 1, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `analyzer_dispersion`

analyzer_dispersion: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

In-plane analyzer coordinate along a dispersive direction, e.g. along an analyzer slit. If a resolved angle, place the calibrated coordinates here.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[1, 0, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `analyzer_rotation`

COLLECTIONCOLUMN: (required) *NXcollectioncolumn* <=

scheme: (recommended) *NX_CHAR* <=

Scheme of the electron collection column.

Any of these values: `angular dispersive|non-dispersive`

angular_acceptance: (recommended) *NX_FLOAT* <=

ENERGYDISPERSION: (required) *NXenergydispersion* <=

diameter: (recommended) *NX_NUMBER* {units=*NX_LENGTH*}

entrance_slit: (recommended) *NXaperture* <=

shape: (required) *NX_CHAR* <=

Any of these values: `straight slit|curved slit`

SAMPLE: (required) *NXsample* <=

situation: (required) *NX_CHAR* <=

Obligatory value: `vacuum`

depends_on: (required) *NX_CHAR* <=

Reference to the end of the transformation chain, orienting the sample surface within the arpes_geometry coordinate system (sample_azimuth or anything depending on it).

transformations: (required) *NXtransformations* <=

Set of transformations, describing the relative orientation of the sample with respect to the arpes_geometry coordinate system.

sample_azimuth: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Rotation about the z axis (azimuthal rotation within the sample plane).

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `offset_azimuth`

offset_azimuth: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Offset of azimuthal rotation.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `sample_tilt`

sample_tilt: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Rotation about the x axis (typically a manipulator tilt).

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[1, 0, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `offset_tilt`

offset_tilt: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Offset of tilt rotation.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[1, 0, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `sample_polar`

sample_polar: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Rotation about the y axis (typically the long manipulator axis).

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 1, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `offset_polar`

offset_polar: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Offset of polar rotation.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 1, 0]`

@depends_on: (required) *NX_CHAR* <=

Path to a transformation that places the sample surface into the origin of the arpes_geometry coordinate system.

DATA: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

There is a field named data that contains the signal.

Obligatory value: `data`

@axes: (required) *NX_CHAR* <=

There are three dimensions, one energy and two angular coordinates. Any coordinates that do not move, are represented by one point.

Obligatory value: `['angular0', 'angular1', 'energy']`

@energy_indices: (required) *NX_INT* <=

@angular0_indices: (required) *NX_INT* <=

@angular1_indices: (required) *NX_INT* <=

energy: (required) *NX_NUMBER* {units=*NX_ENERGY*} <=

Values on the energy axis.

angular0: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Trace of the first angular axis.

angular1: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Trace of the second axis. Could be linked from the respective `@reference` field.

data: (required) *NX_NUMBER* {units=*NX_ANY*} <=

Represents a measurement of photoemission counts over a three-dimensional space where the varied axes are energy, and one or more angular coordinates. 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_arpes/ENTRY-group*
- */NXmpes_arpes/ENTRY/arpes_geometry-group*
- */NXmpes_arpes/ENTRY/arpes_geometry/depends_on-field*
- */NXmpes_arpes/ENTRY/arpes_geometry/TRANSFORMATIONS-group*
- */NXmpes_arpes/ENTRY/DATA-group*
- */NXmpes_arpes/ENTRY/DATA/angular0-field*
- */NXmpes_arpes/ENTRY/DATA/angular1-field*
- */NXmpes_arpes/ENTRY/DATA/data-field*
- */NXmpes_arpes/ENTRY/DATA/energy-field*
- */NXmpes_arpes/ENTRY/DATA@angular0_indices-attribute*
- */NXmpes_arpes/ENTRY/DATA@angular1_indices-attribute*
- */NXmpes_arpes/ENTRY/DATA@axes-attribute*
- */NXmpes_arpes/ENTRY/DATA@energy_indices-attribute*
- */NXmpes_arpes/ENTRY/DATA@signal-attribute*
- */NXmpes_arpes/ENTRY/definition-field*
- */NXmpes_arpes/ENTRY/definition@version-attribute*
- */NXmpes_arpes/ENTRY/INSTRUMENT-group*
- */NXmpes_arpes/ENTRY/INSTRUMENT/angularN_resolution-group*
- */NXmpes_arpes/ENTRY/INSTRUMENT/angularN_resolution/physical_quantity-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/angularN_resolution/resolution-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/angularN_resolution/type-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER-group*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/angularN_resolution-group*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/angularN_resolution/physical_quantity-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/angularN_resolution/resolution-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/angularN_resolution/type-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN-group*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/angular_acceptance-field*
- */NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/scheme-field*

- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/depends_on-field
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION-group
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/diameter-field
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/entrance_slit-group
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/entrance_slit/shape-field
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations-group
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_dispersion-field
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_dispersion@depends_on-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_dispersion@transformation_type-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_dispersion@vector-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_elevation-field
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_elevation@depends_on-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_elevation@transformation_type-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_elevation@vector-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_rotation-field
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_rotation@depends_on-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_rotation@transformation_type-attribute
- /NXmpes_arpes/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_rotation@vector-attribute
- /NXmpes_arpes/ENTRY/method-field
- /NXmpes_arpes/ENTRY/SAMPLE-group
- /NXmpes_arpes/ENTRY/SAMPLE/depends_on-field
- /NXmpes_arpes/ENTRY/SAMPLE/situation-field
- /NXmpes_arpes/ENTRY/SAMPLE/transformations-group
- /NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_azimuth-field
- /NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_azimuth@depends_on-attribute
- /NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_azimuth@transformation_type-attribute
- /NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_azimuth@vector-attribute
- /NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_polar-field
- /NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_polar@depends_on-attribute

- */NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_polar@transformation_type-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_polar@vector-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_tilt-field*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_tilt@depends_on-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_tilt@transformation_type-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/offset_tilt@vector-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_azimuth-field*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_azimuth@depends_on-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_azimuth@transformation_type-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_azimuth@vector-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_polar-field*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_polar@depends_on-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_polar@transformation_type-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_polar@vector-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_tilt-field*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_tilt@depends_on-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_tilt@transformation_type-attribute*
- */NXmpes_arpes/ENTRY/SAMPLE/transformations/sample_tilt@vector-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXmpes_arpes.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 detector distance to the sample is observable independent of the axis chain, that may take precedence over the axis chain calculation.

distance_derived: (recommended) *NX_BOOLEAN*

Boolean to indicate if the distance is a derived, rather than a primary observation. If distance_derived true or is not specified, the distance is assumed to be derived from detector axis specifications.

dead_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

Detector dead time.

count_time: (recommended) *NX_NUMBER* {units=*NX_TIME*} <=

Elapsed actual counting time.

beam_center_derived: (optional) *NX_BOOLEAN*

Boolean to indicate if the distance is a derived, rather than a primary observation. If true or not provided, that value of beam_center_derived is assumed to be true.

beam_center_x: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the x position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute. Normally, this should be derived from the axis chain, but the direct specification may take precedence if it is not a derived quantity.

beam_center_y: (recommended) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the y position where the direct beam would hit the detector. This is a length and can be outside of the actual detector. The length can be in physical units or pixels as documented by the units attribute. Normally, this should be derived from the axis chain, but the direct specification may take precedence if it is not a derived quantity.

angular_calibration_applied: (optional) *NX_BOOLEAN* <=

True when the angular calibration has been applied in the electronics, false otherwise.

angular_calibration: (optional) *NX_FLOAT* (Rank: dataRank, Dimensions: [i, j, [k]]) <=

Angular calibration data.

flatfield_applied: (optional) *NX_BOOLEAN* <=

True when the flat field correction has been applied in the electronics, false otherwise.

flatfield: (optional) *NX_NUMBER* (Rank: dataRank, Dimensions: [i, j, [k]])

Flat field correction data. If provided, it is recommended that it be compressed.

flatfield_error: (optional) *NX_NUMBER* (Rank: dataRank, Dimensions: [i, j, [k]])

*** Deprecated form. Use plural form *** Errors of the flat field correction data. If provided, it is recommended that it be compressed.

flatfield_errors: (optional) *NX_NUMBER* (Rank: dataRank, Dimensions: [i, j, [k]])

Errors of the flat field correction data. If provided, it is recommended that it be compressed.

pixel_mask_applied: (optional) *NX_BOOLEAN* <=

True when the pixel mask correction has been applied in the electronics, false otherwise.

pixel_mask: (recommended) *NX_INT* (Rank: 2, Dimensions: [i, j]) <=

The 32-bit pixel mask for the detector. Can be either one mask for the whole dataset (i.e. an array with indices i, j) or each frame can have its own mask (in which case it would be an array with indices nP, i, j).

Contains a bit field for each pixel to signal dead, blind, high or otherwise unwanted or undesirable pixels. They have the following meaning:

- bit 0: gap (pixel with no sensor)
- bit 1: dead
- bit 2: under-responding

- bit 3: over-responding
- bit 4: noisy
- bit 5: -undefined-
- bit 6: pixel is part of a cluster of problematic pixels (bit set in addition to others)
- bit 7: -undefined-
- bit 8: user defined mask (e.g. around beamstop)
- bits 9-30: -undefined-
- bit 31: virtual pixel (corner pixel with interpolated value)

Normal data analysis software would not take pixels into account when a bit in (mask & 0x0000FFFF) is set. Tag bit in the upper two bytes would indicate special pixel properties that normally would not be a sole reason to reject the intensity value (unless lower bits are set.

If the full bit depths is not required, providing a mask with fewer bits is permissible.

If needed, additional pixel masks can be specified by including additional entries named pixel_mask_N, where N is an integer. For example, a general bad pixel mask could be specified in pixel_mask that indicates noisy and dead pixels, and an additional pixel mask from experiment-specific shadowing could be specified in pixel_mask_2. The cumulative mask is the bitwise OR of pixel_mask and any pixel_mask_N entries.

If provided, it is recommended that it be compressed.

countrate_correction_applied: (optional) *NX_BOOLEAN* <=

Counting detectors usually are not able to measure all incoming particles, especially at higher count-rates. Count-rate correction is applied to account for these errors.

True when count-rate correction has been applied, false otherwise.

countrate_correction_lookup_table: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [m]) <=

The countrate_correction_lookup_table defines the LUT used for count-rate correction. It maps a measured count c to its corrected value $countrate_correction_lookup_table[c]$.

m denotes the length of the table.

virtual_pixel_interpolation_applied: (optional) *NX_BOOLEAN* <=

True when virtual pixel interpolation has been applied, false otherwise.

When virtual pixel interpolation is applied, values of some pixels may contain interpolated values. For example, to account for space between readout chips on a module, physical pixels on edges and corners between chips may have larger sensor areas and counts may be distributed between their logical pixels.

bit_depth_readout: (recommended) *NX_INT* <=

How many bits the electronics record per pixel.

detector_readout_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

Time it takes to read the detector (typically milliseconds). This is important to know for time resolved experiments.

frame_time: (optional) *NX_FLOAT* {units=*NX_TIME*} <=

This is time for each frame. This is exposure_time + readout time.

gain_setting: (optional) *NX_CHAR* <=

The gain setting of the detector. This influences background. This is a detector-specific value meant to document the gain setting of the detector during data collection, for detectors with multiple available gain settings.

Examples of gain settings include:

- `standard`
- `fast`
- `auto`
- `high`
- `medium`
- `low`
- `mixed high to medium`
- `mixed medium to low`

Developers are encouraged to use one of these terms, or to submit additional terms to add to the list.

saturation_value: (optional) *NX_NUMBER* <=

The value at which the detector goes into saturation. Data above this value is known to be invalid.

For example, given a saturation_value and an underload_value, the valid pixels are those less than or equal to the saturation_value and greater than or equal to the underload_value.

underload_value: (optional) *NX_NUMBER* <=

The lowest value at which pixels for this detector would be reasonably be measured.

For example, given a saturation_value and an underload_value, the valid pixels are those less than or equal to the saturation_value and greater than or equal to the underload_value.

sensor_material: (required) *NX_CHAR* <=

At times, radiation is not directly sensed by the detector. Rather, the detector might sense the output from some converter like a scintillator. This is the name of this converter material.

sensor_thickness: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

At times, radiation is not directly sensed by the detector. Rather, the detector might sense the output from some converter like a scintillator. This is the thickness of this converter material.

threshold_energy: (optional) *NX_FLOAT* {units=*NX_ENERGY*} <=

Single photon counter detectors can be adjusted for a certain energy range in which they work optimally. This is the energy setting for this. If the detector supports multiple thresholds, this is an array.

type: (optional) *NX_CHAR* <=

Description of type such as scintillator, ccd, pixel, image plate, CMOS, ...

TRANSFORMATIONS: (optional) *NXtransformations* <=

Location for axes (transformations) to do with the detector. In the case of a single-module detector, the axes of the detector axis chain may be stored here.

COLLECTION: (optional) *NXcollection* <=

Suggested container for detailed non-standard detector information like corrections applied automatically or performance settings.

DETECTOR_MODULE: (required) *NXdetector_module* <=

Many detectors consist of multiple smaller modules that are operated in sync and store their data in a common dataset. To allow consistent parsing of the experimental geometry, this application definition requires all detectors to define a detector module, even if there is only one.

This group specifies the hyperslab of data in the data array associated with the detector that contains the data for this module. If the module is associated with a full data array, rather than with a hyperslab within a larger array, then a single module should be defined, spanning the entire array.

Note, the pixel size is given as values in the array fast_pixel_direction and slow_pixel_direction.

data_origin: (required) *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

NXoptical_spectroscopy

Status:

application definition, extends *NXobject*

Description:

A general application definition of optical spectroscopy elements, which may be used as a template to derive specialized optical spectroscopy experiments.

Possible specializations are ellipsometry, Raman spectroscopy, photoluminescence, reflectivity/transmission spectroscopy.

A general optical experiment consists of (i) a light/photon source, (ii) a sample, (iii) a detector.

For any free-text descriptions, it is recommended to use English, as this ensures the most FAIR (Findable, Accessible, Interoperable, and Reusable) representation of the information.

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_measurements: Number of measurements (1st dimension of measured data arrays). 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.

Groups cited:
NXactuator, *NXbeam_transfer_matrix_table*, *NXbeam*, *NXcalibration*, *NXcomponent*, *NXcoordinate_system*, *NXdata*, *NXdetector*, *NXentry*, *NXenvironment*, *NXfabrication*, *NXhistory*, *NXinstrument*, *NXmanipulator*, *NXmonochromator*, *NXoptical_lens*, *NXoptical_window*, *NXpid_controller*, *NXprocess*, *NXprogram*, *NXresolution*, *NXsample*, *NXsensor*, *NXsource*, *NXtransformations*, *NXuser*, *NXwaveplate*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

An application definition describing a general optical experiment.

Obligatory value: `NXoptical_spectroscopy`

@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.

title: (recommended) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

Datetime of the start of the measurement. Should be a ISO8601 date/time stamp. It is recommended to add an explicit time zone, otherwise, the local time zone is assumed per ISO8601.

It is required to enter at least one of both measurement times, either "start_time" or "end_time".

end_time: (recommended) *NX_DATE_TIME* <=

Datetime of the end of the measurement. Should be a ISO8601 date/time stamp. It is recommended to add an explicit time zone, otherwise the local time zone is assumed per ISO8601.

It is required to enter at least one of both measurement times, either "start_time" or "end_time".

identifier_experiment: (recommended) *NX_CHAR* <=

experiment_description: (optional) *NX_CHAR* <=

An optional free-text description of 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 this field.

The reason is that such a free-text field is difficult to machine-interpret. The motivation behind keeping this field for now is to learn how far the current base classes need extension based on user feedback.

experiment_type: (required) *NX_CHAR*

Specify the type of the optical experiment.

Use another term if none of these methods are suitable. You may specify fundamental characteristics or properties in the experimental sub-type.

For Raman spectroscopy or ellipsometry use the respective specializations of NXoptical_spectroscopy.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `photoluminescence`
- `transmission spectroscopy`
- `reflection spectroscopy`

experiment_sub_type: (optional) *NX_CHAR*

Specify a special property or characteristic of the experiment, which specifies the generic experiment type.

Any of these values or a custom value (if you use a custom value, also set @custom=True): `time resolved | imaging | pump-probe`

beam_ref_frame: (optional) *NXcoordinate_system*

depends_on: (required) *NX_CHAR* <=

This refers to the coordinate system along the beam path. The origin and base is defined at z=0, where the incident beam hits the sample at the surface.

TRANSFORMATIONS: (required) *NXtransformations* <=

This is the default NeXus coordinate system (McStas), if the transformation does not change the coordinate system at all - i.e. it is unity. Otherwise, by this a respective transformation of the beam reference frame to the default reference frame could be made. i.e. exchange of x and z coordinate, rotation of respective coordinates towards each other.

sample_normal_ref_frame: (optional) *NXcoordinate_system*

depends_on: (required) *NX_CHAR* <=

Link to transformations defining the sample-normal base coordinate system, which is defined such that the positive z-axis is parallel to the sample normal, and the x-y-plane lies inside the sample surface.

TRANSFORMATIONS: (required) *NXtransformations* <=

Set of transformations, describing the orientation of the sample-normal coordinate system with respect to the beam coordinate system (.).

USER: (optional) *NXuser* <=

Contact information and eventually details of at persons who performed the measurements. This can be for example the principal investigator or student. Examples are: name, affiliation, address, telephone number, email, role as well as identifiers such as orcid or similar. It is recommended to add multiple users if relevant.

Due to data privacy concerns, there is no minimum requirement. If no user with specific name is allowed to be given, it is required to assign at least an affiliation

INSTRUMENT: (required) *NXinstrument* <=

Devices or elements of the optical spectroscopy setup described with its properties and general information.

This includes for example: - The beam device's or instrument's model, company, serial number, construction year, etc. - Used software or code - Experiment descriptive parameters as reference frames, resolution, calibration - Photon beams with their respective properties such as angles and polarization - Various optical beam path devices, which interact, manipulate or measure optical beams - Characteristics of the medium surrounding the sample - "Beam devices" for a beam path description - Stages(NXmanipulator) - Sensors and actuators to control or measure sample or beam properties

angle_reference_frame: (recommended) *NX_CHAR*

Defines the reference frame which is used to describe the sample orientation with respect to the beam directions.

A beam centered description is the default and uses 4 angles(similar to XRD):

- Omega (angle between sample surface and incident beam)

- 2Theta (angle between the transmitted beam and the detection beam)
- **Chi (sample tilt angle, angle between plane#1 and the surface normal,**
  plane#1 = spanned by incidence beam and detection and detection. If Chi=0°, then plane#1 is the plane of incidence in reflection setups)
- **Phi (inplane rotation of sample, rotation axis is the samples**
  surface normal)

A sample normal centered description is possible as well:

- angle of incidence (angle between incident beam and sample surface)
- angle of detection (angle between detection beam and sample surface)
- angle of incident and detection beam
- angle of in-plane sample rotation (direction along the sample's surface normal)

Any of these values: `beam centered|sample-normal centered`

omega: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle between sample incident beam and sample surface.

twotheta: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle between incident and detection beam

chi: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Sample tilt between sample normal, and the plane spanned by detection and incident beam.

phi: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Inplane rotation of the sample, with rotation axis along sample normal.

angle_of_incidence: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle(s) of the incident beam vs. the normal of the bottom reflective (substrate) surface in the sample. These two directions span the plane of incidence.

angle_of_detection: (optional) *NX_NUMBER* {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. These two directions span the plane of detection.

angle_of_incident_and_detection_beam: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle between the incident and detection beam. If angle_of_detection + angle_of_incidence = angle_of_incident_and_detection_beam, then the setup is a reflection setup. If angle_of_detection + angle_of_incidence != angle_of_incident_and_detection_beam then the setup may be a light scattering setup. (i.e. 90° + 90° != 90°, i.e. incident and detection beam in the sample surface, but the angle source-sample-detector is 90°)

angle_of_in_plane_sample_rotation: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of the inplane orientation of the sample. This might be an arbitrary, angle without specific relation to the sample symmetry, of the angle to a specific sample property (i.e. crystallographic axis or sample shape such as wafer flat)

lateral_focal_point_offset: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Specify if there is a lateral offset on the sample surface, between the focal points of the incident beam and the detection beam.

beam_TYPE: (required) *NXbeam* <=

This can be used to describe properties of a photon beam. A beam can be connected to components, via their "inputs" and "outputs".

It is required to define at least one incident beam which is incident to the sample. You may specify if this beam parameters are actually measured or just nominal. If this beam is the output of a source, chose the same name appendix as for the NXsource instance (e.g. TYPE=532nm)

parameter_reliability: (required) *NX_CHAR*

Select the reliability of the respective beam characteristics. Either, the parameters are measured via another device or method or just given nominally via the properties of a light source properties (532nm, 100mW).

Any of these values: `measured` | `nominal`

incident_wavelength: (recommended) *NX_NUMBER*

incident_wavelength_spread: (recommended) *NX_NUMBER*

incident_polarization: (recommended) *NX_NUMBER* <=

extent: (recommended) *NX_FLOAT* <=

associated_source: (optional) *NX_CHAR*

The path to the device which emitted this beam (light source or frequency doubler).

This parameter is recommended, if the previous optical element is a photon source. In this way, the properties of the laser or light source can be described and associated. The beam should be named with the same appendix as the source, e.g., for TYPE=532nmlaser, there should be both a NXsource named "source_532nmlaser" and a NXbeam named "beam_532nmlaser".

Example: /entry/instrument/source_532nmlaser

beam_polarization_type: (optional) *NX_CHAR*

Any of these values:

- `linear`
- `circular`
- `elliptically`
- `unpolarized`

linear_beam_sample_polarization: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of the linear polarized light, with respect to a fixed arbitrary defined 0° position. Note that the zero reference should be a direction vector for a *reference_plane* normal in an *NXtransformations* group within *NXbeam*. This can be used if no definition of respective coordinate systems for beam and sample normal is done. If coordinate systems are defined, refer to beam "incident_polarization".

detector_TYPE: (required) *NXdetector* <=

detector_channel_type: (required) *NX_CHAR*

Any of these values: `single-channel` | `multichannel`

detector_type: (recommended) *NX_CHAR*

Description of the detector type.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `CCD`
- `photomultiplier`
- `photodiode`
- `avalanche-photodiode`
- `streak camera`
- `bolometer`
- `golay detectors`
- `pyroelectric detector`
- `deuterated triglycine sulphate`

additional_detector_hardware: (optional) *NX_CHAR*

Specify respective hardware which was used for the detector. For example special electronics required for time-correlated single photon counting (TCSPC).

raw_data: (recommended) *NXdata* <=

Contains the raw data collected by the detector before calibration. The data which is considered raw might change from experiment to experiment due to hardware pre-processing of the data. This field ideally collects the data with the lowest level of processing possible.

@signal: (required) *NX_CHAR* <=

Obligatory value: `raw`

raw: (required) *NX_NUMBER* <=

Raw data before calibration.

device_information: (recommended) *NXfabrication* <=

source_TYPE: (recommended) *NXsource* <=

type: (recommended) *NX_CHAR* <=

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `Synchrotron X-ray Source`
- `Rotating Anode X-ray`
- `Fixed Tube X-ray`
- `UV Laser`
- `Optical Laser`
- `Laser`
- `Dye-Laser`
- `Broadband Tunable Light Source`
- `Halogen lamp`
- `LED`
- `Mercury Cadmium Telluride`
- `Deuterium Lamp`
- `Xenon Lamp`
- `Globar`

name: (recommended) *NX_CHAR* <=

standard: (optional) *NX_CHAR*

If available, name/ID/norm of the light source standard.

associated_beam: (recommended) *NX_CHAR*

The path to a beam emitted by this source. Should be named with the same appendix, e.g., for TYPE=532nmlaser, there should as well be a NXbeam named "beam_532nmlaser" together with this source instance named "source_532nmlaser"

Example: /entry/instrument/beam_532nmlaser

device_information: (recommended) *NXfabrication* <=

Details about the device information.

MONOCHROMATOR: (recommended) *NXmonochromator* <=

device_information: (recommended) *NXfabrication* <=

generic_beam_sample_angle_TYPE: (recommended) *NXtransformations*

Set of transformations, describing the relative orientation of different parts of the experiment (beams or sample). You may select one of the specified angles for incident and detection beam or sample, and then use polar and azimuthal angles to define the direction via spherical coordinates. This allows consistent definition between different coordinate system. You may refer to self defined coordinate system as well.

If "angle_reference_frame = beam centered", then this coordinate system is used: McStas system (NeXus default) (https://manual.nexusformat.org/design.html#mcstas-and-nxgeometry-system)

i.e. the z-coordinate math:*[0,0,1]* is along the incident beam direction and the x-coordinate math:*[1,0,0]* is in the horizontal plane. Hence, usually math:*[0,1,0]* is vertically oriented.

If "angle_reference_frame = sample-normal centered", then this coordinate system is used z - math:*[0,0,1]* along sample surface normal x - math:*[1,0,0]* defined by sample surface projected incident beam. y - math:*[0,1,0]* in the sample surface, orthogonal to z and x. For this case, x may be ill defined, if the incident beam is perpendicular to the sample surface. In this case, use the beam centered description.

type: (required) *NX_CHAR*

Any of these values: `incident beam|detection beam|sample`

polar: (required) *NX_NUMBER* {units=*NX_ANGLE*} *<=*

Rotation about the y axis (polar rotation within the sample plane).

@**transformation_type**: (required) *NX_CHAR <=*

Obligatory value: `rotation`

@**vector**: (required) *NX_CHAR*

Obligatory value: `[0, 1, 0]`

@**depends_on**: (required) *NX_CHAR <=*

Path to a transformation that places the sample surface into the origin of the arpes_geometry coordinate system.

azimuth: (required) *NX_NUMBER* {units=*NX_ANGLE*} *<=*

Rotation about the z axis (azimuthal rotation within the sample plane).

@**transformation_type**: (required) *NX_CHAR <=*

Obligatory value: `rotation`

@**vector**: (required) *NX_CHAR*

Obligatory value: `[0, 0, 1]`

@**depends_on**: (required) *NX_CHAR <=*

Obligatory value: `offset_tilt`

COMPONENT: (optional) *NXcomponent*

Optical components along the optical beam path.

Every object which interacts or modifies optical beam properties, may be a component, e.g. Filter, Window, Beamsplitter, Photon Source, Detector, etc,

OPTICAL_LENS: (optional) *NXoptical_lens*

This is the optical element used to focus or collect light. This may be a generic lens or microcope objectives which are used for the Raman scattering process.

type: (required) *NX_CHAR <=*

Any of these values or a custom value (if you use a custom value, also set @custom=True): `objective|lens|glass fiber|none`

device_information: (optional) *NXfabrication <=*

WAVEPLATE: (optional) *NXwaveplate*

OPTICAL_WINDOW: (optional) *NXoptical_window*

polfilter_TYPE: (optional) *NXcomponent*

Polarization filter to prepare light to be measured or to be incident on the sample. Generic polarization filter properties may be implemented via NXfilter_pol at a later stage.

filter_mechanism: (optional) *NX_CHAR*

Physical principle of the polarization filter used to create a defined incident or scattered light state.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `polarization by Fresnel reflection`
- `birefringent polarizers`
- `thin film polarizers`
- `wire-grid polarizers`

specific_polarization_filter_type: (optional) *NX_CHAR*

Specific name or type of the polarizer used.

Free text, for example: Glan-Thompson, Glan-Taylor, Rochon Prism, Wollaston Polarizer…

device_information: (optional) *NXfabrication* <=

spectralfilter_TYPE: (optional) *NXcomponent*

Spectral filter used to modify properties of the scattered or incident light.

filter_type: (optional) *NX_CHAR*

Type of laser-line filter used to suppress the laser, if measurements close to the laser-line are performed.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `long-pass filter`: Blocks shorter wavelengths and transmits longer wavelengths.
- `short-pass filter`: Blocks longer wavelengths and transmits shorter wavelengths.
- `notch filter`: Blocks a narrow wavelength band while transmitting others.
- `reflection filter`: Reflects certain wavelength ranges instead of absorbing them.
- `neutral density filter`: Reduces light intensity uniformly across all wavelengths.

intended_use: (optional) *NX_CHAR*

Type of laser-line filter used to suppress the laser, if measurements close to the laser-line are performed.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `laser line cleanup`
- `raylight line removal`

- `spectral filtering`
- `intensity manipulation`

filter_characteristics: (optional) *NXdata* <=

Properties of the spectral filter such as wavelength dependent transmission or reflectivity.

@characteristics_type: (optional) *NX_CHAR*

Which property is used to form the spectral properties of light, i.e. transmission or reflection properties.

Any of these values: `transmission` | `reflection`

device_information: (optional) *NXfabrication* <=

BEAM_TRANSFER_MATRIX_TABLE: (optional) *NXbeam_transfer_matrix_table*

Allows description of beam properties via matrices, which relate ingoing with outgoing beam properties.

sample_stage: (optional) *NXmanipulator*

Sample stage (or manipulator) for positioning of the sample. This should only contain the spatial orientation of movement.

stage_type: (optional) *NX_CHAR*

Specify the type of the sample stage.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `manual stage`
- `scanning stage`
- `liquid stage`
- `gas cell`
- `cryostat`
- `heater`

beam_sample_relation: (optional) *NX_CHAR*

Description of relation of the beam with the sample. How does the sample hit the beam, e.g. 'center of sample, long edge parallel to the plane of incidence'. This is redundant if a full orientation description is done via the stage's "transformations" entry.

transformations: (optional) *NXtransformations* <=

This allows a description of the stages relation or orientation and position with respect to the sample or beam, if an laboratory or an stage coordinate system is defined.

device_information: (optional) *NXfabrication* <=

temperature_sensor: (recommended) *NXsensor* <=

name: (recommended) *NX_CHAR* <=

measurement: (required) *NX_CHAR* <=

Obligatory value: `temperature`

type: (optional) *NX_CHAR* <=

value: (required) *NX_FLOAT* <=

device_information: (optional) *NXfabrication* <=

temp_control_TYPE: (optional) *NXactuator* <=

Type of control for the sample temperature. Replace TYPE by "cryostat" or "heater" to specify it.

name: (recommended) *NX_CHAR* <=

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `temperature`

cooler_or_heater: (recommended) *NX_CHAR*

Any of these values: `cooler | heater`

type: (optional) *NX_CHAR* <=

Hardware used for actuation, i.e. laser, gas lamp, filament, resistive

PID_CONTROLLER: (recommended) *NXpid_controller* <=

setpoint: (recommended) *NX_FLOAT* <=

device_information: (optional) *NXfabrication* <=

device_information: (recommended) *NXfabrication* <=

General device information of the optical spectroscopy setup, if suitable (e.g. for a tabletop spectrometer or other non-custom build setups). For custom build setups, this may be limited to the construction year.

vendor: (recommended) *NX_CHAR* <=

model: (recommended) *NX_CHAR* <=

identifier: (recommended) *NX_CHAR* <=

construction_year: (optional) *NX_DATE_TIME*

software_TYPE: (recommended) *NXprogram*

program: (required) *NX_CHAR* <=

Commercial or otherwise defined given name of the program that was used to control any parts of the optical spectroscopy setup. The uppercase TYPE should be replaced by a specification name, i.e. "software_detector" or "software_stage" to specify the respective program or software components.

@**version**: (recommended) *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.

@**URL**: (optional) *NX_CHAR*

Description of the software by persistent resource, where the program, code, script etc. can be found.

instrument_calibration_DEVICE: (recommended) *NXcalibration*

Pre-calibration of an arbitrary device of the instrumental setup, which has the name DEVICE. You can specify here how, at which time by which method the calibration was done. As well the accuracy and a link to the calibration dataset.

device_path: (recommended) *NX_CHAR*

Path to the device, which was calibrated. Example: entry/instrument/DEVICE

calibration_status: (recommended) *NX_CHAR*

Was a calibration performed? If yes, when was it done? If the calibration time is provided, it should be specified in calibration_time.

Any of these values:

- `calibration time provided`
- `no calibration`
- `within 1 hour`
- `within 1 day`
- `within 1 week`

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_accuracy: (optional) *NXdata* <=

Provide data about the determined accuracy of the device, this may may be a single value or a dataset like wavelength error vs. wavelength etc.

DATA: (optional) *NXdata* <=

Generic data which does not fit to the 'NX_FLOAT' fields in NXproces. This can be for example the instrument response function.

wavelength_resolution: (optional) *NXresolution*

The overall resolution of the optical instrument.

physical_quantity: (required) *NX_CHAR* <=

Obligatory value: `wavelength`

type: (recommended) *NX_CHAR* <=

resolution: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=

Minimum distinguishable wavelength separation of peaks in spectra.

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.

name: (required) *NX_CHAR* <=

sample_id: (recommended) *NX_CHAR*

Locally unique ID of the sample, used in the research institute or group.

physical_form: (recommended) *NX_CHAR* <=

State the form of the sample, examples are: thin film, single crystal, poly crystal, amorphous, single layer, multi layer, liquid, gas, pellet, powder. Generic properties of liquids or gases see NXsample properties.

description: (optional) *NX_CHAR* <=

Free text description of the sample.

chemical_formula: (recommended) *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: (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'.

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 timestamp when the measured specimen surface was actively prepared.

thickness: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

(Measured) sample thickness.

The information is recorded to qualify if the light used was likely able to shine through the sample.

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.

thickness_determination: (optional) *NX_CHAR*

If a thickness if given, please specify how this thickness was estimated or determined.

layer_structure: (optional) *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.

sample_orientation: (optional) *NX_CHAR*

Specify the sample orientation, how is its sample normal oriented relative in the laboratory reference frame, incident beam reference frame.

substrate: (recommended) *NX_CHAR*

If the sample is grown or fixed on a substrate, specify this here by a free text description.

history: (recommended) *NXhistory* <=

A set of activities that occurred to the sample prior to/during the experiment.

temperature_env: (recommended) *NXenvironment* <=

Sample temperature (either controlled or just measured).

temperature_nominal: (optional) *NX_CHAR*

If no sensor was available for the determination of temperature, selected a nominal value which represents approximately the situation of sample temperature.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `room temperature`
- `liquid helium temperature`
- `liquid nitrogen temperature`

temperature_sensor: (recommended) *NXsensor* <=

Temperature sensor measuring the sample temperature. This should be a link to /entry/instrument/manipulator/temperature_sensor.

sample_heater: (optional) *NXactuator* <=

Device to heat the sample. This should be a link to /entry/instrument/manipulator/sample_heater.

sample_cooler: (optional) *NXactuator* <=

Device for cooling the sample (Cryostat, Airflow cooler, etc.). This should be a link to /entry/instrument/manipulator/cryostat.

ENVIRONMENT: (optional) *NXenvironment* <=

Arbitrary sample property which may be varied during the experiment and controlled by a device. Examples are pressure, voltage, magnetic field etc. Similar to the temperature description of the sample.

sample_medium: (recommended) *NX_CHAR*

Medium, in which the sample is placed.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `air`
- `vacuum`
- `inert atmosphere`
- `oxidising atmosphere`
- `reducing atmosphere`

- `sealed can`
- `water`

sample_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.

DATA: (required) *NXdata* <=

Here generic types of data may be saved. This may refer to data derived from single or multiple raw measurements (i.e. several intensities are evaluated for different parameters: ellipsometry -> psi and delta) - i.e. non-raw data. As well plottable data may be stored/linked here, which provides the most suitable representation of the data (for the respective community).

You may provide multiple instances of NXdata

@axes: (required) *NX_CHAR* <=

Spectrum, i.e. x-axis of the data (e.g. wavelength, energy etc.)

@signal: (required) *NX_CHAR* <=

Spectrum, i.e. y-axis of the data (e.g. counts, intensity)

measurement_data_calibration_TYPE: (recommended) *NXprocess* <=

wavelength_calibration: (optional) *NXcalibration*

calibrated_axis: (recommended) *NX_FLOAT* <=

Calibrated wavelength axis.

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) *NXprogram*

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 <=*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXoptical_spectroscopy/ENTRY-group*
- */NXoptical_spectroscopy/ENTRY/beam_ref_frame-group*
- */NXoptical_spectroscopy/ENTRY/beam_ref_frame/depends_on-field*
- */NXoptical_spectroscopy/ENTRY/beam_ref_frame/TRANSFORMATIONS-group*
- */NXoptical_spectroscopy/ENTRY/DATA-group*
- */NXoptical_spectroscopy/ENTRY/DATA@axes-attribute*
- */NXoptical_spectroscopy/ENTRY/DATA@signal-attribute*
- */NXoptical_spectroscopy/ENTRY/definition-field*
- */NXoptical_spectroscopy/ENTRY/definition@URL-attribute*
- */NXoptical_spectroscopy/ENTRY/definition@version-attribute*
- */NXoptical_spectroscopy/ENTRY/derived_parameters-group*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/ANALYSIS_program-group*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/ANALYSIS_program/program-field*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/ANALYSIS_program/program@version-attribute*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/depolarization-field*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/jones_quality_factor-field*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/reflectivity-field*
- */NXoptical_spectroscopy/ENTRY/derived_parameters/transmittance-field*
- */NXoptical_spectroscopy/ENTRY/end_time-field*
- */NXoptical_spectroscopy/ENTRY/experiment_description-field*
- */NXoptical_spectroscopy/ENTRY/experiment_sub_type-field*
- */NXoptical_spectroscopy/ENTRY/experiment_type-field*
- */NXoptical_spectroscopy/ENTRY/identifier_experiment-field*
- */NXoptical_spectroscopy/ENTRY/INSTRUMENT-group*
- */NXoptical_spectroscopy/ENTRY/INSTRUMENT/angle_of_detection-field*
- */NXoptical_spectroscopy/ENTRY/INSTRUMENT/angle_of_in_plane_sample_rotation-field*
- */NXoptical_spectroscopy/ENTRY/INSTRUMENT/angle_of_incidence-field*
- */NXoptical_spectroscopy/ENTRY/INSTRUMENT/angle_of_incident_and_detection_beam-field*
- */NXoptical_spectroscopy/ENTRY/INSTRUMENT/angle_reference_frame-field*

- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/BEAM_TRANSFER_MATRIX_TABLE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/associated_source-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/beam_polarization_type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/extent-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/incident_polarization-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/incident_wavelength-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/incident_wavelength_spread-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/linear_beam_sample_polarization-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/beam_TYPE/parameter_reliability-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/chi-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/COMPONENT-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/additional_detector_hardware-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/detector_channel_type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/detector_type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/raw_data-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/raw_data/raw-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/detector_TYPE/raw_data@signal-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/device_information/construction_year-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/device_information/identifier-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/device_information/model-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/device_information/vendor-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/azimuth-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/azimuth@depends_on-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/azimuth@transformation_type-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/azimuth@vector-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/polar-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/polar@depends_on-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/polar@transformation_type-attribute

- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/polar@vector-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/generic_beam_sample_angle_TYPE/type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/instrument_calibration_DEVICE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/instrument_calibration_DEVICE/calibration_accuracy-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/instrument_calibration_DEVICE/calibration_status-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/instrument_calibration_DEVICE/calibration_time-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/instrument_calibration_DEVICE/DATA-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/instrument_calibration_DEVICE/device_path-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/lateral_focal_point_offset-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/MONOCHROMATOR-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/MONOCHROMATOR/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/omega-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/OPTICAL_LENS-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/OPTICAL_LENS/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/OPTICAL_LENS/type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/OPTICAL_WINDOW-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/phi-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/polfilter_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/polfilter_TYPE/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/polfilter_TYPE/filter_mechanism-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/polfilter_TYPE/specific_polarization_filter_type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/sample_stage-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/sample_stage/beam_sample_relation-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/sample_stage/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/sample_stage/stage_type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/sample_stage/transformations-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/software_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/software_TYPE/program-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/software_TYPE/program@URL-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/software_TYPE/program@version-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/source_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/source_TYPE/associated_beam-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/source_TYPE/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/source_TYPE/name-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/source_TYPE/standard-field

- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/source_TYPE/type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/spectralfilter_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/spectralfilter_TYPE/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/spectralfilter_TYPE/filter_characteristics-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/spectralfilter_TYPE/filter_characteristics@characteristics_type-attribute
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/spectralfilter_TYPE/filter_type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/spectralfilter_TYPE/intended_use-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/cooler_or_heater-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/name-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/physical_quantity-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/PID_CONTROLLER-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/PID_CONTROLLER/setpoint-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temp_control_TYPE/type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temperature_sensor-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temperature_sensor/device_information-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temperature_sensor/measurement-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temperature_sensor/name-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temperature_sensor/type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/temperature_sensor/value-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/twotheta-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/wavelength_resolution-group
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/wavelength_resolution/physical_quantity-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/wavelength_resolution/resolution-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/wavelength_resolution/type-field
- /NXoptical_spectroscopy/ENTRY/INSTRUMENT/WAVEPLATE-group
- /NXoptical_spectroscopy/ENTRY/measurement_data_calibration_TYPE-group
- /NXoptical_spectroscopy/ENTRY/measurement_data_calibration_TYPE/wavelength_calibration-group
- /NXoptical_spectroscopy/ENTRY/measurement_data_calibration_TYPE/wavelength_calibration/calibrated_axis-field
- /NXoptical_spectroscopy/ENTRY/SAMPLE-group
- /NXoptical_spectroscopy/ENTRY/SAMPLE/atom_types-field
- /NXoptical_spectroscopy/ENTRY/SAMPLE/chemical_formula-field
- /NXoptical_spectroscopy/ENTRY/SAMPLE/description-field
- /NXoptical_spectroscopy/ENTRY/SAMPLE/ENVIRONMENT-group

- */NXoptical_spectroscopy/ENTRY/SAMPLE/ENVIRONMENT/sample_medium-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/ENVIRONMENT/sample_medium_refractive_indices-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/history-group*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/layer_structure-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/name-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/physical_form-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/preparation_date-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/sample_id-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/sample_orientation-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/substrate-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/temperature_env-group*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/temperature_env/sample_cooler-group*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/temperature_env/sample_heater-group*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/temperature_env/temperature_nominal-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/temperature_env/temperature_sensor-group*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/thickness-field*
- */NXoptical_spectroscopy/ENTRY/SAMPLE/thickness_determination-field*
- */NXoptical_spectroscopy/ENTRY/sample_normal_ref_frame-group*
- */NXoptical_spectroscopy/ENTRY/sample_normal_ref_frame/depends_on-field*
- */NXoptical_spectroscopy/ENTRY/sample_normal_ref_frame/TRANSFORMATIONS-group*
- */NXoptical_spectroscopy/ENTRY/start_time-field*
- */NXoptical_spectroscopy/ENTRY/title-field*
- */NXoptical_spectroscopy/ENTRY/USER-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXoptical_spectroscopy.nxdl.xml

NXraman

Status:

application definition, extends *NXoptical_spectroscopy*

Description:

An application definition for Raman spectroscopy experiments.

This application definition supports a wide range of Raman spectroscopy experiments. These may be as simple as acquiring a single Raman spectrum from spontaneous Raman scattering, or as complex as Raman imaging with a Raman spectrometer. The scope also includes surface- and tip-enhanced Raman techniques, X-ray Raman scattering, resonant Raman scattering, and multidimensional Raman spectra collected under varying conditions such as temperature, pressure, or electric field.

The application definition comprises two main components:

1. Structures defined in NXoptical_spectroscopy: * Instrument configuration and data calibration * Sensors monitoring sample or beam conditions

2. Structures specified and extended in NXraman: * Description of the experiment type * Metadata and configuration of the optical setup (e.g., source, monochromator, detector, waveplate, lens) * Detailed description of beam properties and their interaction with the sample * Sample-specific information

Information on Raman spectroscopy are provided in:

General

- Lewis, Ian R.; Edwards, Howell G. M. Handbook of Raman Spectroscopy ISBN 0-8247-0557-2

Raman scattering selection rules

- Dresselhaus, M. S.; Dresselhaus, G.; Jorio, A. Group Theory - Application to the Physics ofCondensed Matter ISBN 3540328971

Semiconductors

- Manuel Cardona Light Scattering in Solids I eBook ISBN: 978-3-540-37568-5 DOI: https://doi.org/10.1007/978-3-540-37568-5
- Manuel Cardona, Gernot Güntherodt Light Scattering in Solids II eBook ISBN: 978-3-540-39075-6 DOI: https://doi.org/10.1007/3-540-11380-0
- See as well other Books from the "Light Scattering in Solids" series: III: Recent Results IV: Electronic Scattering, Spin Effects, SERS, and Morphic Effects V: Superlattices and Other Microstructures VI: Recent Results, Including High-Tc Superconductivity VII: Crystal-Field and Magnetic Excitations VIII: Fullerenes, Semiconductor Surfaces, Coherent Phonons IX: Novel Materials and Techniques

Glasses, Liquids, Gasses, …

Review articles: Stimulated Raman scattering, Coherent anti-Stokes Raman scattering, Surface-enhanced Raman scattering, Tip-enhanced Raman scattering * https://doi.org/10.1186/s11671-019-3039-2

Symbols:

Variables used throughout the document, e.g. dimensions or parameters.

N_scattering_configurations: Number of scattering configurations used in the measurement. It is 1 for only parallel polarization measurement, 2 for parallel and cross polarization measurement or larger, if i.e. the incident and scattered photon direction is varied.

Groups cited:
NXbeam, *NXentry*, *NXinstrument*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

An application definition for Raman spectroscopy.

Obligatory value: `NXraman`

@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.

title: (recommended) *NX_CHAR* <=

experiment_type: (required) *NX_CHAR* <=

Specify the type of the optical experiment.

You may specify fundamental characteristics or properties in the experimental subtype.

Obligatory value: `Raman spectroscopy`

raman_experiment_type: (required) *NX_CHAR*

Specify the type of Raman experiment.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `in situ Raman spectroscopy`
- `resonant Raman spectroscopy`
- `non-resonant Raman spectroscopy`
- `Raman imaging`
- `tip-enhanced Raman spectroscopy (TERS)`
- `surface-enhanced Raman spectroscopy (SERS)`
- `surface plasmon polariton enhanced Raman scattering (SPPERS)`
- `hyper Raman spectroscopy (HRS)`
- `stimulated Raman spectroscopy (SRS)`
- `inverse Raman spectroscopy (IRS)`
- `coherent anti-Stokes Raman spectroscopy (CARS)`

INSTRUMENT: (required) *NXinstrument* <=

Metadata of the setup, its optical elements and physical properties which defines the Raman measurement.

scattering_configuration: (required) *NX_CHAR*

Scattering configuration as defined by the porto notation by three states, which are orthogonal to each other. Example: z(xx)z for parallel polarized backscattering configuration.

See: https://www.cryst.ehu.es/cgi-bin/cryst/programs/nph-doc-raman

A(BC)D

A = The propagation direction of the incident light (k_i) B = The polarization direction of the incident light (E_i) C = The polarization direction of the scattered light (E_s) D = The propagation direction of the scattered light (k_s)

An orthogonal base is assumed. Linear polarized light is displayed by e.g. “x”,”y” or “z” Unpolarized light is displayed by “.” For non-orthogonal vectors, use the attribute porto_notation_vectors.

@**porto_notation_vectors**: (recommended) *NX_NUMBER*

Scattering configuration as defined by the porto notation given by respective vectors.

Vectors in the porto notation are defined as for A, B, C, D above. Linear light polarization is assumed.

beam_incident: (required) *NXbeam* <=

Beam which is incident to the sample.

wavelength: (required) *NX_NUMBER*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXraman/ENTRY-group*
- */NXraman/ENTRY/definition-field*
- */NXraman/ENTRY/definition@URL-attribute*
- */NXraman/ENTRY/definition@version-attribute*
- */NXraman/ENTRY/experiment_type-field*
- */NXraman/ENTRY/INSTRUMENT-group*
- */NXraman/ENTRY/INSTRUMENT/beam_incident-group*
- */NXraman/ENTRY/INSTRUMENT/beam_incident/wavelength-field*
- */NXraman/ENTRY/INSTRUMENT/scattering_configuration-field*
- */NXraman/ENTRY/INSTRUMENT/scattering_configuration@porto_notation_vectors-attribute*
- */NXraman/ENTRY/raman_experiment_type-field*
- */NXraman/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXraman.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 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

NXstress

Status:

application definition, extends *NXobject*

Description:

Application definition for stress and strain analysis of crystalline material defined by the EASI-STRESS consortium.

When a crystal is loaded (applied or residual stress) its crystallographic parameters change.

Stress and strain analysis calculates deformation (strain) and the associated force (stress) from diffraction data.

This application definition essentially standardizes the result of diffraction pattern analysis from different types of diffraction experiments for the purpose of stress and strain analysis. The analysis is typically some form of diffraction peak indexing and fitting. The experiments are for example

- energy-dispersive X-ray powder diffraction
- angular-dispersive X-ray powder diffraction
- angular-dispersive neutron powder diffraction
- time-of-flight (TOF) neutron powder diffraction.

In addition, the application definition guarantees that the information about instrumental setups, measurement conditions, and data analysis workflows are described. This ensures not only the reproducability and tracability of the measured data, but also the metadata. Since not all participating beamlines or instruments can provide an input to all the NeXus fields listed in this application definition, not all of them are "required".

However, when possible and technically feasible, the instrument using the NXstress application definition is expected to provide the type of information outlined below.

Sample and detector positions can be defined with *NXtransformations*. If you don't specify the direction of gravity and the direction of the beam then the standard NeXus Coordinate System is used.

It is highly recommended that when certain parameters or values are the same for all the measurements (acquistions) in the same file, they are stored only in one location and then linked in the other instances. For example, if during an acquisition all

instrumental parameters but one stay the same and only the sample table moves in one direction (e.g. Xtranslation), then all the static instrumental parameters should be saved just once (e.g. in just one NXentry or in a *Shared_Information group*) and their values linked to every *instrument group* under all the other acquisitions. The value for the variable that changes, Xtranslation in this example, is suggested to be saved only at every instrument group under each acquistion but not in the *Shared_Information group*.

It is not always necessary to link each field. In case all the fields with an entire group are the same, the entire group can be linked.

Symbols:

n_X: Number of diffractogram channels.

n_D: Number of diffractograms. For example the number of energy-dispersive detectors or the number of azimuthal sections in an area detector.

n_Peaks: Number of reflections.

x_Unit: Diffractogram X units.

y_Unit: Diffractogram Y units.

c_Unit: Converted diffractogram X units (could be the same as *x_Unit*).

n_Temp: number of temperatures

n_sField: number of values in applied stress field

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

Groups cited:
NXbeam, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXnote*, *NXparameters*, *NXprocess*, *NXreflections*, *NXsample*, *NXsource*, *NXtransformations*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

The name of the *NXentry group(s)* can be freely chosen by the facility. The *NXentry group* can contain any form of data acquisition (e.g. a measurement point, multiple measurement points, a line scan, a mesh, all data points from one sample ...).

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXstress`

title: (optional) *NX_CHAR* <=

Extended title for the entry.

experiment_identifier: (optional) *NX_CHAR* <=

Unique identifier for the experiment as defined by the facility (e.g. DOI, proposal id, ...). At ILL, this could be, for example, `exp_1-02-286`, `exp_INDU-229`, or `exp_INTER-569`.

experiment_description: (optional) *NX_CHAR* <=

Brief summary of the experiment, including key objectives. At least one of the following information should be provided:

- `energy-dispersive X-ray powder diffraction`
- `angular-dispersive X-ray powder diffraction`
- `angular-dispersive neutron powder diffraction`
- `time-of-flight (TOF) neutron powder diffraction`

start_time: (required) *NX_DATE_TIME* <=

The starting time(s) of measurement(s) which can be provided in form of a list if multiple measurements are included in the same NXentry.

end_time: (required) *NX_DATE_TIME* <=

The end time(s) of measurement(s) which can be provided in form of a list if multiple measurements are included in the same NXentry.

collection_identifier: (optional) *NX_CHAR* <=

User or Data Acquisition defined identifier from which the content of this application definition is derived. This can be freely chosen by the user or the instrument scientist and could be, for example, `05_DA_650_AX_B3P5`, `SENB-14`, `Quartz`,....

collection_description: (optional) *NX_CHAR* <=

Brief summary of the collection, including grouping criteria. The information provided in this field can highlight, for example, the measurement setup or information about experimental conditions.

processing_type: (required) *NX_CHAR*

Describes the way strain ε can be calculated from the *center* peak parameter.

Any of these values:

- `two-theta`: $\varepsilon = \frac{sin(\theta_0)}{sin(\theta)} - 1$
- `energy`: $\varepsilon = \frac{\mathrm{E_0}}{\mathrm{E}} - 1$
- `d-spacing`: $\varepsilon = \frac{\mathrm{d}}{\mathrm{d_0}} - 1$
- `time-of-flight`: $\varepsilon = \frac{\mathrm{TOF}}{\mathrm{TOF_0}} - 1$
- `sin2psi`: A description of the $\sin^2\psi$ method can be found in the literature. Two examples are Fitzpatrick et al. 2005 and DIN ISO 15305:2009-01.

measurement_direction: (optional) *NX_CHAR*

Describes the specific measurement direction covered by the data in this file.

Any of these values:

- `radial`
- `longitudinal`
- `normal`
- `tangential`
- `multiple`

experiment_responsible: (optional) *NXuser* <=

Information about the person who performed the experiment.

name: (optional) *NX_CHAR* <=

role: (optional) *NX_CHAR* <=

Role of user responsible for this entry. Suggested roes are, for example, `local contact`, `beamline_scientist`, `post_doc`,...

instrument: (required) *NXinstrument* <=

name: (required) *NX_CHAR* <=

Name of the diffractometer, instrument, or beamline used for the experiment. This could be, for example, *Strain Analyser for Large and Small scale engineering Applications*.

@short_name: (optional) *NX_CHAR* <=

Short name for the instrument, perhaps the acronym, which would be for the the example above `SALSA`.

CALIBRATION: (optional) *NXnote* <=

This group contains information about the geometry and/or efficiency measurement(s).

calibration_type: (optional) *NX_CHAR*

Describe the type of calibration.

file_name: (optional) *NX_CHAR* <=

File name(s) and/or path(s) (within file(s)) containing data from the last calibration(s). This can be an array.

data: (optional) *NX_BINARY* <=

Calibration file content.

type: (optional) *NX_CHAR* <=

Mime content type of calibration *data* field e.g. text/plain, application/json,...

author: (optional) *NX_CHAR* <=

Author or creator of the calibration.

date: (optional) *NX_DATE_TIME* <=

Date calibration was created/added

SOURCE: (required) *NXsource* <=

type: (required) *NX_CHAR* <=

Type of radiation source

Any of these values:

- `Spallation Neutron Source`
- `Pulsed Reactor Neutron Source`
- `Reactor Neutron Source`
- `Synchrotron X-ray Source`
- `Rotating Anode X-ray`
- `Fixed Tube X-ray`
- `Metal Jet X-ray`

probe: (required) *NX_CHAR* <=

Type of radiation probe

Any of these values: `neutron` | `X-ray`

DETECTOR: (required) *NXdetector* <=

Zero or more of these groups describe the detectors used in the experiment.

description: (optional) *NX_CHAR* <=

name/manufacturer/model/etc. information

type: (required) *NX_CHAR* <=

Description of type such as ^{3}He gas cylinder, ^{3}He PSD, scintillator, fission chamber, proportion counter, ion chamber, CCD, pixel, image plate, CMOS, ...

distance: (optional) *NX_NUMBER* (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.

efficiency: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_DIMENSIONLESS*}

efficiency of the detector

wavelength: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [i, j]) {units=*NX_WAVELENGTH*}

This field can be two things:

1. For a pixel detector it provides the nominal wavelength for which the detector has been calibrated.

2. For other detectors this field has to be seen together with the efficiency field above. For some detectors, the efficiency is wavelength dependent. Thus this field provides the wavelength axis for the efficiency field. In this use case, the efficiency and wavelength arrays must have the same dimensionality.

dead_time: (optional) *NX_FLOAT* (Rank: 3, Dimensions: [nP, i, j]) {units=*NX_TIME*} <=

Detector dead time

count_time: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [nP]) {units=*NX_TIME*} <=

Elapsed actual counting time

depends_on: (optional) *NX_CHAR* <=

The axis on which the detector position depends may be stored anywhere, but is normally stored in the *NXtransformations group* within the *NXdetector group*.

TRANSFORMATIONS: (optional) *NXtransformations* <=

This is the recommended location for detector goniometer and other related axes.

beam_intensity_profile: (required) *NXbeam* <=

Defines the dimensions of the beam profile used for probing the sample which corresponds to or can be used to determine the instrumental gauge volume. A description of the subsequent fields can be found in the folowing figure. The term "primary" in the subsequent fields refers to the beam path between the sample and the source. The term "secondary" refers to the beam path between the sample and the detector(s).

beam_evaluation: (optional) *NX_CHAR*

If the beam profile was measured, the filename(s) of the measurement can be specified here.

primary_vertical_type: (optional) *NX_CHAR*

Defines the last device right in front of the sample used to shape the beam. This could be, for example, a *(radial) collimator* or a *slit*.

primary_vertical_source_width: (optional) *NX_NUMBER*

Defines the primary beam size intensity profile on the side closer to the source in the vertical direction.

primary_vertical_sample_width: (optional) *NX_NUMBER*

Defines the primary beam size intensity profile on the side closer to the sample in the vertical direction.

Fig. 12: Some examples for the beam intensity profile. The 1D description of the beam profile on the right can equally be applied for the horizontal and vertical direction for the primary and the secondary side.

primary_vertical_distance: (optional) *NX_NUMBER*

Defines the distance between the center of the gauge volume and the beam shaping device.

primary_vertical_evaluation: (optional) *NX_CHAR*

Describes how the beam intensity profile in the primary vertical direction was determined. Examples of valid entries are: `measured`, `theoretical`, `estimated`, ...

primary_horizontal_type: (optional) *NX_CHAR*

Defines the last device right in front of the sample used to shape the beam. This could be, for example, a *(radial) collimator* or a *slit*.

primary_horizontal_source_width: (optional) *NX_NUMBER*

Defines the primary beam size intensity profile on the side closer to the source in the horizontal direction.

primary_horizontal_sample_width: (optional) *NX_NUMBER*

Defines the primary beam size intensity profile on the side closer to the sample in the horizontal direction.

primary_horizontal_distance: (optional) *NX_NUMBER*

Defines the distance between the center of the gauge volume and the beam shaping device.

primary_horizontal_evaluation: (optional) *NX_CHAR*

Describes how the beam intensity profile in the primary horizontal direction was determined. Examples of valid entries are: `measured`, `theoretical`, `estimated`, ...

secondary_horizontal_type: (optional) *NX_CHAR*

Defines the last device right in front of the sample used to shape the beam. This could be, for example, a *(radial) collimator* or a *slit*.

secondary_horizontal_detector_width: (optional) *NX_NUMBER*

Defines the secondary beam size intensity profile on the side closer to the detector in the horizontal direction.

secondary_horizontal_sample_width: (optional) *NX_NUMBER*

Defines the secondary beam size intensity profile on the side closer to the sample in the horizontal direction.

secondary_horizontal_distance: (optional) *NX_NUMBER*

Defines the distance between the center of the gauge volume and the beam shaping device.

secondary_horizontal_evaluation: (optional) *NX_CHAR*

Describes how the beam intensity profile in the secondary horizontal direction was determined. Examples of valid entries are: `measured`, `theoretical`, `estimated`, ...

incident_energy: (optional) *NX_FLOAT* {units=*NX_ENERGY*} <=

Incident energy mostly useful for monochromatic beams.

incident_wavelength: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*} <=

Incident wavelength mostly useful for monochromatic beams.

SAMPLE_DESCRIPTION: (required) *NXsample* <=

This is the recommended location for describing parameters associated with the sample.

name: (required) *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

stress_field: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_sField]) {units=*NX_ANY*} <=

Applied external stress field

@**direction**: (required) *NX_CHAR* <=

Any of these values: `x | y | z`

depends_on: (optional) *NX_CHAR* <=

The axis on which the sample position depends may be stored anywhere, but is normally stored in the NXtransformations group within the NXsample group.

gauge_volume: (optional) *NXparameters* <=

Fig. 13: Gauge volume parameters and coordinate system.

The gauge volume can be described with the following parameters:

a: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Length of the first diagonal.

b: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Length of the second diagonal normal to *x*.

c: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Height of the gauge volume.

depends_on: (optional) *NX_CHAR*

In the local coordinate system, the beam is aligned along the X-axis, and the Z-axis is oriented in the opposite direction of gravity. The origin is the center to the gauge volume.

TRANSFORMATIONS: (optional) *NXtransformations*

The last field typically depends on the first field of the *sample transformations*.

TRANSFORMATIONS: (optional) *NXtransformations* <=

This is the recommended location for sample goniometer and other related axes.

FIT: (required) *NXprocess* <=

Zero or more groups to describe the data processing steps to obtain the content of this application definition.

raw_data_file: (required) *NX_CHAR*

The raw data file name(s) used during the data reduction process. This can be a list.

date: (required) *NX_DATE_TIME* <=

Date when the raw data was reduced and the data in the *NXstress* file format generated.

program: (required) *NX_CHAR* <=

Software package used to perform data reduction including the version number or release date.

integration_type: (optional) *NX_CHAR*

Describes how the data was integrated.

bins: (optional) *NX_CHAR*

Describes the type of binning used during data reduction.

fit_type: (optional) *NX_CHAR*

Describes how the fitting of the peaks was done. For example, single peak fit, multiple peak fit, Pawley refinement, Rietveld refinement, ...

fit_range: (optional) *NX_CHAR*

Describes the data range used for peak fitting.

goodness_of_fit: (optional) *NX_CHAR*

Type and value describing the goodness of fit. For example, Rw 0.23.

normalization: (optional) *NX_CHAR*

Describes whether the data was normalized and if so , how. Examples of valid entries are: `None`, `time`, `primary monitor`, `detector`, ...

data_reduction_responsible: (optional) *NXuser*

Information about the person who performed the data reduction.

name: (optional) *NX_CHAR* <=

role: (optional) *NX_CHAR* <=

Role of user responsible for this entry. Suggested roles are, for example, `local contact`, `beamline_scientist`, `post_doc`,...

DESCRIPTION: (required) *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 a simple text.

The name will be numbered to allow for ordering of steps.

peak_parameters: (required) *NXparameters* <=

This group contains all diffraction peak fit parameters. This information is not required for stress and strain calculations.

title: (required) *NX_CHAR*

Diffraction peak profile.

Any of these values:

- `gaussian`
- `lorentzian`
- `voigt`
- `pseudo-voigt`
- `split pseudo-voigt`
- `pearson VII`

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Diffraction peak area (not including the background) in *y_Unit* units.

@units: (required) *NX_CHAR*

Specify the *y_Unit* units

area_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *area*

center: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Diffraction peak position in *x_Unit* units.

@units: (required) *NX_CHAR*

Specify the *x_Unit* units

center_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *center*

height: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Diffraction peak height (not including the background) in *y_Unit* units.

@units: (required) *NX_CHAR*

Specify the *y_Unit* units

height_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *height*

fwhm: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Diffraction peak full width at half maximum in *x_Unit* units.

@units: (required) *NX_CHAR*

Specify the *x_Unit* units

fwhm_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *fwhm*

fwhm_left: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Left-side FWHM for split profiles in *x_Unit* units.

@units: (required) *NX_CHAR*

Specify the *x_Unit* units

fwhm_left_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *fwhm_left*

fwhm_right: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Right-side FWHM for split profiles in *x_Unit* units.

@units: (required) *NX_CHAR*

Specify the *x_Unit* units

fwhm_right_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *fwhm_right*

form_factor: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

- Voigt or Pseudo-Voigt: Lorentzian fraction
- Pearson VII: decay parameter
- Other profiles: not applicable

form_factor_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error value(s) asscociated with *form_factor*

azimuth: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANGLE*}

Angle that defines the position of the integrated sector in the diffraction cone for angular-dispersive diffraction or the position of the detector for energy-dispersive diffraction.

background_parameters: (required) *NXparameters* <=

This group contains all background fit parameters. This information is not required for stress and strain calculations.

title: (required) *NX_CHAR*

Diffraction background profile. Required when background parameter fields are present. Some example values with equations are shown below:

- `manual` : No equations nor variables needed to describe this background.
- `linear` : $background = A0 + A1 \cdot x$
- `5-degree polynomial` : $background = A0 + A1 \cdot x + A2 \cdot \mathrm{x}^2 + A3 \cdot \mathrm{x}^3 + A4 \cdot \mathrm{x}^4 + A5 \cdot \mathrm{x}^5$

- `shape function plus polynomial` : A shape function is not a mathematical function, it contains a manual background obtained from a fit and a polynomial part. This allows to adapt and modify the fit for subsequent measurements in the same measurement campaign. The function describing it is the following: $background = as + b \cdot SHAPE(x - o)$ Where SHAPE is the name of the variable used to describe the background value at the position x. x can be e.g. the scattering angle 2θ in degrees.

A: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Background parameter(s). For example a second-degree polynomial will have fields `A0`, `A1` and `A2`.

as: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Background parameter *constant* for SHAPE function.

as_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error associated with background parameter *constant* for SHAPE function.

b: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Background parameter *amplitude* for SHAPE function.

b_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error associated with background parameter *amplitude* for SHAPE function.

o: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Background parameter *offset* for SHAPE function.

o_errors: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Error associated with background parameter *offset* for SHAPE function.

background_area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

The background area in *y_Unit* units, integrated over a confidence interval around the center (*0.95* by default).

@units: (required) *NX_CHAR*

Specify the *y_Unit* units

background_area_interval: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Confidence interval from which the background counts are integrated. For example *0.95* means that the background is integrated over the range in which the integrated peak area is 95% of the total peak area.

DIFFRACTOGRAM: (required) *NXdata* <=

Diffractogram with fit results in *peak_parameters* and *background_parameters*. This information is not required for stress and strain calculations.

@**axes**: (required) *NX_CHAR* <=

List of the one to two axes field name(s) to be used by default. The axes are further described in the fields DAXIS and XAXIS.

@**signal**: (required) *NX_CHAR* <=

Default field name to be plotted.

Obligatory value: `diffractogram`

@**auxiliary_signals**: (required) *NX_CHAR* <=

List of additional field names to be plotted. This could be e.g. fit, background, residuals, …

DAXIS: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_D])

One or more fields that contain the values for the **n_D** dimension. For example the azimuthal positions of different energy-dispersive detectors or the average azimuth of different azimuthal sections on an area detector.

XAXIS: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_X]) {units=*NX_ANY*} <=

One or more fields that contain the values for the **n_X** dimension in *x_Unit* units. For example: MCA channels, scattering angle 2θ in degrees, scattering vector length q in nm^{-1}, …

@**units**: (required) *NX_CHAR*

Specify the *x_Unit* units

diffractogram: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_D, n_X]) {units=*NX_ANY*}

Diffractogram counts in *y_Unit* units (default signal)

@**interpretation**: (required) *NX_CHAR*

Obligatory value: `spectrum`

@**units**: (required) *NX_CHAR*

Specify the *y_Unit* units

diffractogram_errors: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_D, n_X]) {units=*NX_ANY*} <=

Diffractogram counts error in *y_Unit* units (default signal)

@**interpretation**: (required) *NX_CHAR*

Obligatory value: `spectrum`

@**units**: (required) *NX_CHAR*

Specify the *y_Unit* units

fit: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_D, n_X]) <=

Diffractogram fit counts (auxiliary signal).

@interpretation: (required) *NX_CHAR*

Obligatory value: `spectrum`

fit_errors: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_D, n_X]) <=

Diffractogram fit counts error (auxiliary signal).

background: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_D, n_X]) <=

In case the diffraction background was manually determined. Diffractogram background counts (auxiliary signal).

@interpretation: (required) *NX_CHAR*

Obligatory value: `spectrum`

residuals: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_D, n_X])

Difference between diffractogram and fit (auxiliary signal).

@interpretation: (required) *NX_CHAR*

Obligatory value: `spectrum`

NOTES: (optional) *NXnote* <=

User description of the data acquisitions. A description of data analysis goes in the *fit descriptions*.

peaks: (required) *NXreflections*

This group contains all diffraction peak parameters that could be needed for stress and strain calculations. These parameters are derived from *peak_parameters* and additional metadata.

h: (required) *NX_INT* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_UNITLESS*}

First Miller index.

k: (required) *NX_INT* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_UNITLESS*}

Second Miller index.

l: (required) *NX_INT* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_UNITLESS*}

Third Miller index.

lattice: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_Peaks])

Crystal lattice systems (*cubic*, *hexagonal*, …)

space_group: (optional) *NX_CHAR* (Rank: 1, Dimensions: [n_Peaks])

Crystallographic space group $(Fm\bar{3}m, Im\bar{3}m, ...)$

phase_name: (required) *NX_CHAR* (Rank: 1, Dimensions: [n_Peaks])

Name of the crystallographic phase (hematite, goethite, α-Al_2O_3, …).

qx: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

First component of the *normalized* scattering vector *Q* in the sample reference frame. The sample reference frame is defined by the *sample transformations*.

qy: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Second component of the *normalized* scattering vector Q in the sample reference frame. The sample reference frame is defined by the *sample transformations*.

qz: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_DIMENSIONLESS*}

Third component of the *normalized* scattering vector Q in the sample reference frame. The sample reference frame is defined by the *sample transformations*.

center: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*}

Diffraction peak position in *c_Unit* units.

@**units**: (required) *NX_CHAR*

Specify the *c_Unit* units (see *center_type*)

Any of these values:

- `degrees`: two-theta
- `keV`: energy
- `1/angstrom`: momentum-transfer
- `angstrom`: d-spacing
- `microseconds`: time-of-flight
- `''`: channel (dimensionless)

center_errors: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_ANY*} <=

Uncentrainties on *center* in *c_Unit* units.

@**units**: (required) *NX_CHAR*

Specify the *c_Unit* units (see *center_type*)

Any of these values:

- `degrees`: two-theta
- `keV`: energy
- `1/angstrom`: momentum-transfer
- `angstrom`: d-spacing
- `microseconds`: time-of-flight
- `''`: channel (dimensionless)

center_type: (required) *NX_CHAR*

The space in which *center* is defined. It defines the *c_Unit* as follows

- if *center_type="two-theta"* then *c_Unit* must have the angle unit *degrees*
- if *center_type="energy"* then *c_Unit* must have the unit *keV*
- if *center_type="momentum-transfer"* then *c_Unit* must have the unit $^{-1}$

- if *center_type="d-spacing"* then *c_Unit* must have the unit
- if *center_type="channel"* then *c_Unit* must be *dimensioness*
- if *center_type="time-of-flight"* then *c_Unit* must have the unit μs

Any of these values:

- `two-theta`
- `energy`
- `momentum-transfer`
- `d-spacing`
- `channel`
- `time-of-flight`

sx: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_LENGTH*}

First component of the sample position in the sample reference frame. The sample reference frame is defined by the *sample transformations*.

sy: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_LENGTH*}

First component of the sample position in the sample reference frame. The sample reference frame is defined by the *sample transformations*.

sz: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_Peaks]) {units=*NX_LENGTH*}

First component of the sample position in the sample reference frame. The sample reference frame is defined by the *sample transformations*.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXstress/ENTRY-group*
- */NXstress/ENTRY/collection_description-field*
- */NXstress/ENTRY/collection_identifier-field*
- */NXstress/ENTRY/definition-field*
- */NXstress/ENTRY/end_time-field*
- */NXstress/ENTRY/experiment_description-field*
- */NXstress/ENTRY/experiment_identifier-field*
- */NXstress/ENTRY/experiment_responsible-group*
- */NXstress/ENTRY/experiment_responsible/name-field*
- */NXstress/ENTRY/experiment_responsible/role-field*
- */NXstress/ENTRY/FIT-group*
- */NXstress/ENTRY/FIT/background_parameters-group*
- */NXstress/ENTRY/FIT/background_parameters/A-field*

- /NXstress/ENTRY/FIT/background_parameters/as-field
- /NXstress/ENTRY/FIT/background_parameters/as_errors-field
- /NXstress/ENTRY/FIT/background_parameters/b-field
- /NXstress/ENTRY/FIT/background_parameters/b_errors-field
- /NXstress/ENTRY/FIT/background_parameters/background_area-field
- /NXstress/ENTRY/FIT/background_parameters/background_area@units-attribute
- /NXstress/ENTRY/FIT/background_parameters/background_area_interval-field
- /NXstress/ENTRY/FIT/background_parameters/o-field
- /NXstress/ENTRY/FIT/background_parameters/o_errors-field
- /NXstress/ENTRY/FIT/background_parameters/title-field
- /NXstress/ENTRY/FIT/bins-field
- /NXstress/ENTRY/FIT/data_reduction_responsible-group
- /NXstress/ENTRY/FIT/data_reduction_responsible/name-field
- /NXstress/ENTRY/FIT/data_reduction_responsible/role-field
- /NXstress/ENTRY/FIT/date-field
- /NXstress/ENTRY/FIT/DESCRIPTION-group
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM-group
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/background-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/background@interpretation-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/DAXIS-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/diffractogram-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/diffractogram@interpretation-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/diffractogram@units-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/diffractogram_errors-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/diffractogram_errors@interpretation-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/diffractogram_errors@units-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/fit-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/fit@interpretation-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/fit_errors-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/residuals-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/residuals@interpretation-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/XAXIS-field
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM/XAXIS@units-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM@auxiliary_signals-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM@axes-attribute
- /NXstress/ENTRY/FIT/DIFFRACTOGRAM@signal-attribute

- /NXstress/ENTRY/FIT/fit_range-field
- /NXstress/ENTRY/FIT/fit_type-field
- /NXstress/ENTRY/FIT/goodness_of_fit-field
- /NXstress/ENTRY/FIT/integration_type-field
- /NXstress/ENTRY/FIT/normalization-field
- /NXstress/ENTRY/FIT/peak_parameters-group
- /NXstress/ENTRY/FIT/peak_parameters/area-field
- /NXstress/ENTRY/FIT/peak_parameters/area@units-attribute
- /NXstress/ENTRY/FIT/peak_parameters/area_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/azimuth-field
- /NXstress/ENTRY/FIT/peak_parameters/center-field
- /NXstress/ENTRY/FIT/peak_parameters/center@units-attribute
- /NXstress/ENTRY/FIT/peak_parameters/center_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/form_factor-field
- /NXstress/ENTRY/FIT/peak_parameters/form_factor_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/fwhm-field
- /NXstress/ENTRY/FIT/peak_parameters/fwhm@units-attribute
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_left-field
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_left@units-attribute
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_left_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_right-field
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_right@units-attribute
- /NXstress/ENTRY/FIT/peak_parameters/fwhm_right_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/height-field
- /NXstress/ENTRY/FIT/peak_parameters/height@units-attribute
- /NXstress/ENTRY/FIT/peak_parameters/height_errors-field
- /NXstress/ENTRY/FIT/peak_parameters/title-field
- /NXstress/ENTRY/FIT/program-field
- /NXstress/ENTRY/FIT/raw_data_file-field
- /NXstress/ENTRY/instrument-group
- /NXstress/ENTRY/instrument/beam_intensity_profile-group
- /NXstress/ENTRY/instrument/beam_intensity_profile/beam_evaluation-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/incident_energy-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/incident_wavelength-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_horizontal_distance-field

- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_horizontal_evaluation-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_horizontal_sample_width-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_horizontal_source_width-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_horizontal_type-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_vertical_distance-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_vertical_evaluation-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_vertical_sample_width-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_vertical_source_width-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/primary_vertical_type-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/secondary_horizontal_detector_width-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/secondary_horizontal_distance-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/secondary_horizontal_evaluation-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/secondary_horizontal_sample_width-field
- /NXstress/ENTRY/instrument/beam_intensity_profile/secondary_horizontal_type-field
- /NXstress/ENTRY/instrument/CALIBRATION-group
- /NXstress/ENTRY/instrument/CALIBRATION/author-field
- /NXstress/ENTRY/instrument/CALIBRATION/calibration_type-field
- /NXstress/ENTRY/instrument/CALIBRATION/data-field
- /NXstress/ENTRY/instrument/CALIBRATION/date-field
- /NXstress/ENTRY/instrument/CALIBRATION/file_name-field
- /NXstress/ENTRY/instrument/CALIBRATION/type-field
- /NXstress/ENTRY/instrument/DETECTOR-group
- /NXstress/ENTRY/instrument/DETECTOR/count_time-field
- /NXstress/ENTRY/instrument/DETECTOR/dead_time-field
- /NXstress/ENTRY/instrument/DETECTOR/depends_on-field
- /NXstress/ENTRY/instrument/DETECTOR/description-field
- /NXstress/ENTRY/instrument/DETECTOR/distance-field
- /NXstress/ENTRY/instrument/DETECTOR/efficiency-field
- /NXstress/ENTRY/instrument/DETECTOR/TRANSFORMATIONS-group
- /NXstress/ENTRY/instrument/DETECTOR/type-field
- /NXstress/ENTRY/instrument/DETECTOR/wavelength-field
- /NXstress/ENTRY/instrument/name-field
- /NXstress/ENTRY/instrument/name@short_name-attribute
- /NXstress/ENTRY/instrument/SOURCE-group
- /NXstress/ENTRY/instrument/SOURCE/probe-field
- /NXstress/ENTRY/instrument/SOURCE/type-field

- */NXstress/ENTRY/measurement_direction-field*
- */NXstress/ENTRY/NOTES-group*
- */NXstress/ENTRY/peaks-group*
- */NXstress/ENTRY/peaks/center-field*
- */NXstress/ENTRY/peaks/center@units-attribute*
- */NXstress/ENTRY/peaks/center_errors-field*
- */NXstress/ENTRY/peaks/center_errors@units-attribute*
- */NXstress/ENTRY/peaks/center_type-field*
- */NXstress/ENTRY/peaks/h-field*
- */NXstress/ENTRY/peaks/k-field*
- */NXstress/ENTRY/peaks/l-field*
- */NXstress/ENTRY/peaks/lattice-field*
- */NXstress/ENTRY/peaks/phase_name-field*
- */NXstress/ENTRY/peaks/qx-field*
- */NXstress/ENTRY/peaks/qy-field*
- */NXstress/ENTRY/peaks/qz-field*
- */NXstress/ENTRY/peaks/space_group-field*
- */NXstress/ENTRY/peaks/sx-field*
- */NXstress/ENTRY/peaks/sy-field*
- */NXstress/ENTRY/peaks/sz-field*
- */NXstress/ENTRY/processing_type-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION-group*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/chemical_formula-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/depends_on-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/gauge_volume-group*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/gauge_volume/a-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/gauge_volume/b-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/gauge_volume/c-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/gauge_volume/depends_on-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/gauge_volume/TRANSFORMATIONS-group*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/name-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/stress_field-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/stress_field@direction-attribute*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/temperature-field*
- */NXstress/ENTRY/SAMPLE_DESCRIPTION/TRANSFORMATIONS-group*
- */NXstress/ENTRY/start_time-field*

- */NXstress/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXstress.nxdl.xml

NXstxm

Status:

application definition, extends *NXobject*

Description:

Application definition for a STXM instrument.

The interferometer position measurements, monochromator photon energy values and detector measurements are all treated as NXdetectors and stored within the NXinstrument group as lists of values stored in chronological order. The NXdata group then holds another version of the data in a regular 3D array (NumE by NumY by NumX, for a total of nP points in a sample image stack type scan). The former data values should be stored with a minimum loss of precision, while the latter values can be simplified and/or approximated in order to fit the constraints of a regular 3D array. 'Line scans' and 'point spectra' are just sample_image scan types with reduced dimensions in the same way as single images have reduced E dimensions compared to image 'stacks'.

Symbols:

These symbols will be used below to coordinate the shapes of the datasets.

nP: Total number of scan points

nE: Number of photon energies scanned

nX: Number of pixels in X direction

nY: Number of pixels in Y direction

detectorRank: Rank of data array provided by the detector for a single measurement

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXmonochromator*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

end_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXstxm`

INSTRUMENT: (required) *NXinstrument* <=

SOURCE: (required) *NXsource* <=

type: (required) *NX_CHAR* <=

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

monochromator: (required) *NXmonochromator* <=

energy: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) <=

DETECTOR: (required) *NXdetector* <=

data: (required) *NX_NUMBER* (Rank: 1+detectorRank, Dimensions: [nP]) <=

sample_x: (optional) *NXdetector* <=

Measurements of the sample position from the x-axis interferometer.

data: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

sample_y: (optional) *NXdetector* <=

Measurements of the sample position from the y-axis interferometer.

data: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

sample_z: (optional) *NXdetector* <=

Measurements of the sample position from the z-axis interferometer.

data: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP])

SAMPLE: (required) *NXsample* <=

rotation_angle: (required) *NX_FLOAT* <=

DATA: (required) *NXdata* <=

stxm_scan_type: (required) *NX_CHAR*

Label for typical scan types as a convenience for humans.
Each label corresponds to a specific set of axes being scanned to produce a data array of shape:

- sample point spectrum: (photon_energy,)
- sample line spectrum: (photon_energy, sample_y/sample_x)
- sample image: (sample_y, sample_x)
- sample image stack: (photon_energy, sample_y, sample_x)
- sample focus: (zoneplate_z, sample_y/sample_x)
- osa image: (osa_y, osa_x)
- osa focus: (zoneplate_z, osa_y/osa_x)
- detector image: (detector_y, detector_x)

The "generic scan" string is to be used when none of the other choices are appropriate.

Any of these values:

- `sample point spectrum`
- `sample line spectrum`
- `sample image`
- `sample image stack`
- `sample focus`

- `osa image`
- `osa focus`
- `detector image`
- `generic scan`

data: (required) *NX_NUMBER* <=

Detectors that provide more than one value per scan point should be summarised
to a single value per scan point for this array in order to simplify plotting.

Note that 'Line scans' and focus type scans measure along one spatial dimension but are not restricted to being parallel to the X or Y axes. Such scans should therefore use a single dimension for the positions along the spatial line. The 'sample_x' and 'sample_y' fields should then contain lists of the x- and y-positions and should both have the 'axis' attribute pointing to the same dimension.

energy: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nE])

List of photon energies of the X-ray beam. If scanned through multiple values,
then an 'axis' attribute will be required to link the field to the appropriate data array dimension.

sample_y: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nY])

List of Y positions on the sample. If scanned through multiple values,
then an 'axis' attribute will be required to link the field to the appropriate data array dimension.

sample_x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nX])

List of X positions on the sample. If scanned through multiple values,
then an 'axis' attribute will be required to link the field to the appropriate data array dimension.

control: (optional) *NXmonitor* <=

data: (required) *NX_FLOAT*

Values to use to normalise for time-variations in photon flux. Typically, the synchrotron storage ring
electron beam current is used as a proxy for the X-ray beam intensity. Array must have same shape as the NXdata groups.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXstxm/ENTRY-group*
- */NXstxm/ENTRY/control-group*
- */NXstxm/ENTRY/control/data-field*
- */NXstxm/ENTRY/DATA-group*
- */NXstxm/ENTRY/DATA/data-field*
- */NXstxm/ENTRY/DATA/energy-field*

- */NXstxm/ENTRY/DATA/sample_x-field*
- */NXstxm/ENTRY/DATA/sample_y-field*
- */NXstxm/ENTRY/DATA/stxm_scan_type-field*
- */NXstxm/ENTRY/definition-field*
- */NXstxm/ENTRY/end_time-field*
- */NXstxm/ENTRY/INSTRUMENT-group*
- */NXstxm/ENTRY/INSTRUMENT/DETECTOR-group*
- */NXstxm/ENTRY/INSTRUMENT/DETECTOR/data-field*
- */NXstxm/ENTRY/INSTRUMENT/monochromator-group*
- */NXstxm/ENTRY/INSTRUMENT/monochromator/energy-field*
- */NXstxm/ENTRY/INSTRUMENT/sample_x-group*
- */NXstxm/ENTRY/INSTRUMENT/sample_x/data-field*
- */NXstxm/ENTRY/INSTRUMENT/sample_y-group*
- */NXstxm/ENTRY/INSTRUMENT/sample_y/data-field*
- */NXstxm/ENTRY/INSTRUMENT/sample_z-group*
- */NXstxm/ENTRY/INSTRUMENT/sample_z/data-field*
- */NXstxm/ENTRY/INSTRUMENT/SOURCE-group*
- */NXstxm/ENTRY/INSTRUMENT/SOURCE/name-field*
- */NXstxm/ENTRY/INSTRUMENT/SOURCE/probe-field*
- */NXstxm/ENTRY/INSTRUMENT/SOURCE/type-field*
- */NXstxm/ENTRY/SAMPLE-group*
- */NXstxm/ENTRY/SAMPLE/rotation_angle-field*
- */NXstxm/ENTRY/start_time-field*
- */NXstxm/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXstxm.nxdl.xml

NXtas

Status:

application definition, extends *NXobject*

Description:

This is an application definition for a triple axis spectrometer.

It is for the trademark scan of the TAS, the Q-E scan. For your alignment scans use the rules in *NXscan*.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXcrystal, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtas`

INSTRUMENT: (required) *NXinstrument* <=

SOURCE: (required) *NXsource* <=

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

Any of these values: `neutron` | `x-ray`

monochromator: (required) *NXcrystal* <=

ei: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ENERGY*}

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

analyser: (required) *NXcrystal* <=

ef: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ENERGY*}

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

DETECTOR: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 1, Dimensions: [nP])

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

SAMPLE: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

qh: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_DIMENSIONLESS*}

qk: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_DIMENSIONLESS*}

ql: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_DIMENSIONLESS*}

en: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ENERGY*}

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} *<=*

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

sgu: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

sgl: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

unit_cell: (required) *NX_FLOAT* (Rank: 1, Dimensions: [6]) {units=*NX_LENGTH*} *<=*

orientation_matrix: (required) *NX_FLOAT* (Rank: 1, Dimensions: [9]) {units=*NX_DIMENSIONLESS*} *<=*

MONITOR: (required) *NXmonitor* *<=*

mode: (required) *NX_CHAR* *<=*

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: `monitor | timer`

preset: (required) *NX_FLOAT*

preset value for time or monitor

data: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANY*}

Total integral monitor counts

DATA: (required) *NXdata* *<=*

One of the ei,ef,qh,qk,ql,en should get a primary=1 attribute to denote the main scan axis

ei: *link* (suggested target: `/NXentry/NXinstrument/monochromator:NXcrystal/ei`)

ef: *link* (suggested target: `/NXentry/NXinstrument/analyser:NXcrystal/ef`)

en: *link* (suggested target: `/NXentry/NXsample/en`)

qh: *link* (suggested target: `/NXentry/NXsample/qh`)

qk: *link* (suggested target: `/NXentry/NXsample/qk`)

ql: *link* (suggested target: `/NXentry/NXsample/ql`)

data: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtas/ENTRY-group*
- */NXtas/ENTRY/DATA-group*
- */NXtas/ENTRY/DATA/data-link*
- */NXtas/ENTRY/DATA/ef-link*
- */NXtas/ENTRY/DATA/ei-link*
- */NXtas/ENTRY/DATA/en-link*
- */NXtas/ENTRY/DATA/qh-link*
- */NXtas/ENTRY/DATA/qk-link*
- */NXtas/ENTRY/DATA/ql-link*
- */NXtas/ENTRY/definition-field*
- */NXtas/ENTRY/INSTRUMENT-group*
- */NXtas/ENTRY/INSTRUMENT/analyser-group*
- */NXtas/ENTRY/INSTRUMENT/analyser/ef-field*
- */NXtas/ENTRY/INSTRUMENT/analyser/polar_angle-field*
- */NXtas/ENTRY/INSTRUMENT/analyser/rotation_angle-field*
- */NXtas/ENTRY/INSTRUMENT/DETECTOR-group*
- */NXtas/ENTRY/INSTRUMENT/DETECTOR/data-field*
- */NXtas/ENTRY/INSTRUMENT/DETECTOR/polar_angle-field*
- */NXtas/ENTRY/INSTRUMENT/monochromator-group*
- */NXtas/ENTRY/INSTRUMENT/monochromator/ei-field*
- */NXtas/ENTRY/INSTRUMENT/monochromator/rotation_angle-field*
- */NXtas/ENTRY/INSTRUMENT/SOURCE-group*
- */NXtas/ENTRY/INSTRUMENT/SOURCE/name-field*
- */NXtas/ENTRY/INSTRUMENT/SOURCE/probe-field*
- */NXtas/ENTRY/MONITOR-group*
- */NXtas/ENTRY/MONITOR/data-field*
- */NXtas/ENTRY/MONITOR/mode-field*
- */NXtas/ENTRY/MONITOR/preset-field*
- */NXtas/ENTRY/SAMPLE-group*
- */NXtas/ENTRY/SAMPLE/en-field*
- */NXtas/ENTRY/SAMPLE/name-field*
- */NXtas/ENTRY/SAMPLE/orientation_matrix-field*
- */NXtas/ENTRY/SAMPLE/polar_angle-field*
- */NXtas/ENTRY/SAMPLE/qh-field*

- */NXtas/ENTRY/SAMPLE/qk-field*
- */NXtas/ENTRY/SAMPLE/ql-field*
- */NXtas/ENTRY/SAMPLE/rotation_angle-field*
- */NXtas/ENTRY/SAMPLE/sgl-field*
- */NXtas/ENTRY/SAMPLE/sgu-field*
- */NXtas/ENTRY/SAMPLE/unit_cell-field*
- */NXtas/ENTRY/start_time-field*
- */NXtas/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtas.nxdl.xml

NXtofnpd

Status:

application definition, extends *NXobject*

Description:

This is a application definition for raw data from a TOF neutron powder diffractometer

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nDet: Number of detectors

nTimeChan: nTimeChan description

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtofnpd`

pre_sample_flightpath: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the flight path before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

user: (required) *NXuser* <=

name: (required) *NX_CHAR* <=

INSTRUMENT: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 2, Dimensions: [nDet, nTimeChan])

detector_number: (required) *NX_INT* (Rank: 1, Dimensions: [nDet]) <=

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_LENGTH*} <=

distance to sample for each detector

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*} <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} <=

polar angle for each detector element

azimuthal_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} <=

azimuthal angle for each detector element

SAMPLE: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

MONITOR: (required) *NXmonitor* <=

mode: (required) *NX_CHAR* <=

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: `monitor | timer`

preset: (required) *NX_FLOAT*

preset value for time or monitor

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

data: (required) *NX_INT* (Rank: 1, Dimensions: [nTimeChan])

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*} <=

data: (required) *NXdata* <=

data: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

detector_number: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/detector_number`)

time_of_flight: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/time_of_flight`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtofnpd/ENTRY-group*
- */NXtofnpd/ENTRY/data-group*
- */NXtofnpd/ENTRY/data/data-link*
- */NXtofnpd/ENTRY/data/detector_number-link*
- */NXtofnpd/ENTRY/data/time_of_flight-link*
- */NXtofnpd/ENTRY/definition-field*
- */NXtofnpd/ENTRY/INSTRUMENT-group*
- */NXtofnpd/ENTRY/INSTRUMENT/detector-group*
- */NXtofnpd/ENTRY/INSTRUMENT/detector/azimuthal_angle-field*
- */NXtofnpd/ENTRY/INSTRUMENT/detector/data-field*
- */NXtofnpd/ENTRY/INSTRUMENT/detector/detector_number-field*
- */NXtofnpd/ENTRY/INSTRUMENT/detector/distance-field*
- */NXtofnpd/ENTRY/INSTRUMENT/detector/polar_angle-field*
- */NXtofnpd/ENTRY/INSTRUMENT/detector/time_of_flight-field*
- */NXtofnpd/ENTRY/MONITOR-group*
- */NXtofnpd/ENTRY/MONITOR/data-field*
- */NXtofnpd/ENTRY/MONITOR/distance-field*
- */NXtofnpd/ENTRY/MONITOR/mode-field*
- */NXtofnpd/ENTRY/MONITOR/preset-field*
- */NXtofnpd/ENTRY/MONITOR/time_of_flight-field*
- */NXtofnpd/ENTRY/pre_sample_flightpath-field*
- */NXtofnpd/ENTRY/SAMPLE-group*
- */NXtofnpd/ENTRY/SAMPLE/name-field*
- */NXtofnpd/ENTRY/start_time-field*
- */NXtofnpd/ENTRY/title-field*
- */NXtofnpd/ENTRY/user-group*
- */NXtofnpd/ENTRY/user/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtofnpd.nxdl.xml

NXtofraw

Status:

application definition, extends *NXobject*

Description:

This is an application definition for raw data from a generic TOF instrument

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nDet: Number of detectors

nTimeChan: nTimeChan description

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtofraw`

duration: (required) *NX_FLOAT*

run_number: (required) *NX_INT*

pre_sample_flightpath: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the flight path before the sample position. This can be determined by a chopper, by the moderator, or the source itself. In other words: it is the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

user: (required) *NXuser* <=

name: (required) *NX_CHAR* <=

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 2, Dimensions: [nDet, nTimeChan])

detector_number: (required) *NX_INT* (Rank: 1, Dimensions: [nDet]) <=

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_LENGTH*} <=

distance to sample for each detector

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*} <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} <=

polar angle for each detector element

azimuthal_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} *<=*

azimuthal angle for each detector element

SAMPLE: (required) *NXsample <=*

name: (required) *NX_CHAR <=*

Descriptive name of sample

nature: (required) *NX_CHAR*

Any of these values: `powder|liquid|single crystal`

MONITOR: (required) *NXmonitor <=*

mode: (required) *NX_CHAR <=*

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: `monitor|timer`

preset: (required) *NX_FLOAT*

preset value for time or monitor

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} *<=*

data: (required) *NX_INT* (Rank: 1, Dimensions: [nTimeChan])

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*} *<=*

integral_counts: (required) *NX_INT* {units=*NX_UNITLESS*}

data: (required) *NXdata <=*

data: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

detector_number: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/detector_number`)

time_of_flight: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/time_of_flight`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtofraw/ENTRY-group*
- */NXtofraw/ENTRY/data-group*
- */NXtofraw/ENTRY/data/data-link*
- */NXtofraw/ENTRY/data/detector_number-link*
- */NXtofraw/ENTRY/data/time_of_flight-link*
- */NXtofraw/ENTRY/definition-field*
- */NXtofraw/ENTRY/duration-field*

- */NXtofraw/ENTRY/instrument-group*
- */NXtofraw/ENTRY/instrument/detector-group*
- */NXtofraw/ENTRY/instrument/detector/azimuthal_angle-field*
- */NXtofraw/ENTRY/instrument/detector/data-field*
- */NXtofraw/ENTRY/instrument/detector/detector_number-field*
- */NXtofraw/ENTRY/instrument/detector/distance-field*
- */NXtofraw/ENTRY/instrument/detector/polar_angle-field*
- */NXtofraw/ENTRY/instrument/detector/time_of_flight-field*
- */NXtofraw/ENTRY/MONITOR-group*
- */NXtofraw/ENTRY/MONITOR/data-field*
- */NXtofraw/ENTRY/MONITOR/distance-field*
- */NXtofraw/ENTRY/MONITOR/integral_counts-field*
- */NXtofraw/ENTRY/MONITOR/mode-field*
- */NXtofraw/ENTRY/MONITOR/preset-field*
- */NXtofraw/ENTRY/MONITOR/time_of_flight-field*
- */NXtofraw/ENTRY/pre_sample_flightpath-field*
- */NXtofraw/ENTRY/run_number-field*
- */NXtofraw/ENTRY/SAMPLE-group*
- */NXtofraw/ENTRY/SAMPLE/name-field*
- */NXtofraw/ENTRY/SAMPLE/nature-field*
- */NXtofraw/ENTRY/start_time-field*
- */NXtofraw/ENTRY/title-field*
- */NXtofraw/ENTRY/user-group*
- */NXtofraw/ENTRY/user/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtofraw.nxdl.xml

NXtofsingle

Status:

application definition, extends *NXobject*

Description:

This is a application definition for raw data from a generic TOF instrument

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

xSize: xSize description

ySize: ySize description

nDet: Number of detectors

nTimeChan: nTimeChan description

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtofsingle`

duration: (required) *NX_FLOAT*

pre_sample_flightpath: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the flight path before the sample position. This can be determined by a chopper, by the moderator or the source itself. In other words: it the distance to the component which gives the T0 signal to the detector electronics. If another component in the NXinstrument hierarchy provides this information, this should be a link.

user: (required) *NXuser* <=

name: (required) *NX_CHAR* <=

INSTRUMENT: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 3, Dimensions: [xSize, ySize, nTimeChan])

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [1]) {units=*NX_LENGTH*} <=

Distance to sample for the center of the detector

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*} <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} <=

polar angle for each detector element

azimuthal_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} <=

azimuthal angle for each detector element

SAMPLE: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

nature: (required) *NX_CHAR*

Any of these values: `powder|liquid|single crystal`

MONITOR: (required) *NXmonitor* <=

mode: (required) *NX_CHAR* <=

Count to a preset value based on either clock time (timer) or received monitor counts (monitor).

Any of these values: `monitor | timer`

preset: (required) *NX_FLOAT*

preset value for time or monitor

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

data: (required) *NX_INT* (Rank: 1, Dimensions: [nTimeChan])

time_of_flight: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nTimeChan]) {units=*NX_TIME_OF_FLIGHT*} <=

data: (required) *NXdata* <=

data: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

time_of_flight: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/time_of_flight`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtofsingle/ENTRY-group*
- */NXtofsingle/ENTRY/data-group*
- */NXtofsingle/ENTRY/data/data-link*
- */NXtofsingle/ENTRY/data/time_of_flight-link*
- */NXtofsingle/ENTRY/definition-field*
- */NXtofsingle/ENTRY/duration-field*
- */NXtofsingle/ENTRY/INSTRUMENT-group*
- */NXtofsingle/ENTRY/INSTRUMENT/detector-group*
- */NXtofsingle/ENTRY/INSTRUMENT/detector/azimuthal_angle-field*
- */NXtofsingle/ENTRY/INSTRUMENT/detector/data-field*
- */NXtofsingle/ENTRY/INSTRUMENT/detector/distance-field*
- */NXtofsingle/ENTRY/INSTRUMENT/detector/polar_angle-field*
- */NXtofsingle/ENTRY/INSTRUMENT/detector/time_of_flight-field*
- */NXtofsingle/ENTRY/MONITOR-group*
- */NXtofsingle/ENTRY/MONITOR/data-field*
- */NXtofsingle/ENTRY/MONITOR/distance-field*
- */NXtofsingle/ENTRY/MONITOR/mode-field*
- */NXtofsingle/ENTRY/MONITOR/preset-field*
- */NXtofsingle/ENTRY/MONITOR/time_of_flight-field*

- */NXtofsingle/ENTRY/pre_sample_flightpath-field*
- */NXtofsingle/ENTRY/SAMPLE-group*
- */NXtofsingle/ENTRY/SAMPLE/name-field*
- */NXtofsingle/ENTRY/SAMPLE/nature-field*
- */NXtofsingle/ENTRY/start_time-field*
- */NXtofsingle/ENTRY/title-field*
- */NXtofsingle/ENTRY/user-group*
- */NXtofsingle/ENTRY/user/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtofsingle.nxdl.xml

NXtomo

Status:

application definition, extends *NXobject*

Description:

This is the application definition for x-ray or neutron tomography raw data.

In tomography a number of dark field images are measured, some bright field images and, of course the sample. In order to distinguish between them images carry a image_key.

Symbols:

These symbols will be used below to coordinate datasets with the same shape.

nFrames: Number of frames

xSize: Number of pixels in X direction

ySize: Number of pixels in Y direction

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

title: (optional) *NX_CHAR* <=

start_time: (optional) *NX_DATE_TIME* <=

end_time: (optional) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtomo`

instrument: (required) *NXinstrument* <=

SOURCE: (optional) *NXsource* <=

type: (optional) *NX_CHAR* <=

name: (optional) *NX_CHAR* <=

probe: (optional) *NX_CHAR* <=

Any of these values: `neutron` | `x-ray` | `electron`

detector: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 3, Dimensions: [nFrames, xSize, ySize])

image_key: (required) *NX_INT* (Rank: 1, Dimensions: [nFrames]) <=

In order to distinguish between sample projections, dark and flat images, a magic number is recorded per frame. The key is as follows:

- projection = 0
- flat field = 1
- dark field = 2
- invalid = 3

x_pixel_size: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

y_pixel_size: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

distance: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

Distance between detector and sample

x_rotation_axis_pixel_position: (optional) *NX_FLOAT*

y_rotation_axis_pixel_position: (optional) *NX_FLOAT*

sample: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nFrames]) {units=*NX_ANGLE*} <=

In practice this axis is always aligned along one pixel direction on the detector and usually vertical. There are experiments with horizontal rotation axes, so this would need to be indicated somehow. For now the best way for that is an open question.

x_translation: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nFrames]) {units=*NX_LENGTH*} <=

y_translation: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nFrames]) {units=*NX_LENGTH*}

z_translation: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nFrames]) {units=*NX_LENGTH*}

control: (optional) *NXmonitor* <=

data: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nFrames]) {units=*NX_ANY*}

Total integral monitor counts for each measured frame. Allows a to correction for fluctuations in the beam between frames.

data: (required) *NXdata* <=

data: *link* (suggested target: `/NXentry/NXinstrument/detector:NXdetector/data`)

rotation_angle: *link* (suggested target: `/NXentry/NXsample/rotation_angle`)

image_key: *link* (suggested target: `/NXentry/NXinstrument/detector:NXdetector/image_key`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtomo/ENTRY-group*
- */NXtomo/ENTRY/control-group*
- */NXtomo/ENTRY/control/data-field*
- */NXtomo/ENTRY/data-group*
- */NXtomo/ENTRY/data/data-link*
- */NXtomo/ENTRY/data/image_key-link*
- */NXtomo/ENTRY/data/rotation_angle-link*
- */NXtomo/ENTRY/definition-field*
- */NXtomo/ENTRY/end_time-field*
- */NXtomo/ENTRY/instrument-group*
- */NXtomo/ENTRY/instrument/detector-group*
- */NXtomo/ENTRY/instrument/detector/data-field*
- */NXtomo/ENTRY/instrument/detector/distance-field*
- */NXtomo/ENTRY/instrument/detector/image_key-field*
- */NXtomo/ENTRY/instrument/detector/x_pixel_size-field*
- */NXtomo/ENTRY/instrument/detector/x_rotation_axis_pixel_position-field*
- */NXtomo/ENTRY/instrument/detector/y_pixel_size-field*
- */NXtomo/ENTRY/instrument/detector/y_rotation_axis_pixel_position-field*
- */NXtomo/ENTRY/instrument/SOURCE-group*
- */NXtomo/ENTRY/instrument/SOURCE/name-field*
- */NXtomo/ENTRY/instrument/SOURCE/probe-field*
- */NXtomo/ENTRY/instrument/SOURCE/type-field*
- */NXtomo/ENTRY/sample-group*
- */NXtomo/ENTRY/sample/name-field*
- */NXtomo/ENTRY/sample/rotation_angle-field*
- */NXtomo/ENTRY/sample/x_translation-field*
- */NXtomo/ENTRY/sample/y_translation-field*
- */NXtomo/ENTRY/sample/z_translation-field*

- */NXtomo/ENTRY/start_time-field*
- */NXtomo/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtomo.nxdl.xml

NXtomophase

Status:

application definition, extends *NXobject*

Description:

This is the application definition for x-ray or neutron tomography raw data with phase contrast variation at each point.

In tomography first some dark field images are measured, some bright field images and, of course the sample. In order to properly sort the order of the images taken, a sequence number is stored with each image.

Symbols:

These symbols will be used below to coordinate datasets with the same shape.

nBrightFrames: Number of bright frames

nDarkFrames: Number of dark frames

nSampleFrames: Number of image (sample) frames

nPhase: Number of phase settings

xSize: Number of pixels in X direction

ySize: Number of pixels in Y direction

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

start_time: (required) *NX_DATE_TIME* <=

end_time: (required) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtomophase`

instrument: (required) *NXinstrument* <=

SOURCE: (required) *NXsource* <=

type: (required) *NX_CHAR* <=

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

Any of these values: `neutron` | `x-ray` | `electron`

bright_field: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 3, Dimensions: [nBrightFrames, xSize, ySize])

sequence_number: (required) *NX_INT* (Rank: 1, Dimensions: [nBrightFrames]) <=

dark_field: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 3, Dimensions: [nDarkFrames, xSize, ySize])

sequence_number: (required) *NX_INT* (Rank: 1, Dimensions: [nDarkFrames]) <=

sample: (required) *NXdetector* <=

data: (required) *NX_INT* (Rank: 4, Dimensions: [nSampleFrames, nPhase, xSize, ySize])

sequence_number: (required) *NX_INT* (Rank: 2, Dimensions: [nSampleFrames, nPhase]) <=

x_pixel_size: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

y_pixel_size: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

Distance between detector and sample

sample: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nSampleFrames]) {units=*NX_ANGLE*} <=

x_translation: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nSampleFrames]) {units=*NX_LENGTH*} <=

y_translation: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nSampleFrames]) {units=*NX_LENGTH*}

z_translation: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nSampleFrames]) {units=*NX_LENGTH*}

control: (required) *NXmonitor* <=

integral: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDarkFrames + nBrightFrames + nSampleFrame]) {units=*NX_ANY*}

Total integral monitor counts for each measured frame. Allows a correction for fluctuations in the beam between frames.

data: (required) *NXdata* <=

data: *link* (suggested target: `/NXentry/NXinstrument/sample:NXdetector/data`)

rotation_angle: *link* (suggested target: `/NXentry/NXsample/rotation_angle`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXtomophase/ENTRY-group*
- */NXtomophase/ENTRY/control-group*
- */NXtomophase/ENTRY/control/integral-field*
- */NXtomophase/ENTRY/data-group*
- */NXtomophase/ENTRY/data/data-link*
- */NXtomophase/ENTRY/data/rotation_angle-link*
- */NXtomophase/ENTRY/definition-field*
- */NXtomophase/ENTRY/end_time-field*
- */NXtomophase/ENTRY/instrument-group*
- */NXtomophase/ENTRY/instrument/bright_field-group*
- */NXtomophase/ENTRY/instrument/bright_field/data-field*
- */NXtomophase/ENTRY/instrument/bright_field/sequence_number-field*
- */NXtomophase/ENTRY/instrument/dark_field-group*
- */NXtomophase/ENTRY/instrument/dark_field/data-field*
- */NXtomophase/ENTRY/instrument/dark_field/sequence_number-field*
- */NXtomophase/ENTRY/instrument/sample-group*
- */NXtomophase/ENTRY/instrument/sample/data-field*
- */NXtomophase/ENTRY/instrument/sample/distance-field*
- */NXtomophase/ENTRY/instrument/sample/sequence_number-field*
- */NXtomophase/ENTRY/instrument/sample/x_pixel_size-field*
- */NXtomophase/ENTRY/instrument/sample/y_pixel_size-field*
- */NXtomophase/ENTRY/instrument/SOURCE-group*
- */NXtomophase/ENTRY/instrument/SOURCE/name-field*
- */NXtomophase/ENTRY/instrument/SOURCE/probe-field*
- */NXtomophase/ENTRY/instrument/SOURCE/type-field*
- */NXtomophase/ENTRY/sample-group*
- */NXtomophase/ENTRY/sample/name-field*
- */NXtomophase/ENTRY/sample/rotation_angle-field*
- */NXtomophase/ENTRY/sample/x_translation-field*
- */NXtomophase/ENTRY/sample/y_translation-field*
- */NXtomophase/ENTRY/sample/z_translation-field*
- */NXtomophase/ENTRY/start_time-field*
- */NXtomophase/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXtomophase.nxdl.xml

NXtomoproc

Status:

application definition, extends *NXobject*

Description:

This is an application definition for the final result of a tomography experiment: a 3D construction of some volume of physical properties.

Symbols:

These symbols will be used below to coordinate datasets with the same shape.

nX: Number of voxels in X direction

nY: Number of voxels in Y direction

nZ: Number of voxels in Z direction

Groups cited:
NXdata, *NXentry*, *NXinstrument*, *NXparameters*, *NXprocess*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry*

title: (required) *NX_CHAR* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtomoproc`

INSTRUMENT: (required) *NXinstrument* <=

SOURCE: (required) *NXsource* <=

type: (required) *NX_CHAR* <=

name: (required) *NX_CHAR* <=

probe: (required) *NX_CHAR* <=

Any of these values: `neutron` | `x-ray` | `electron`

SAMPLE: (required) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name of sample

reconstruction: (required) *NXprocess* <=

program: (required) *NX_CHAR* <=

Name of the program used for reconstruction

version: (required) *NX_CHAR* <=

Version of the program used

date: (required) *NX_DATE_TIME* <=

Date and time of reconstruction processing.

parameters: (required) *NXparameters* <=

raw_file: (required) *NX_CHAR*

Original raw data file this data was derived from

data: (required) *NXdata* <=

data: (required) *NX_NUMBER* (Rank: 3, Dimensions: [nX, nY, 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.

- */NXxas/ENTRY-group*
- */NXxas/ENTRY/DATA-group*
- */NXxas/ENTRY/DATA/absorbed_beam-link*
- */NXxas/ENTRY/DATA/energy-link*
- */NXxas/ENTRY/DATA/mode-field*
- */NXxas/ENTRY/definition-field*
- */NXxas/ENTRY/INSTRUMENT-group*
- */NXxas/ENTRY/INSTRUMENT/absorbed_beam-group*
- */NXxas/ENTRY/INSTRUMENT/absorbed_beam/data-field*
- */NXxas/ENTRY/INSTRUMENT/incoming_beam-group*
- */NXxas/ENTRY/INSTRUMENT/incoming_beam/data-field*
- */NXxas/ENTRY/INSTRUMENT/monochromator-group*
- */NXxas/ENTRY/INSTRUMENT/monochromator/energy-field*
- */NXxas/ENTRY/INSTRUMENT/SOURCE-group*
- */NXxas/ENTRY/INSTRUMENT/SOURCE/name-field*
- */NXxas/ENTRY/INSTRUMENT/SOURCE/probe-field*
- */NXxas/ENTRY/INSTRUMENT/SOURCE/type-field*
- */NXxas/ENTRY/MONITOR-group*
- */NXxas/ENTRY/MONITOR/data-field*
- */NXxas/ENTRY/MONITOR/mode-field*
- */NXxas/ENTRY/MONITOR/preset-field*
- */NXxas/ENTRY/SAMPLE-group*
- */NXxas/ENTRY/SAMPLE/name-field*
- */NXxas/ENTRY/start_time-field*
- */NXxas/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxas.nxdl.xml

NXxasproc

Status:

application definition, extends *NXobject*

Description:

Processed data from XAS. This is energy versus I(incoming)/I(absorbed).

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, *NXentry*, *NXparameters*, *NXprocess*, *NXsample*

Structure:

ENTRY: (required) *NXentry*

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.

- */NXxbase/ENTRY-group*
- */NXxbase/ENTRY/control-group*
- */NXxbase/ENTRY/control/integral-field*
- */NXxbase/ENTRY/control/mode-field*
- */NXxbase/ENTRY/control/preset-field*
- */NXxbase/ENTRY/DATA-group*
- */NXxbase/ENTRY/DATA/data-link*
- */NXxbase/ENTRY/definition-field*
- */NXxbase/ENTRY/instrument-group*
- */NXxbase/ENTRY/instrument/detector-group*

- */NXxbase/ENTRY/instrument/detector/data-field*
- */NXxbase/ENTRY/instrument/detector/data@signal-attribute*
- */NXxbase/ENTRY/instrument/detector/distance-field*
- */NXxbase/ENTRY/instrument/detector/frame_start_number-field*
- */NXxbase/ENTRY/instrument/detector/x_pixel_size-field*
- */NXxbase/ENTRY/instrument/detector/y_pixel_size-field*
- */NXxbase/ENTRY/instrument/monochromator-group*
- */NXxbase/ENTRY/instrument/monochromator/wavelength-field*
- */NXxbase/ENTRY/instrument/source-group*
- */NXxbase/ENTRY/instrument/source/name-field*
- */NXxbase/ENTRY/instrument/source/probe-field*
- */NXxbase/ENTRY/instrument/source/type-field*
- */NXxbase/ENTRY/sample-group*
- */NXxbase/ENTRY/sample/distance-field*
- */NXxbase/ENTRY/sample/name-field*
- */NXxbase/ENTRY/sample/orientation_matrix-field*
- */NXxbase/ENTRY/sample/temperature-field*
- */NXxbase/ENTRY/sample/unit_cell-field*
- */NXxbase/ENTRY/sample/x_translation-field*
- */NXxbase/ENTRY/sample/y_translation-field*
- */NXxbase/ENTRY/start_time-field*
- */NXxbase/ENTRY/title-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxbase.nxdl.xml

NXxeuler

Status:

application definition, extends *NXxbase*

Description:

raw data from a four-circle diffractometer with an eulerian cradle, extends *NXxbase*

It extends *NXxbase*, so the full definition is the content of *NXxbase* plus the data defined here. All four angles are logged in order to support arbitrary scans in reciprocal space.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXsample*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxeuler`

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

The polar_angle (two theta) where the detector is placed at each scan point.

sample: (required) *NXsample* <=

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

This is an array holding the sample rotation angle at each scan point

chi: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the chi angle of the eulerian cradle at each scan point

phi: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the phi rotation of the eulerian cradle at each scan point

name: (required) *NXdata* <=

polar_angle: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/polar_angle`)

rotation_angle: *link* (suggested target: `/NXentry/NXsample/rotation_angle`)

chi: *link* (suggested target: `/NXentry/NXsample/chi`)

phi: *link* (suggested target: `/NXentry/NXsample/phi`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxeuler/ENTRY-group*
- */NXxeuler/ENTRY/definition-field*
- */NXxeuler/ENTRY/instrument-group*
- */NXxeuler/ENTRY/instrument/detector-group*
- */NXxeuler/ENTRY/instrument/detector/polar_angle-field*
- */NXxeuler/ENTRY/name-group*
- */NXxeuler/ENTRY/name/chi-link*
- */NXxeuler/ENTRY/name/phi-link*

- */NXxeuler/ENTRY/name/polar_angle-link*
- */NXxeuler/ENTRY/name/rotation_angle-link*
- */NXxeuler/ENTRY/sample-group*
- */NXxeuler/ENTRY/sample/chi-field*
- */NXxeuler/ENTRY/sample/phi-field*
- */NXxeuler/ENTRY/sample/rotation_angle-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxeuler.nxdl.xml

NXxkappa

Status:

application definition, extends *NXxbase*

Description:

raw data from a kappa geometry (CAD4) single crystal diffractometer, extends *NXxbase*

This is the application definition for raw data from a kappa geometry (CAD4) single crystal diffractometer. It extends *NXxbase*, so the full definition is the content of *NXxbase* plus the data defined here.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXsample*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxkappa`

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

The polar_angle (two theta) at each scan point

sample: (required) *NXsample* <=

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

This is an array holding the sample rotation angle at each scan point

kappa: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the kappa angle at each scan point

phi: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the phi angle at each scan point

alpha: (required) *NX_FLOAT* {units=*NX_ANGLE*}

This holds the inclination angle of the kappa arm.

name: (required) *NXdata* <=

polar_angle: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/polar_angle`)

rotation_angle: *link* (suggested target: `/NXentry/NXsample/rotation_angle`)

kappa: *link* (suggested target: `/NXentry/NXsample/kappa`)

phi: *link* (suggested target: `/NXentry/NXsample/phi`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxkappa/ENTRY-group*
- */NXxkappa/ENTRY/definition-field*
- */NXxkappa/ENTRY/instrument-group*
- */NXxkappa/ENTRY/instrument/detector-group*
- */NXxkappa/ENTRY/instrument/detector/polar_angle-field*
- */NXxkappa/ENTRY/name-group*
- */NXxkappa/ENTRY/name/kappa-link*
- */NXxkappa/ENTRY/name/phi-link*
- */NXxkappa/ENTRY/name/polar_angle-link*
- */NXxkappa/ENTRY/name/rotation_angle-link*
- */NXxkappa/ENTRY/sample-group*
- */NXxkappa/ENTRY/sample/alpha-field*
- */NXxkappa/ENTRY/sample/kappa-field*
- */NXxkappa/ENTRY/sample/phi-field*
- */NXxkappa/ENTRY/sample/rotation_angle-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxkappa.nxdl.xml

NXxlaue

Status:

application definition, extends *NXxrot*

Description:

raw data from a single crystal laue camera, extends *NXxrot*

This is the application definition for raw data from a single crystal laue camera. It extends *NXxrot*.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nE: Number of energies

Groups cited:
NXdata, *NXentry*, *NXinstrument*, *NXsource*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxlaue`

instrument: (required) *NXinstrument* <=

source: (required) *NXsource* <=

distribution: (required) *NXdata* <=

This is the wavelength distribution of the beam

data: (required) *NX_CHAR* (Rank: 1, Dimensions: [nE])

expect `signal=1 axes="energy"`

wavelength: (required) *NX_CHAR* (Rank: 1, Dimensions: [nE]) {units=*NX_WAVELENGTH*}

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxlaue/ENTRY-group*
- */NXxlaue/ENTRY/definition-field*
- */NXxlaue/ENTRY/instrument-group*
- */NXxlaue/ENTRY/instrument/source-group*
- */NXxlaue/ENTRY/instrument/source/distribution-group*
- */NXxlaue/ENTRY/instrument/source/distribution/data-field*
- */NXxlaue/ENTRY/instrument/source/distribution/wavelength-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxlaue.nxdl.xml

NXxlaueplate

Status:

application definition, extends *NXxlaue*

Description:

raw data from a single crystal Laue camera, extends *NXxlaue*

This is the application definition for raw data from a single crystal Laue camera with an image plate as a detector. It extends *NXxlaue*.

Symbols:

No symbol table

Groups cited:
NXdetector, *NXentry*, *NXinstrument*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxlaueplate`

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

diameter: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

The diameter of a cylindrical detector

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxlaueplate/ENTRY-group*
- */NXxlaueplate/ENTRY/definition-field*
- */NXxlaueplate/ENTRY/instrument-group*
- */NXxlaueplate/ENTRY/instrument/detector-group*
- */NXxlaueplate/ENTRY/instrument/detector/diameter-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxlaueplate.nxdl.xml

NXxnb

Status:

application definition, extends *NXxbase*

Description:

raw data from a single crystal diffractometer, extends *NXxbase*

This is the application definition for raw data from a single crystal diffractometer measuring in normal beam mode. It extends *NXxbase*, so the full definition is the content of *NXxbase* plus the data defined here. All angles are logged in order to support arbitrary scans in reciprocal space.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXsample*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxnb`

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

The polar_angle (gamma) of the detector for each scan point.

tilt_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

The angle by which the detector has been tilted out of the scattering plane.

sample: (required) *NXsample* <=

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

This is an array holding the sample rotation angle at each scan point

name: (required) *NXdata* <=

polar_angle: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/polar_angle`)

tilt: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/tilt`)

rotation_angle: *link* (suggested target: `/NXentry/NXsample/rotation_angle`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxnb/ENTRY-group*
- */NXxnb/ENTRY/definition-field*
- */NXxnb/ENTRY/instrument-group*
- */NXxnb/ENTRY/instrument/detector-group*
- */NXxnb/ENTRY/instrument/detector/polar_angle-field*
- */NXxnb/ENTRY/instrument/detector/tilt_angle-field*
- */NXxnb/ENTRY/name-group*
- */NXxnb/ENTRY/name/polar_angle-link*
- */NXxnb/ENTRY/name/rotation_angle-link*
- */NXxnb/ENTRY/name/tilt-link*
- */NXxnb/ENTRY/sample-group*
- */NXxnb/ENTRY/sample/rotation_angle-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxnb.nxdl.xml

NXxps

Status:

application definition, extends *NXmpes*

Description:

This is the application definition for X-ray photoelectron spectroscopy.

Symbols:

No symbol table

Groups cited:
NXbeam, *NXcalibration*, *NXcollectioncolumn*, *NXcoordinate_system*, *NXdata*, *NXelectronanalyzer*, *NXenergy-dispersion*, *NXentry*, *NXfit_function*, *NXfit*, *NXinstrument*, *NXparameters*, *NXpeak*, *NXsample*, *NXsource*, *NX-transformations*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXxps`

method: (required) *NX_CHAR* <=

A name of the experimental method according to Clause 11 of the ISO 18115-1:2023 specification.

Examples for XPS-related experiments include:

- X-ray photoelectron spectroscopy (XPS)

- angle-resolved X-ray photoelectron spectroscopy (ARXPS)
- ultraviolet photoelectron spectroscopy (UPS)
- hard X-ray photoemission spectroscopy (HAXPES)
- near ambient pressure X-ray photoelectron spectroscopy (NAPXPS)
- electron spectroscopy for chemical analysis (ESCA)

transitions: (recommended) *NX_CHAR* <=

xps_coordinate_system: (recommended) *NXcoordinate_system* <=

In traditional surface science, a left-handed coordinate system is used such that the positive z-axis points along the normal of the sample stage, and the x- and y-axes lie in the plane of the sample stage. However, in NeXus, a coordinate system that is the same as McStas is used. *xps_coordinate_system* gives the user the opportunity to work in the traditional base coordinate system.

origin: (required) *NX_CHAR* <=

Obligatory value: `sample stage`

z_direction: (required) *NX_CHAR* <=

Obligatory value: `sample stage normal`

x: (required) *NX_NUMBER* <=

Obligatory value: `[-1, 0, 0]`

y: (required) *NX_NUMBER* <=

Obligatory value: `[0, 1, 0]`

z: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

depends_on: (required) *NX_CHAR* <=

Link to transformations defining the XPS base coordinate system, which is defined such that the positive z-axis points along the sample stage normal, and the x- and y-axes lie in the plane of the sample stage.

TRANSFORMATIONS: (optional) *NXtransformations* <=

Set of transformations, describing the orientation of the XPS coordinate system with respect to the beam coordinate system (.) or another coordinate system. The transformations in the `NXtransformations` group depend on the actual instrument geometry. If the z-axis is pointing in the direction of gravity (i.e., if the sample is mounted horizontally), the following transformations can be used for describing the XPS coordinate system with respect to the beam coordinate system (.):

```
xps_coordinate_system:NXcoordinate_system
  depends_on=coordinate_transformations/sample_stage_to_
↪source_azimuth
  coordinate_transformations:NXtransformations
    sample_stage_to_source_azimuth=beam_azimuth_angle
      @depends_on=sample_stage_to_source_polar
      @transformation_type=rotation
      @vector=[0, 0, -1]
      @units=degree
    sample_stage_to_source_polar=beam_polar_angle_of_
↪incidence
      @depends_on=.
      @transformation_type=rotation
      @vector=[1, 0, 0]
      @units=degree
```

Note that this `NXtransformations` group is not needed when the defined *transformations in beam_probe* are used. In this case, this group shall not be written here to avoid circular references in the transformations chain.

INSTRUMENT: (required) *NXinstrument* <=

Description of the XPS spectrometer and its individual parts.

This concept is related to term 12.58 of the ISO 18115-1:2023 standard.

source_probe: (recommended) *NXsource* <=

power: (recommended) *NX_FLOAT* {units=*NX_POWER*} <=

beam_probe: (required) *NXbeam* <=

depends_on: (recommended) *NX_CHAR* <=

Reference to the transformation describing the direction of the beam relative to a defined coordinate system.

Should point to /entry/instrument/beam_probe/transformations/beam_direction.

transformations: (recommended) *NXtransformations* <=

beam_direction: (required) *NX_NUMBER* {units=*NX_UNITLESS*} <=

Beam direction in the XPS coordinate system after rotation.

@**vector**: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, -1]`

@**depends_on**: (required) *NX_CHAR* <=

Obligatory value: `beam_polar_angle_of_incidence`

beam_polar_angle_of_incidence: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Incidence angle of the beam with respect to the upward z-direction, defined by the sample stage.

@**transformation_type**: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[-1, 0, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `beam_azimuth_angle`

beam_azimuth_angle: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Azimuthal rotation of the beam from the y-direction defined by the sample stage.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@depends_on: (required) *NX_CHAR* <=

This should point to the coordinate system defined in /entry/xps_coordinate_system.

ELECTRONANALYZER: (required) *NXelectronanalyzer* <=

work_function: (required) *NX_FLOAT* <=

depends_on: (recommended) *NX_CHAR* <=

Reference to the transformation describing the orientation of the analyzer relative to a defined coordinate system.

transmission_function: (recommended) *NXdata* <=

COLLECTIONCOLUMN: (required) *NXcollectioncolumn* <=

magnification: (recommended) *NX_FLOAT* {units=*NX_DIMENSIONLESS*} <=

ENERGYDISPERSION: (required) *NXenergydispersion* <=

radius: (recommended) *NX_NUMBER* {units=*NX_LENGTH*}

energy_scan_mode: (required) *NX_CHAR* <=

transformations: (recommended) *NXtransformations* <=

analyzer_take_off_polar_angle: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Polar tilt of the analyzer with respect to the upward z-direction, defined by the sample stage.

The angle between the incoming beam and the analyzer is given by beam_analyzer_angle = beam_polar_angle_of_incidence + analyzer_take_off_polar_angle. In practice, the analyzer axis is often set as the z axis (analyzer_take_off_polar_angle = 0), so that beam_analyzer_angle = beam_polar_angle_of_incidence. For magic angle configurations, this angle is 54.5°.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[-1, 0, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `analyzer_take_off_azimuth_angle`

analyzer_take_off_azimuth_angle: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Azimuthal rotation of the analyzer from the y-direction defined by the sample stage.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@depends_on: (required) *NX_CHAR* <=

This should point to the coordinate system defined in /entry/xps_coordinate_system.

energy_referencing: (recommended) *NXcalibration* <=

transmission_correction: (recommended) *NXcalibration* <=

FIT: (recommended) *NXfit* <=

Peak model for XPS fitting. Each *NXfit* instance shall be used for the description of _one_ peak fit in _one_ XPS region. As an example, this could be used to describe the fitting of one measured C 1s spectrum.

This concept is related to term 3.29 of the ISO 18115-1:2023 standard.

label: (required) *NX_CHAR* <=

figure_of_meritMETRIC: (recommended) *NX_NUMBER* <=

@metric: (required) *NX_CHAR* <=

data: (required) *NXdata* <=

Input data and results of the fit.

input_dependent: (required) *NX_NUMBER* {units=*NX_ANY*} <=

Dependent variable for this fit procedure.

This could be a link to entry/data/data.

input_independent: (required) *NX_NUMBER* {units=*NX_ENERGY*} <=

Independent variable for this fit procedure.

This could be a link to entry/data/energy.

fit_sum: (required) *NX_NUMBER* {units=*NX_ANY*} <=

residual: (recommended) *NX_NUMBER* {units=*NX_ANY*} <=

peakPEAK: (required) *NXpeak* <=

label: (required) *NX_CHAR* <=

total_area: (recommended) *NX_NUMBER* {units=*NX_ANY*} <=

Total area under the peak after background removal.

This concept is related to term 3.16 of the ISO 18115-1:2023 standard.

relative_atomic_concentration: (optional) *NX_FLOAT* {units=*NX_ANY*}

Atomic concentration of the species defined by this peak. This should be a value between 0 and 1.

data: (required) *NXdata* <=

position: (required) *NX_NUMBER* {units=*NX_ENERGY*} <=

This could be a link to entry/data/energy.

intensity: (required) *NX_NUMBER* <=

Intensity values of the fitted function at each energy in the position field.

This concept is related to term 3.15 of the ISO 18115-1:2023 standard.

function: (recommended) *NXfit_function* <=

function_type: (required) *NX_CHAR* <=

Type of fit function used.

The user is encouraged to use one of the options defined in the enumeration, but in case none of these fit (e.g., in the case of very complex line shapes), a different value for the `function_type` field can be used. In that case in particular, but also if one of the suggested values is used, the functional form of the peak should be given by the `formula_description` field. The user is also encouraged to use the `description` field for describing the fit function in a human-readable way.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `Gaussian`: The Gaussian function models peaks with a symmetric shape, commonly arising from instrumental broadening and thermal motion in XPS. It is defined as $G(x) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, where - A is the peak amplitude, μ is the peak position - and σ is the standard deviation. The full width at half maximum (FWHM) is given by: FWHM $= 2\sqrt{2\ln(2)}\sigma$.
- `Lorentzian`: The Lorentzian function describes peaks that originate from lifetime broadening due to the finite lifetime of excited states. It has more pronounced tails compared to the Gaussian function. The Lorentzian function is defined as $L(x) = \frac{A}{\pi}\frac{\gamma}{(x-\mu)^2+\gamma^2}$, where - A is the peak amplitude, μ is the peak center, - and γ is the half-width at half-maximum (HWHM). The full width at half maximum (FWHM) is given by: FWHM $= 2\gamma$.
- `Voigt`: The Voigt function is a convolution of a Gaussian and a Lorentzian function. It provides a more accurate representation of XPS peaks by incorporating both instrumental broadening (Gaussian) and intrinsic lifetime broadening (Lorentzian). It is defined as $V(x;\sigma,\gamma) = \int_{-\infty}^{\infty} G(x')L(x-x')\,dx'$, where - $G(x')$ is the Gaussian function with standard deviation σ, -

and $L(x - x')$ is the Lorentzian function with half-width γ. The FWHM of the Voigt function is approximately: FWHM $\approx 0.5346 \cdot \text{FWHM}_L + \sqrt{0.2166 \cdot \text{FWHM}_L^2 + \text{FWHM}_G^2}$ where - $\text{FWHM}_L = 2\gamma$ (Lorentzian FWHM) and - $\text{FWHM}_G = 2\sqrt{2\ln(2)}\sigma$ (Gaussian FWHM).

- `Gaussian-Lorentzian Sum`: The Gaussian-Lorentzian Sum is an approximation to the Voigt function where the line shape is simply the sum of a Gaussian and a Lorentzian function $GL_{\text{sum}}(x) = \eta L(x) + (1-\eta)G(x)$, where η (ranging from 0 to 1) controls the relative contribution of the Lorentzian component.

- `Gaussian-Lorentzian Product`: The **Gaussian-Lorentzian Product** function provides an approximation to the true Voigt function by multiplying a Gaussian and a Lorentzian line shape. This model is often used to describe peak broadening due to both instrumental resolution (Gaussian component) and intrinsic lifetime effects (Lorentzian component). The function is defined as: $P(x, \eta) = (1 - \eta)G(x) \cdot \eta L(x)$ where: η is the **mixing parameter** ($0 \leq \eta \leq 1$), controlling the balance between the Gaussian and Lorentzian contributions. This method ensures that the resulting peak retains characteristics of both functions while avoiding excessive weight in either the Gaussian or Lorentzian component.

- `Asymmetric Lorentzian`: The Asymmetric Lorentzian function modifies the standard Lorentzian shape to asymmetry on either side of the peak. Oftentimes, the the asymmetric modification is achieved by introducing asymmetric broadening through power-law scaling on either side of the peak. The **asymmetric modification** is introduced using exponents α and β to alter the line shape differently for $x \leq E$ and $x > E$: $LA(\alpha, \beta) = \begin{cases} [L(x)]^\alpha, & x \leq E \\ [L(x)]^\beta, & x > E \end{cases}$ where: - $L(x)$ is the symmetric Lorentzian function. - α controls the asymmetry on the **low-energy** side ($x \leq E$). - β controls the asymmetry on the **high-energy** side ($x > E$). This asymmetric Lorentzian model is particularly useful in XPS to account for asymmetries arising from inelastic scattering processes and core-hole screening effects. Note that this is not the only possibility to introduce asymmetry. The exact functional form should be given by the `formula_description` field.

- `Doniach-Sunjic`: The Doniach-Sunjic function models asymmetric peaks arising from many-body interactions in XPS, particularly for core-level spectra influenced by conduction electrons. It is given by $DS(x) = A\frac{\cos\left(\pi\alpha/2 + (1-\alpha)\tan^{-1}\left(\frac{x-\mu}{\gamma}\right)\right)}{(x-\mu)^2+\gamma^2}^{(1-\alpha)/2}$, where α (0 leq α < 1) controls the asymmetry. When $\alpha = 0$, the function reduces to a symmetric Lorentzian.

- `Asymmetric Finite`: One major challenge in using asymmetric peaks is ensuring that the fitted line shape correctly models the peak intensity while being confined to physically meaningful integration limits. Standard asymmetric peaks, like the

`Lorentzian Asymmetric` function, may extend indefinitely, making it difficult to compare intensities when peak areas exceed the boundaries defined by background subtraction or experimental data constraints. The Asymmetric Finite function is an empirical modification of the asymmetric functions that introduces a damping parameter to gradually suppress the asymmetric tail, ensuring that peak intensity remains localized within the relevant spectral region. Typically, the asymmetric parameters are modified depending on the distance from the peak such that the overall area under the peak remains finite. This function is especially valuable when standard background approximations like **Shirley or Linear backgrounds** constrain the data to a specific range, making infinite asymmetric tails impractical. The functional form depends on the implementation and should be given by the `formula_description` field, if possible.

description: (recommended) *NX_CHAR* <=

Human-readable description of the peak fit function.

formula_description: (recommended) *NX_CHAR* <=

fit_parameters: (recommended) *NXparameters* <=

area: (optional) *NX_NUMBER* {units=*NX_ANY*}

Area of the peak.

width: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

Width of a peak at a defined fraction of the peak height.

Usually, this will be the Full Width at Half Maximum of the peak (FWHM). For asymmetric peaks, convenient measures of peak width are the half-widths of each side of the peak at half maximum intensity.

This concept is related to term 3.28 of the ISO 18115-1:2023 standard.

position: (optional) *NX_NUMBER* {units=*NX_ENERGY*}

Position of the peak on the energy axis.

backgroundBACKGROUND: (required) *NXpeak* <=

Functional form of one of the fitted XPS backgrounds.

This concept is related to term 3.21 of the ISO 18115-1:2023 standard.

label: (recommended) *NX_CHAR* <=

data: (required) *NXdata* <=

position: (required) *NX_NUMBER* {units=*NX_ENERGY*} <=

intensity: (required) *NX_NUMBER* <=

function: (recommended) *NXfit_function* <=

function_type: (required) *NX_CHAR* <=

Type of fit function used.

The user is encouraged to use one of the options defined in the enumeration, but in case none of these fit (e.g., in the case of very complex line shapes), a different value for the `function_type` field can be used. In that case in particular, but also if one of the suggested values is used, the functional form of the background should be given by the `formula_description` field. The user is also encouraged to use the `description` field for describing the fit function in a human-readable way.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `Linear`: Linear background, i.e., a simple straight line from the minimal to the maximal abscissa value. The Linear background is the simplest background model, which assumes a straight line between the minimum and maximum binding energy values. This model is used when there is a gradual, uniform increase or decrease in the background across the spectrum.
- `Shirley`: The Shirley background is based on the idea that the background intensity at any given binding energy is proportional to the total intensity of the peaks above that background in the lower binding energy range. Essentially, the Shirley background estimates the contribution of the background under each peak and adds them up to define the overall background shape. The Shirley background is commonly used in the analysis of high-resolution XPS data when peaks are located near edges or in regions of significant overlap with the background signal. While the exact mathematical formulation can be complex, the background is represented as an integral over the lower binding energy range to describe the cumulative contribution to the background: $B(x) = \int_{-\infty}^{x} \frac{I(\mu)}{1+\mu-x} d\mu$ where - $I(\mu)$ is the intensity of the peaks at binding energy μ, - x is the binding energy at the current point.
- `Tougaard`: The Tougaard background is based on the concept of universal cross-sections and is used for modeling the XPS background in situations where the background is integrated from the peak intensities at each binding energy to higher kinetic energies. It is useful when fitting the background in spectra that display significant low-energy tailing or when the background exhibits a nonlinear rise with binding energy. The model incorporates the notion of electron energy loss and the behavior of the photoelectrons as they travel through the material and lose energy. It is a more sophisticated background model compared to the linear and Shirley approaches. Mathematically, the Tougaard background can be described as: $B(x) = \int_{x_0}^{x} \sigma(E) I(E)\, dE$ where - $\sigma(E)$ is the universal cross-section at energy E, - $I(E)$ is the intensity at energy E, - x_0 is the threshold energy for the background.
- `Step Down`: The Step Down background is commonly used to fit data when a sharp decrease in the signal occurs at a certain binding energy, such as in the case of an abrupt edge or a "step-down" feature in the spectrum. It is modeled using a complementary error function (erf), which provides a smooth transition from the

higher intensity region to the lower intensity region. This model is useful when the data exhibits a sharp cutoff, for example when modelling the Fermi Edge. The mathematical expression for the step down background is: $B(x) = A \cdot \text{erfc}\left(\frac{x-x_0}{\sigma}\right)$ where - A is the amplitude, - x_0 is the threshold binding energy where the step occurs, - σ is the width of the transition region, - $\text{erfc}(x)$ is the complementary error function.

- `Step Up`: The Step Up background is similar to the Step Down background but is used to model a sharp increase in intensity, such as when a peak or feature rises suddenly above the baseline. This background is also modeled using the complementary error function, providing a smooth transition from a low intensity region to a high intensity region. The mathematical formulation for the step up background is: $B(x) = A \cdot \text{erf}\left(\frac{x-x_0}{\sigma}\right)$ where - A is the amplitude, - x_0 is the binding energy where the step up begins, - σ is the width of the transition, - $\text{erf}(x)$ is the error function.

description: (recommended) *NX_CHAR* <=

Human-readable description of the background fit function.

formula_description: (recommended) *NX_CHAR* <=

global_fit_function: (recommended) *NXfit_function* <=

function_type: (recommended) *NX_CHAR* <=

description: (recommended) *NX_CHAR* <=

formula_description: (recommended) *NX_CHAR* <=

error_function: (recommended) *NXfit_function* <=

function_type: (recommended) *NX_CHAR* <=

description: (recommended) *NX_CHAR* <=

formula_description: (recommended) *NX_CHAR* <=

SAMPLE: (required) *NXsample* <=

depends_on: (recommended) *NX_CHAR* <=

Reference to the transformation describing the orientation of the sample relative to a defined coordinate system.

transformations: (recommended) *NXtransformations* <=

sample_rotation_angle: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Rotation about the sample normal.

@**transformation_type**: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@**vector**: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@**depends_on**: (required) *NX_CHAR* <=

Obligatory value: `sample_normal_polar_angle_of_tilt`

sample_normal_polar_angle_of_tilt: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Polar tilt of the sample with respect to the upward z-direction, defined by the sample stage.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[-1, 0, 0]`

@depends_on: (required) *NX_CHAR* <=

Obligatory value: `sample_normal_tilt_azimuth_angle`

sample_normal_tilt_azimuth_angle: (required) *NX_NUMBER* {units=*NX_ANGLE*} <=

Azimuthal rotation of the sample from the y-direction defined by the sample stage.

@transformation_type: (required) *NX_CHAR* <=

Obligatory value: `rotation`

@vector: (required) *NX_NUMBER* <=

Obligatory value: `[0, 0, 1]`

@depends_on: (required) *NX_CHAR* <=

This should point to the coordinate system defined in /entry/xps_coordinate_system.

DATA: (required) *NXdata* <=

@energy_indices: (required) *NX_INT* <=

energy: (required) *NX_NUMBER* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxps/ENTRY-group*
- */NXxps/ENTRY/DATA-group*
- */NXxps/ENTRY/DATA/energy-field*
- */NXxps/ENTRY/DATA@energy_indices-attribute*
- */NXxps/ENTRY/definition-field*
- */NXxps/ENTRY/energy_referencing-group*
- */NXxps/ENTRY/FIT-group*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND-group*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/data-group*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/data/intensity-field*

- */NXxps/ENTRY/FIT/backgroundBACKGROUND/data/position-field*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/function-group*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/function/description-field*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/function/formula_description-field*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/function/function_type-field*
- */NXxps/ENTRY/FIT/backgroundBACKGROUND/label-field*
- */NXxps/ENTRY/FIT/data-group*
- */NXxps/ENTRY/FIT/data/fit_sum-field*
- */NXxps/ENTRY/FIT/data/input_dependent-field*
- */NXxps/ENTRY/FIT/data/input_independent-field*
- */NXxps/ENTRY/FIT/data/residual-field*
- */NXxps/ENTRY/FIT/error_function-group*
- */NXxps/ENTRY/FIT/error_function/description-field*
- */NXxps/ENTRY/FIT/error_function/formula_description-field*
- */NXxps/ENTRY/FIT/error_function/function_type-field*
- */NXxps/ENTRY/FIT/figure_of_meritMETRIC-field*
- */NXxps/ENTRY/FIT/figure_of_meritMETRIC@metric-attribute*
- */NXxps/ENTRY/FIT/global_fit_function-group*
- */NXxps/ENTRY/FIT/global_fit_function/description-field*
- */NXxps/ENTRY/FIT/global_fit_function/formula_description-field*
- */NXxps/ENTRY/FIT/global_fit_function/function_type-field*
- */NXxps/ENTRY/FIT/label-field*
- */NXxps/ENTRY/FIT/peakPEAK-group*
- */NXxps/ENTRY/FIT/peakPEAK/data-group*
- */NXxps/ENTRY/FIT/peakPEAK/data/intensity-field*
- */NXxps/ENTRY/FIT/peakPEAK/data/position-field*
- */NXxps/ENTRY/FIT/peakPEAK/function-group*
- */NXxps/ENTRY/FIT/peakPEAK/function/description-field*
- */NXxps/ENTRY/FIT/peakPEAK/function/fit_parameters-group*
- */NXxps/ENTRY/FIT/peakPEAK/function/fit_parameters/area-field*
- */NXxps/ENTRY/FIT/peakPEAK/function/fit_parameters/position-field*
- */NXxps/ENTRY/FIT/peakPEAK/function/fit_parameters/width-field*
- */NXxps/ENTRY/FIT/peakPEAK/function/formula_description-field*
- */NXxps/ENTRY/FIT/peakPEAK/function/function_type-field*
- */NXxps/ENTRY/FIT/peakPEAK/label-field*
- */NXxps/ENTRY/FIT/peakPEAK/relative_atomic_concentration-field*

- */NXxps/ENTRY/FIT/peakPEAK/total_area-field*
- */NXxps/ENTRY/INSTRUMENT-group*
- */NXxps/ENTRY/INSTRUMENT/beam_probe-group*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/depends_on-field*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations-group*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_azimuth_angle-field*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_azimuth_angle@depends_on-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_azimuth_angle@transformation_type-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_azimuth_angle@vector-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_direction-field*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_direction@depends_on-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_direction@vector-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_polar_angle_of_incidence-field*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_polar_angle_of_incidence@depends_on-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_polar_angle_of_incidence@transformation_type-attribute*
- */NXxps/ENTRY/INSTRUMENT/beam_probe/transformations/beam_polar_angle_of_incidence@vector-attribute*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER-group*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN-group*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/COLLECTIONCOLUMN/magnification-field*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/depends_on-field*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION-group*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/energy_scan_mode-field*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/ENERGYDISPERSION/radius-field*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations-group*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_azimuth_angle-field*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_azimuth_angle@depends_on-attribute*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_azimuth_angle@transformation_type attribute*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_azimuth_angle@vector-attribute*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_polar_angle-field*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_polar_angle@depends_on-attribute*

- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_polar_angle@transformation_type-attribute*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transformations/analyzer_take_off_polar_angle@vector-attribute*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/transmission_function-group*
- */NXxps/ENTRY/INSTRUMENT/ELECTRONANALYZER/work_function-field*
- */NXxps/ENTRY/INSTRUMENT/source_probe-group*
- */NXxps/ENTRY/INSTRUMENT/source_probe/power-field*
- */NXxps/ENTRY/method-field*
- */NXxps/ENTRY/SAMPLE-group*
- */NXxps/ENTRY/SAMPLE/depends_on-field*
- */NXxps/ENTRY/SAMPLE/transformations-group*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_polar_angle_of_tilt-field*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_polar_angle_of_tilt@depends_on-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_polar_angle_of_tilt@transformation_type-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_polar_angle_of_tilt@vector-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_tilt_azimuth_angle-field*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_tilt_azimuth_angle@depends_on-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_tilt_azimuth_angle@transformation_type-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_normal_tilt_azimuth_angle@vector-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_rotation_angle-field*
- */NXxps/ENTRY/SAMPLE/transformations/sample_rotation_angle@depends_on-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_rotation_angle@transformation_type-attribute*
- */NXxps/ENTRY/SAMPLE/transformations/sample_rotation_angle@vector-attribute*
- */NXxps/ENTRY/transitions-field*
- */NXxps/ENTRY/transmission_correction-group*
- */NXxps/ENTRY/xps_coordinate_system-group*
- */NXxps/ENTRY/xps_coordinate_system/depends_on-field*
- */NXxps/ENTRY/xps_coordinate_system/origin-field*
- */NXxps/ENTRY/xps_coordinate_system/TRANSFORMATIONS-group*
- */NXxps/ENTRY/xps_coordinate_system/x-field*
- */NXxps/ENTRY/xps_coordinate_system/y-field*
- */NXxps/ENTRY/xps_coordinate_system/z-field*
- */NXxps/ENTRY/xps_coordinate_system/z_direction-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxps.nxdl.xml

NXxrot

Status:

application definition, extends *NXxbase*

Description:

raw data from a rotation camera, extends *NXxbase*

This is the application definition for raw data from a rotation camera. It extends *NXxbase*, so the full definition is the content of *NXxbase* plus the data defined here.

Symbols:

The symbol(s) listed here will be used below to coordinate datasets with the same shape.

nP: Number of points

Groups cited:
NXattenuator, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXsample*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms.

Obligatory value: `NXxrot`

instrument: (required) *NXinstrument* <=

detector: (required) *NXdetector* <=

polar_angle: (required) *NX_FLOAT* {units=*NX_ANGLE*} <=

The polar_angle (two theta) where the detector is placed.

beam_center_x: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the x position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.

beam_center_y: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

This is the y position where the direct beam would hit the detector. This is a length, not a pixel position, and can be outside of the actual detector.

attenuator: (required) *NXattenuator* <=

attenuator_transmission: (required) *NX_FLOAT* {units=*NX_ANY*} <=

sample: (required) *NXsample* <=

rotation_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*} <=

This is an array holding the sample rotation start angle at each scan point

rotation_angle_step: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nP]) {units=*NX_ANGLE*}

This is an array holding the step made for sample rotation angle at each scan point

name: (required) *NXdata* <=

rotation_angle: *link* (suggested target: `/NXentry/NXsample/rotation_angle`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxrot/ENTRY-group*
- */NXxrot/ENTRY/definition-field*
- */NXxrot/ENTRY/instrument-group*
- */NXxrot/ENTRY/instrument/attenuator-group*
- */NXxrot/ENTRY/instrument/attenuator/attenuator_transmission-field*
- */NXxrot/ENTRY/instrument/detector-group*
- */NXxrot/ENTRY/instrument/detector/beam_center_x-field*
- */NXxrot/ENTRY/instrument/detector/beam_center_y-field*
- */NXxrot/ENTRY/instrument/detector/polar_angle-field*
- */NXxrot/ENTRY/name-group*
- */NXxrot/ENTRY/name/rotation_angle-link*
- */NXxrot/ENTRY/sample-group*
- */NXxrot/ENTRY/sample/rotation_angle-field*
- */NXxrot/ENTRY/sample/rotation_angle_step-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/applications/NXxrot.nxdl.xml

3.3.3 Contributed Definitions

A description of each NeXus contributed definition is given. NXDL files in the NeXus contributed definitions include propositions from the community for NeXus base classes or application definitions, as well as other NXDL files for long-term archival by NeXus. Consider the contributed definitions as either in *incubation* or a special case not for general use. The *NIAC: The NeXus International Advisory Committee* is charged to review any new contributed definitions and provide feedback to the authors before ratification and acceptance as either a base class or application definition.

These contributions are grouped together based on the research fields where these are typically used. Definitions that address multiple research fields are listed in each category:

Working with Samples

Conventions and Data Analysis

Computational and Constructive Solid Geometry

Atom Probe Microscopy

Optical Spectroscopy

Scanning Probe Microscopy

Transport Measurements

Microstructures Characterization and Representation

Complete List

Working with Samples

Application Definitions

Base classes

NXcontainer
State of a container holding the sample under investigation.

NXsubstance
A form of matter with a constant, definite chemical composition.

Data Analysis

Application Definitions

Base classes

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*.

NXsimilarity_grouping
Metadata to the results of a similarity grouping analysis.

Computational and Constructive Solid Geometry

Application Definitions

Base classes

NXcsg
Constructive Solid Geometry base class, using *NXquadric* and *NXoff_geometry*

NXquadric
Definition of a quadric surface

NXsolid_geometry
The head node for constructively defined geometry

Atom Probe Microscopy

Introduction

For the use case atom probe tomography the community contributed not only the *NXapm* application definition. It was also explored how using several instances of *NXprocess* is useful for documenting the many data processing steps that are typical in atom probe research to investigate structural features of the material demanding reconstructions, i.e., models of the crystal and defect network by analyzing the collective position data of atoms. The following definitions are a summary of the status quo how NeXus can be used for documenting these processing steps to improve numerical reproducibility and assist researchers with documenting procedural aspects of their data analysis workflows.

Base Classes

The processing steps of ranging and reconstructing are documented as two specializations of *NXprocess*:

NXapm_ranging
Metadata to ranging definitions made for a dataset in atom probe microscopy.

NXapm_reconstruction
Metadata of a dataset (tomographic reconstruction) in atom probe microscopy.

Spatial or other type of filters which are frequently used for atom probe to select specific atom positions or portions of the data based on isotopic identity are modeled as base classes for filters, which are defined atom-probe-agnostic empower reuse:

NXdelocalization
Base class to describe the delocalization of point-like objects on a grid.

NXisocontour
Computational geometry description of isocontouring/phase-fields in Euclidean space.

NXmatch_filter
Base class to filter ions based on their type or other descriptors like hit multiplicity.

NXspatial_filter
Base class to filter based on position. This base class takes advantage of *NXcg_ellipsoid*,

NXcg_cylinder*, *NXcg_hexahedron
Base classes to describe commonly used geometric primitives (not only) in atom probe. The primitives are used for defining the shape and extent of a region of interest (ROI) *NXroi_process* of material.

NXsubsampling_filter
Base class for a filter that can also be used for specifying how entries like ions can be filtered via sub-sampling.

Tools and applications in APM

There exist several research software tools in the APM community that deal with handling and analyzing APM data.

One of these is the paraprobe-toolbox The software is developed by M. Kühbach et al..

The paraprobe-toolbox is an example of an open-source parallelized software for analyzing point cloud data, for assessing meshes in 3D continuum space, and for studying the effects of parameterization on descriptors of micro- and nanoscale structural features (crystal defects) within materials when characterized and studied with atom probe.

There is a set of contributed application definitions describing each computational step in the paraprobe-toolbox. These were added to describe the whole workflow in this particular software, but can also act as a blueprint for how computational steps of other software tools (including commercial ones) could be developed further to benefit from NeXus.

The need for a thorough documentation of the tools was motivated by several needs:

First, users of software would like to better understand and also be able to study for themselves which individual parameters and settings for each tool exist and how configuring these affects analyses quantitatively. This stresses the aspect how to improve documentation.

Second, scientific software like paraprobe-toolbox implement numerical/algorithmical (computational) workflows whereby data coming from multiple input sources (like previous analysis results) are processed and carried through more involved analyses within several steps inside the tool. The tool then creates output as files. This provenance and workflow should be documented.

Individual tools of paraprobe-toolbox are developed in C/C++ and/or Python. Provenance tracking is useful as it is one component and requirement for making workflows exactly numerically reproducible and thus to enable reproducibility (the "R" of the FAIR principles of data stewardship).

For tools of the paraprobe-toolbox each workflow step is a pair or triple of sub-steps: 1. The creation of a configuration file. 2. The actual analysis using a given Python/or C/C++ tool from the toolbox. 3. The optional analyses/visualization of the results based on data in NeXus/HDF5 files generated by each tool.

Data and metadata between the tools are exchanged with NeXus/HDF5 files. This means that data inside HDF5 binary containers are named, formatted, and hierarchically structured according to NeXus application definitions.

In a refactoring project, within the FAIRmat project, which is part of the German National Research Data Infrastructure, the tools of the paraprobe-toolbox were modified to read from and write data using NeXus application definitions.

For example the application definition *NXapm_paraprobe_surfacer_config*: specifies the expectation how a configuration file for the paraprobe-surfacer tool is formatted and which parameters it contains including optionality and cardinality constraints.

Thereby, each config file uses a controlled vocabulary of terms. The config files store SHA256 checksum for each input file, thereby implementing an uninterrupted provenance tracking chain documenting the computational workflow.

As an example, a user may first range their reconstruction and then compute spatial correlation functions. The config file for the ranging tool stores the files which hold the reconstructed ion position and ranging definitions. The ranging tool generates a results file with the labels of each molecular ion. This results file is formatted according to the tool-specific *results* application definition. The generated results file and the reconstruction is imported by the spatial statistics tool which again keeps track of all files and reports its results in a spatial statistics tool results file.

This design makes it possible to rigorously trace which numerical results were achieved with specific inputs and settings using specifically-versioned tools. Noteworthy, this includes Y-junction on a graph which is where multiple input sources are combined to generate new results.

Defining, documenting, using, and sharing application definitions is a useful and future-proof strategy for software development and data analyses as it enables automated provenance tracking working silently in the background.

In summary, the following application definitions were defined for the paraprobe-toolbox. These are always pairs of application definitions — one for the configuration (input) side and one for the results (output) side. For each tool one such pair is proposed:

Application Definitions

NXapm_paraprobe_ranger_config*, *NXapm_paraprobe_ranger_results
Configuration and results respectively of the paraprobe-ranger tool. Apply ranging definitions and explore possible molecular ions. Store applied ranging definitions and combinatorial analyses of possible iontypes.

NXapm_paraprobe_surfacer_config*, *NXapm_paraprobe_surfacer_results
Configuration and results respectively of the paraprobe-surfacer tool. Create a model for the edge of a point cloud via convex hulls, alpha shapes, or alpha-wrappings. Store triangulated surface meshes of models for the edge of a dataset.

NXapm_paraprobe_distancer_config*, *NXapm_paraprobe_distancer_results
Configuration and results respectively of the paraprobe-distancer tool. Compute and store analytical distances between ions to a set of triangles.

NXapm_paraprobe_tessellator_config*, *NXapm_paraprobe_tessellator_results
Configuration and results respectively of the paraprobe-tessellator tool. Compute and store Voronoi cells and properties of these for all ions in a dataset.

NXapm_paraprobe_selector_config*, *NXapm_paraprobe_selector_results
Configuration and results respectively of the paraprobe-selector tool. Defining complex spatial regions-of-interest to filter reconstructed datasets. Store which points are inside or on the boundary of complex spatial regions-of-interest.

NXapm_paraprobe_spatstat_config, *NXapm_paraprobe_spatstat_results*
Configuration and results respectively of the paraprobe-spatstat tool. Compute spatial statistics on the entire or selected regions of the reconstructed dataset.

NXapm_paraprobe_nanochem_config, *NXapm_paraprobe_nanochem_results*
Configuration and results respectively of the paraprobe-nanochem tool. Compute delocalization, iso-surfaces, analyze 3D objects, composition profiles, and mesh interfaces.

NXapm_paraprobe_clusterer_config, *NXapm_paraprobe_clusterer_results*
Configuration and results respectively of the paraprobe-clusterer tool. Compute cluster analyses with established machine learning algorithms using CPU or GPUs.

NXapm_paraprobe_intersector_config, *NXapm_paraprobe_intersector_results*
Configuration and results resepctively of the paraprobe-intersector tool. Analyze volumetric intersections and proximity of 3D objects discretized as triangulated surface meshes in continuum space to study the effect the parameterization of surface extraction algorithms on the resulting shape, spatial arrangement, and colocation of 3D objects via graph-based techniques.

Joint work German NFDI consortia NFDI-MatWerk and FAIRmat

Members of the FAIRmat and the NFDI-MatWerk consortia of the German National Research Data Infrastructure are working together within the Infrastructure Use Case IUC09 of the NFDI-MatWerk project to work on examples how software tools in both consortia become better documented and interoperable to use. Within this project, we have also added the CompositionSpace tool by A. Saxena et al. that has been developed at the Max Planck Institute for Sustainable Materials in Düsseldorf

NXapm_compositionspace_config, *NXapm_compositionspace_results*
Results of a run with Alaukik Saxena's composition space tool.

Optical Spectroscopy

Introduction

Application definitions and *base classes* to describe optical spectroscopy experiments are already part of the NeXus standard. In addition, there are several contributed definitions that are currently under discussion.

Application Definitions

NXtransmission
Application definition for transmission experiments

Base Classes

These are new base classes to describe additional, yet to be standardized components of optical spectroscopy experiments.

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.

NXoptical_fiber
An optical fiber, e.g. glass fiber.

NXoptical_polarizer
An optical polarizer.

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 `eps = eps_inf + sum[A * lambda ** 2 / (lambda ** 2 - B ** 2)]` (Sellmeier formula). See the formula grammar below for an ebnf grammar for this form.

NXdispersion_single_parameter
This denotes a parameter which is used outside the summed part of a dispersion function, e.g. `eps_inf` in the formula example above.

NXdispersion_repeated_parameter
This denotes arrays of repeated parameters which are used to build a sum of parameter values, e.g. `A` and `B` are repeated parameters in the formula above.

NXdispersion_table
This describes a tabular dispersion where the permittivity is an array versus wavelength or energy.

Formula Grammar

Below you find a grammar to which the formula should adhere and which can be used to parse and evaluate the dispersion function. The terms `single_param_name` and `param_name` should be filled with the respective single and repeated params from the stored data. The grammer is written in the EBNF dialect of Lark, which is a parsing toolkit for python. It is easily translatable to general EBNF and other parser generator dialects. Here is a reference implementation in Rust/Python with a grammar written in lalrpop.

```
?assignment: "eps" "=" kkr_expression -> eps
          | "n" "=" kkr_expression -> n

?kkr_expression: expression
               | "<kkr>" "+" "1j" "*" term -> kkr_term

?expression: term
            | expression "+" term -> add
            | expression "-" term -> sub

?term: factor
      | term "*" factor -> mul
      | term "/" factor -> div

?factor: power
      | power "**" power -> power


?power: "(" expression ")"
      | FUNC "(" expression ")" -> func
      | "sum" "[" repeated_expression "]" -> sum_expr
      | NAME -> single_param_name
      | SIGNED_NUMBER -> number
      | BUILTIN -> builtin

?repeated_expression: repeated_term
                  | repeated_expression "+" repeated_term -> add
                  | repeated_expression "-" repeated_term -> sub


?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
```

Transport Phenomena

Introduction

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.

Application Definitions

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:

NXsensor_scan
: Application definition for a generic scan using sensors.

NXiv_temp
: Application definition for temperature-dependent current-voltage (IV) curve measurements.

Microstructure Characterization and Representation

Introduction

The internal structure of a material modeled as crystals and the defect network that connect these.

Application Definitions

Base classes

NXmicrostructure
: Base class to describe elements of the microstructure of a material.

NXmicrostructure_pf*, *NXmicrostructure_ipf*, *NXmicrostructure_odf
: Base classes for describing parameterization, results, and data from texture analysis, specifically pole figure (pf), inverse pole figure (ipf), and orientation distribution function (odf), respectively.

NXmicrostructure_feature
: Set of topological/spatial features in materials built from atoms, from coarse-grained representations of atoms, or from other microstructure features.

NXmicrostructure_slip_system
: Base class for describing a set of crystallographic slip systems.

NXmicrostructure_mtex_config
: Base class for documenting the parameterization of MTex, which is a software for analyzing material texture written in MATLAB.

NXmicrostructure_score_config
: Application definition to control a simulation with the SCORE cellular automata simulation tool.

NXmicrostructure_score_results
: Application definition for storing results of the SCORE cellular automata simulation tool.

NXmicrostructure_kanapy_results
Application definition for storing results of the kanapy microstructure synthesis tool.

Scanning Probe Microscopy

Scanning probe microscopy (SPM) is a branch of microscopes that forms images of surfaces using a physical probe that scans the specimen. Using proper instrument setup, SPM can be used to measure various properties of the material surface, such as its topography, magnetic and electric properties, and chemical composition.

Fig. 14: A schematic diagram of Scanning Tunneling Microscopy (STM) and Spectroscopy (STS) setup generate topography image and I/V curve.

The application definition NXspm provides a shared structural framework for capturing essential components such as the instrument configuration, experimental, and sample environments, scan data acquired during measurements, and other relevant metadata. The current version aims to ensure that fundamental technical elements are inherited consistently across specific experiment types like STM, STS, and AFM.

Fig. 15: Inheritance relationship among SPM application definitions.

Application Definition

***NXspm*:**
An application definition for scanning Probe Microscopy domain experiments. The *NXspm* in herited from the *NXsensor_scan* is considered as a generic structure for all SPM experiments. The App. Def. *NXsts* is also capable to handle *NXsts* application definition considering STS as a fundamental experiment of SPM family.

***NXsts*:**
The application definition NXsts for Scanning Tunneling Spectroscopy is proxy to be used for STS experiments and it is inherited from the *NXspm*. The *NXsts* is an alias of the *NXspm* application definition.

***NXstm*:**
An application definition for Scanning Tunneling Microscopy experiments inherited from the *NXspm*.

***NXafm*:**
An application definition for Atomic Force Microscopy experiments inherited from the *NXspm*.

Base Classes

NXlockin: A base class to describe lock-in amplifier instrument.

NXspm_bias_spectroscopy: A base class to describe bias spectroscopy measurement to measure I/V curve in STS expriment.

NXspm_cantilever: A base class to characterize cantilever used in AFM experiments.

NXspm_cantilever_config: A base class to describe cantilever configuration in AFM experiments.

NXspm_cantilever_oscillator: A base class to describe cantilever oscillator in AFM experiments.

NXphase_lock_loop: A base class to describe phase lock loop in AFM experiments.

NXspm_piezo_sensor: A base class to describe piezo sensor in SPM experiments.

NXspm_piezo_config: A base class to describe piezo configuration in SPM experiments.

NXspm_piezoelectric_material: A base class to draw piezoelectric material properties used in cantilever tip.

NXspm_positioner: A base class to describe PID positioner in SPM experiments.

NXspm_scan_control: A base class to characterize the movement of scan probe in a multi-dimensional phase space.

NXspm_scan_pattern: A base class to define the pattern of a scan in a given scan region.

NXspm_scan_region: A base class to define the phase space or sub-phase space for scan in SPM experiments.

NXspm_temperature_sensor: A base class to describe temperature sensor in SPM experiments.

Contributed Definitions

This is the complete list of contributed definitions:

NXafm
An application definition to describe atomic force microscopy (AFM).

NXamplifier
Base classed definition for amplifier devices.

NXapm_compositionspace_config
Application definition for a configuration of the CompositionSpace tool used in atom probe.

NXapm_compositionspace_results
Application definition for results of the CompositionSpace tool used in atom probe.

NXapm_paraprobe_clusterer_config
Application definition for a configuration file of the paraprobe-clusterer tool.

NXapm_paraprobe_clusterer_results
Application definition for a results file of the paraprobe-clusterer tool.

NXapm_paraprobe_distancer_config
Application definition for a configuration file of the paraprobe-distancer tool.

NXapm_paraprobe_distancer_results
Application definition for a results file of the paraprobe-distancer tool.

NXapm_paraprobe_intersector_config
Application definition for a configuration file of the paraprobe-intersector

NXapm_paraprobe_intersector_results
Application definition for results files of the paraprobe-intersector tool.

NXapm_paraprobe_nanochem_config
Application definition for a configuration file of the paraprobe-nanochem tool.

NXapm_paraprobe_nanochem_results
Application definition for a results file of the paraprobe-nanochem tool.

NXapm_paraprobe_ranger_config
Application definition for a configuration file of the paraprobe-ranger tool.

NXapm_paraprobe_ranger_results
Application definition for results files of the paraprobe-ranger tool.

NXapm_paraprobe_selector_config
Application definition for a configuration file of the paraprobe-selector tool.

NXapm_paraprobe_selector_results
Application definition for a results file of the paraprobe-selector tool.

NXapm_paraprobe_spatstat_config
Application definition for a configuration file of the paraprobe-spatstat tool.

NXapm_paraprobe_spatstat_results
Application definition for a results file of the paraprobe-spatstat tool.

NXapm_paraprobe_surfacer_config
Application definition for a configuration file of the paraprobe-surfacer tool.

NXapm_paraprobe_surfacer_results
Application definition for a results file of the paraprobe-surfacer tool.

NXapm_paraprobe_tessellator_config
Application definition for a configuration file of the paraprobe-tessellator tool.

NXapm_paraprobe_tessellator_results
Application definition for a results file of the paraprobe-tessellator tool.

NXapm_paraprobe_tool_common
Base class documenting organizational metadata used by all tools of the

NXapm_paraprobe_tool_config
Application definition for a (configuration) file of a tool from the paraprobe-toolbox.

NXapm_paraprobe_tool_parameters
Base class documenting parameters for processing used by all tools of the

NXapm_paraprobe_tool_process
Base class documenting a processing step within a tool of the paraprobe-toolbox.

NXapm_paraprobe_tool_results
Application definition for storing processing results of a tool from the paraprobe-toolbox.

NXbeam_splitter
A beam splitter, i.e. a device splitting the light into two or more beams.

NXcontainer
State of a container holding the sample under investigation.

NXcsg
Constructive Solid Geometry (CSG) base class.

NXcxi_ptycho
Application definition for a ptychography experiment, compatible with CXI from version 1.6.

NXdelocalization
Base class of the configuration and results of a delocalization algorithm.

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
An application definition for describing a dispersive material.

NXelectrostatic_kicker
Base class for an electrostatic kicker.

NXem_calorimetry
Application definition for minimal example in-situ calorimetry.

NXisocontour
Base class for describing isocontouring/phase-fields in Euclidean space.

NXiv_temp
Application definition for temperature-dependent IV curve measurements.

NXlockin
A base class definition for a lock-in amplifier.

NXmagnetic_kicker
Base class for a magnetic kicker.

NXmatch_filter
Base class of a filter to select members of a set based on their identifier.

NXmicrostructure
Base class to describe a microstructure, its structural aspects, associated descriptors, properties.

NXmicrostructure_feature
Base class for documenting structuring features of a microstructure.

NXmicrostructure_ipf
Base class to store an inverse pole figure (IPF) mapping (IPF map).

NXmicrostructure_kanapy_results
Application definition for the microstructure generator kanapy from ICAMS Bochum.

NXmicrostructure_mtex_config
Base class to store the configuration when using the MTex/Matlab software.

NXmicrostructure_odf
Base class to store an orientation distribution function (ODF).

NXmicrostructure_pf
Base class to store a pole figure (PF) computation.

NXmicrostructure_score_config
Application definition to configure a simulation with the SCORE model.

NXmicrostructure_score_results
Application definition for storing results of the SCORE cellular automata model.

NXmicrostructure_slip_system
Base class for describing a set of crystallographic slip systems.

NXoptical_fiber
An optical fiber, e.g. glass fiber.

NXoptical_polarizer
An optical polarizer.

NXphase_lock_loop
A base class definition for a phase-locked loop (PLL) used in AFM experiments.

NXquadric
Definition of a quadric surface.

NXquadrupole_magnet
Base class for a quadrupole magnet.

NXrcs
A base class for the Real Time Control System (RCS).

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
Base class for an electrostatic separator.

NXsimilarity_grouping
Base class to store results obtained from applying a similarity grouping (clustering) algorithm.

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
Base class for a spatial filter for objects within a region-of-interest (ROI).

NXspin_rotator
Base class for a spin rotator.

NXspm
Scanning Probe Microscopy (SPM) is a branch of microscopy that utilizes a physical probe to scan the surface of

NXspm_bias_spectroscopy
A base class for bias spectroscopy to describe the change in the physical properties

NXspm_cantilever
A base class to describe the cantilever used in Atomic Force Microscopy (AFM).

NXspm_cantilever_config
This file defines the NXspm_cantilever_config base class, which contains configuration

NXspm_cantilever_oscillator
In generally speaking a cantilever resembles a leaf-spring which can be treated as a

NXspm_piezo_config
A base class describing piezo actuator settings for scanning probe microscopy.

NXspm_piezo_sensor
This piezo sensor group refers to the height (or Z) piezo sensor.

NXspm_piezoelectric_material
Description and properties of the piezoelectric actuator materials.

NXspm_positioner
An extension of positioner, used to maintain a measurement signal through

NXspm_scan_control
A scan is performed inside an N-dimensional phase space, where each dimension can correspond not only to real space coordinates (x,y) but also to any other parameter. This class contains detailed information about controlling the scan in such a phase space (or its subspace).

NXspm_scan_pattern
Basic base class to define the pattern of a scan in a given scan region.

NXspm_scan_region
The scan region is the area of phase space or sub-phase space where the scan is

NXspm_temperature_sensor
The *NXspm_temperature_sensor* class describes a temperature sensor

NXstm
An application definition to describe Scanning Tunneling Microscopy (STM).

NXsts
An application definition to describe Scanning Tunneling Spectroscopy (STS).

NXsubsampling_filter
Base class of a filter to sample members in a set based on their indices.

NXsubstance
A form of matter with a constant, definite chemical composition.

NXtransmission
Application definition for transmission experiments

NXxpcs
X-ray Photon Correlation Spectroscopy (XPCS) data (results).

NXxrd
NXxrd on top of NXmonopd

NXxrd_pan
NXxrd_pan is a specialization of NXxrd with extra properties

NXafm

Status:

application definition (contribution), extends *NXspm*

Description:

An application definition to describe atomic force microscopy (AFM).

Symbols:

No symbol table

Groups cited:
NXcollection, *NXdetector*, *NXentry*, *NXenvironment*, *NXinstrument*, *NXphase_lock_loop*, *NXpositioner*, *NXsensor*, *NXspm_cantilever_oscillator*, *NXspm_cantilever*, *NXspm_piezo_sensor*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Name of the definition that is used for the application.

Obligatory value: `NXafm`

experiment_technique: (required) *NX_CHAR* <=

The AFM technique.

Obligatory value: `AFM`

scan_mode: (required) *NX_CHAR* <=

The mode of the scan.

contact mode: Cantilever attempts to move on the sample surface in very close contact with the sample.
The cantilever deflection is usually employed to control the cantilever position using a PID feedback loop.

non-contact mode: Cantilever attempts to oscillate above the sample surface. Cantilever attempts to stay in the
interaction (atomic force) zone therefore cantilever oscillator amplitude and frequency are deformed. The cantilever frequency or oscillation amplitude is usually employed to control the cantilever position using a PID feedback loop.

tapping mode: Resembles to the non-contact mode, but at every point of scan the cantilever tip comes closer
to the sample surface ("taps it"). The cantilever oscillation amplitude is usually employed to control the cantilever position using a PID feedback loop.

peak force tapping mode: Like the tapping mode, but at each point of the scan force-distance curve is recorded.
The maximum force is usually employed to control the cantilever position using a PID feedback loop.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `contact mode`
- `tapping mode`
- `non-contact mode`
- `peak force tapping mode`

INSTRUMENT: (required) *NXinstrument* <=

photo_detector: (optional) *NXdetector* <=

Information about the quadrant photodiode deflection detector or interferometer.

SENSOR: (optional) *NXsensor*

The sensor information for the photodiode or interferometer.

SPM_CANTILEVER: (recommended) *NXspm_cantilever*

The cantilever information.

cantilever_oscillator: (recommended) *NXspm_cantilever_oscillator* <=

When a cantilever is oscillated close to its resonance, this describes the oscillator properties.

A cantilever can be used in direct contact mode to detect interaction forces or oscillated close to its resonance frequency. Changes in the oscillation amplitude, phase (between oscillated tail and moving tip) or resonance frequency are very sensitive to changes in the interaction potential field, giving rise of various measurement modes, such as non-contact or intermittent-contact (tapping) modes.

reference_amplitude: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

reference_frequency: (recommended) *NX_NUMBER* {units=*NX_FREQUENCY*} <=

reference_phase: (recommended) *NX_NUMBER* {units=*NX_ANGLE*} <=

phase_positioner: (recommended) *NXpositioner* <=

The phase positioner of the cantilever.

amplitude_positioner: (recommended) *NXpositioner* <=

The amplitude positioner of the cantilever.

phase_lock_loop: (recommended) *NXphase_lock_loop*

Phase locked loop for cantilever lock-in device.

SCAN_ENVIRONMENT: (required) *NXenvironment* <=

The environment information.

XY**piezo_sensor**: (optional) *NXspm_piezo_sensor* <=

This should be a link to the concept entry/instrument/xy_piezo_sensor.

head_temperature_sensor: (optional) *NXsensor* <=

This should be a link to the concept entry/instrument/head_temperature_sensor.

head_temperature_sensor: (optional) *NXsensor*

The temperature of the scan environment or head of the cantilever.

reproducibility_indicators: (optional) *NXcollection* <=

The group of indicators (links to the existing fields in different groups) that measure the reproducibility of the experiment.

head_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/head_temperature

cryo_bottom_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_bottom_temperature

cryo_shield_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_shield_temperature

cantilever_oscillator: (optional) *NXspm_cantilever_oscillator*

This should be a link to /entry/instrument/cantilever_spm/cantilever_oscillator

resolution_indicators: (optional) *NXcollection* <=

The group of indicators (links to the existing fields in different groups) that

head_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/head_temperature

cryo_bottom_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_bottom_temperature

cryo_shield_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_shield_temperature

cantilever_config: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/cantilever_spm/cantilever_config

amplitude_excitation: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/phase_lock_loop/amplitude_excitation

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXafm/ENTRY-group*
- */NXafm/ENTRY/definition-field*
- */NXafm/ENTRY/experiment_technique-field*
- */NXafm/ENTRY/INSTRUMENT-group*
- */NXafm/ENTRY/INSTRUMENT/head_temperature_sensor-group*
- */NXafm/ENTRY/INSTRUMENT/phase_lock_loop-group*
- */NXafm/ENTRY/INSTRUMENT/photo_detector-group*
- */NXafm/ENTRY/INSTRUMENT/photo_detector/SENSOR-group*
- */NXafm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT-group*
- */NXafm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/head_temperature_sensor-group*
- */NXafm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/XYpiezo_sensor-group*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER-group*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER/amplitude_positioner-group*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER/cantilever_oscillator-group*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER/cantilever_oscillator/reference_amplitude-field*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER/cantilever_oscillator/reference_frequency-field*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER/cantilever_oscillator/reference_phase-field*
- */NXafm/ENTRY/INSTRUMENT/SPM_CANTILEVER/phase_positioner-group*
- */NXafm/ENTRY/reproducibility_indicators-group*
- */NXafm/ENTRY/reproducibility_indicators/cantilever_oscillator-group*
- */NXafm/ENTRY/reproducibility_indicators/cryo_bottom_temperature-field*
- */NXafm/ENTRY/reproducibility_indicators/cryo_shield_temperature-field*
- */NXafm/ENTRY/reproducibility_indicators/head_temperature-field*
- */NXafm/ENTRY/resolution_indicators-group*
- */NXafm/ENTRY/resolution_indicators/amplitude_excitation-field*
- */NXafm/ENTRY/resolution_indicators/cantilever_config-field*
- */NXafm/ENTRY/resolution_indicators/cryo_bottom_temperature-field*
- */NXafm/ENTRY/resolution_indicators/cryo_shield_temperature-field*

- */NXafm/ENTRY/resolution_indicators/head_temperature-field*
- */NXafm/ENTRY/scan_mode-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXafm.nxdl.xml

NXamplifier

Status:

base class (contribution), extends *NXcomponent*

Description:

Base classed definition for amplifier devices.

Symbols:

No symbol table

Groups cited:
NXfabrication

Structure:

classification: (optional) *NX_CHAR*

Type of the amplifier base on the response on frequency.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `low-pass`: A low-pass filter is a filter that passes signals with a frequency lower than a certain cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency.
- `high-pass`: A high-pass filter is a filter that passes signals with a frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency.
- `band-pass`: A band-pass filter is a filter that passes signals within a certain frequency range and attenuates signals outside that range.
- `band-stop`: A band-stop filter is a filter (opposite to the band-pass) that attenuates signals within a certain frequency range and passes signals outside that range.
- `broadband`: A broadband filter is a filter that passes signals over a wide range of frequencies with minimal attenuation.

num_of_channels: (optional) *NX_NUMBER*

The number of preamplifier channels are assigned.

active_channels: (optional) *NX_NUMBER*

The number of preamplifier channels are ready to be used. (array for active channels)

openloop_amplification: (optional) *NX_NUMBER*

The output signal does not go through a feedback loop to adjust the amplification of the amplifier. (array for active channels)

signal_over_noise: (optional) *NX_NUMBER*

The ratio of the amplitude of the target signal to the amplitude of the noise in the output signal of the amplifier. S/N=V_signal/V_noise. (array for active channels)

crosstalk_factor: (optional) *NX_NUMBER* {units=db}

The unwanted coupling between different channels (if active >1). In ideal amplifier, channels are independent of each other, But due to different resources sharing (e.g., same power supply, electromagnetic interference), there may have some unwanted coupling between different channels, which is called crosstalk.

crosstalk_compensation: (optional) *NX_BOOLEAN*

If measures are taken for reducing interferences between different signalling pathways.

bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The spectrum of frequency it can amplify, from its lowest to highest frequency limits. If it is difference of the frequencies, please also provide :ref: *center_frequency </NXamplifier/center_frequency-field>*.

center_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The frequency in the middle of the bandwidth.

lower_cutoff_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The lower frequency point of the bandwidth where gain drops significantly (e.g., -3dB point).

upper_cutoff_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The upper frequency point of the bandwidth where gain drops significantly (e.g., -3dB point).

gain: (optional) *NX_NUMBER*

The ratio of the output signal to the input signal of the amplifier.

hardware: (optional) *NXfabrication <=*

(IC, device) (NXmanufacturer?)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXamplifier/active_channels-field*
- */NXamplifier/bandwidth-field*
- */NXamplifier/center_frequency-field*
- */NXamplifier/classification-field*
- */NXamplifier/crosstalk_compensation-field*
- */NXamplifier/crosstalk_factor-field*
- */NXamplifier/gain-field*
- */NXamplifier/hardware-group*
- */NXamplifier/lower_cutoff_frequency-field*
- */NXamplifier/num_of_channels-field*
- */NXamplifier/openloop_amplification-field*
- */NXamplifier/signal_over_noise-field*
- */NXamplifier/upper_cutoff_frequency-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXamplifier.nxdl.xml

NXapm_compositionspace_config

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for a configuration of the CompositionSpace tool used in atom probe.

- A. Saxena et al.

This is an application definition for the common NFDI-MatWerk/FAIRmat infrastructure use case IUC09 that explores how to improve the organization and results storage of the CompositionSpace tool by using the NeXus data model and semantics.

Symbols:

No symbol table

Groups cited:
NXentry, *NXnote*, *NXparameters*, *NXprocess*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_compositionspace_config`

@version: (optional) *NX_CHAR* <=

identifier_analysis: (recommended) *NX_UINT*

reconstruction: (required) *NXnote* <=

Specification of the tomographic reconstruction used for this analysis.

Reconstructions in the field of atom probe tomography are communicated via a file which stores the reconstructed position and mass-to-charge-state-ratio value for each ion.

Container file formats like HDF5, such as NeXus/HDF5 files using *NXapm*, can store multiple reconstructions. In this case, the position and mass_to_charge concepts point to specific instances in the file referred to by file_name for the analysis with CompositionSpace.

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

position: (required) *NX_CHAR*

Name of the node which resolves the reconstructed ion position values to use for this analysis.

mass_to_charge: (optional) *NX_CHAR*

Name of the node which resolves the mass-to-charge-state-ratio values for each reconstructed ion to use for this analysis.

ranging: (required) *NXnote* <=

Specification of the ranging definitions used for this analysis.

Ranging definitions in the field of atom probe tomography are communicated via a file which stores the mass-to-charge-state-ratio interval and the number of elements of which each (molecular) ion is composed. These values are stored for each ion.

Container file formats like HDF5, such as NeXus/HDF5 files using *NXapm*, can store multiple ranging definitions.

Indices of ions start from 1. The value 0 is reserved for the null model of unranged positions whose iontype is referred to as the unknown_type. The value 0 is also reserved for voxels that lie outside the dataset.

file_name: (required) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

ranging_definitions: (required) *NX_CHAR*

Name of that (parent) node whose child stores the ranging definitions that are applied in this analysis with CompositionSpace.

voxelization: (required) *NXprocess* <=

Step during which the point cloud is discretized to compute element-specific composition fields. Iontypes are atomically decomposed to correctly account for the multiplicity of each element that was ranged for each ion.

edge_length: (required) *NX_NUMBER* {units=*NX_LENGTH*}

Edge length of cubic voxels building the 3D grid that is used for discretizing the point cloud.

autophase: (required) *NXprocess*

Optional step during which the subsequent segmentation step is prepared with the aim to eventually reduce the dimensionality of the chemical space in which the machine learning model works.

In this step a supervised reduction of the dimensionality of the chemical space is quantified using the (Gini) feature importance of each element to suggest which columns of the composition matrix should be taken for the subsequent segmentation step.

use: (required) *NX_BOOLEAN*

Was the automated phase assignment used?

initial_guess: (required) *NX_POSINT* {units=*NX_UNITLESS*}

Estimated guess for which a Gaussian mixture model is evaluated to preprocess a result that is subsequently post-processed with a random_forest_classifier to lower the number of dimensions in the chemical space to the subset of trunc_species many elements with the highest feature importance.

trunc_species: (required) *NX_POSINT* {units=*NX_UNITLESS*}

The number of elements to use for reducing the dimensionality.

random_forest_classifier: (optional) *NXprocess*

Configuration for the random forest classification model.

segmentation: (required) *NXprocess* <=

Step during which the voxel set is segmented into voxel sets with different chemical composition. **pca**: (optional) *NXprocess*

A principal component analysis of the chemical space to guide a decision into how many sets of voxels with different chemical composition the machine learning algorithm suggests to split the voxel set.

ic_opt: (required) *NXprocess*

The decision is guided through the evaluation of the information criterion minimization.

n_max_ic_cluster: (required) *NX_POSINT* {units=*NX_UNITLESS*}

The maximum number of chemical classes to probe with the Gaussian mixture model with which the voxel set is segmented into a mixture of voxels with that many different chemical compositions.

gaussian_mixture: (optional) *NXprocess*

Configuration for the Gaussian mixture model that is used in the segmentation step.

clustering: (required) *NXprocess* <=

Step during which the chemically segmented voxel sets are analyzed for their spatial organization. **dbscan**: (required) *NXparameters* <=

Configuration for the DBScan algorithm that is used in the clustering step.

eps: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The maximum distance between voxel pairs in a neighborhood to be considered connected.

min_samples: (required) *NX_UINT* {units=*NX_UNITLESS*}

The number of voxels in a neighborhood for a voxel to be considered as a core point.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_compositionspace_config/ENTRY-group*
- */NXapm_compositionspace_config/ENTRY/clustering-group*
- */NXapm_compositionspace_config/ENTRY/clustering/dbscan-group*
- */NXapm_compositionspace_config/ENTRY/clustering/dbscan/eps-field*
- */NXapm_compositionspace_config/ENTRY/clustering/dbscan/min_samples-field*
- */NXapm_compositionspace_config/ENTRY/definition-field*
- */NXapm_compositionspace_config/ENTRY/definition@version-attribute*
- */NXapm_compositionspace_config/ENTRY/identifier_analysis-field*
- */NXapm_compositionspace_config/ENTRY/ranging-group*

- */NXapm_compositionspace_config/ENTRY/ranging/algorithm-field*
- */NXapm_compositionspace_config/ENTRY/ranging/checksum-field*
- */NXapm_compositionspace_config/ENTRY/ranging/file_name-field*
- */NXapm_compositionspace_config/ENTRY/ranging/ranging_definitions-field*
- */NXapm_compositionspace_config/ENTRY/reconstruction-group*
- */NXapm_compositionspace_config/ENTRY/reconstruction/algorithm-field*
- */NXapm_compositionspace_config/ENTRY/reconstruction/checksum-field*
- */NXapm_compositionspace_config/ENTRY/reconstruction/file_name-field*
- */NXapm_compositionspace_config/ENTRY/reconstruction/mass_to_charge-field*
- */NXapm_compositionspace_config/ENTRY/reconstruction/position-field*
- */NXapm_compositionspace_config/ENTRY/segmentation-group*
- */NXapm_compositionspace_config/ENTRY/segmentation/ic_opt-group*
- */NXapm_compositionspace_config/ENTRY/segmentation/ic_opt/gaussian_mixture-group*
- */NXapm_compositionspace_config/ENTRY/segmentation/ic_opt/n_max_ic_cluster-field*
- */NXapm_compositionspace_config/ENTRY/segmentation/pca-group*
- */NXapm_compositionspace_config/ENTRY/voxelization-group*
- */NXapm_compositionspace_config/ENTRY/voxelization/autophase-group*
- */NXapm_compositionspace_config/ENTRY/voxelization/autophase/initial_guess-field*
- */NXapm_compositionspace_config/ENTRY/voxelization/autophase/random_forest_classifier-group*
- */NXapm_compositionspace_config/ENTRY/voxelization/autophase/trunc_species-field*
- */NXapm_compositionspace_config/ENTRY/voxelization/autophase/use-field*
- */NXapm_compositionspace_config/ENTRY/voxelization/edge_length-field*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_compositionspace_config.nxdl.xml

NXapm_compositionspace_results

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for results of the CompositionSpace tool used in atom probe.

- A. Saxena et al.

This is an application definition for the common NFDI-MatWerk/FAIRmat infrastructure use case IUC09 that explores how to improve the organization and results storage of the CompositionSpace software using NeXus.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

grid_dim: The dimensionality of the grid.

n_voxels: Total number of voxels.

n_ions: Total number of ions in the reconstructed dataset.

Groups cited:
NXatom, *NXcg_grid*, *NXcollection*, *NXcs_profiling*, *NXdata*, *NXentry*, *NXnote*, *NXprocess*, *NXprogram*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_compositionspace_results`

@version: (optional) *NX_CHAR* <=

identifier_analysis: (recommended) *NX_UINT*

profiling: (optional) *NXcs_profiling*

current_working_directory: (recommended) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

program1: (required) *NXprogram*

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: (required) *NXnote* <=

Configuration file that was used in this analysis.

file_name: (required) *NX_CHAR* <=

algorithm: (recommended) *NX_CHAR* <=

checksum: (recommended) *NX_CHAR* <=

userID: (optional) *NXuser* <=

specimen: (recommended) *NXsample* <=

Contextualize back to the specimen from which the dataset was collected that was here analyzed with CompositionSpace tool.

is_simulation: (required) *NX_BOOLEAN*

True, if the specimen that the reconstructed dataset describes is a simulated one. False, if the specimen that the reconstructed dataset describes is a real one.

atom_types: (required) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the specimen. If the specimen 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 data instances.

voxelization: (optional) *NXprocess* <=

Step during which the point cloud is discretized to compute element-specific composition fields. Iontypes are atomically decomposed to correctly account for the multiplicity of each element that was ranged for each ion.

Using a discretization grid that is larger than the average distance between reconstructed ion positions reduces computational costs. This is the key idea of the CompositionSpace tool compared to other methods used in atom probe for characterizing microstructural features that use the ion position data directly.

sequence_index: (required) *NX_POSINT* <=

Obligatory value: `1`

weight: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_voxels]) {units=*NX_UNITLESS*}

Total number of weight (counts for discretization with a rectangular transfer function) for the occupancy of each voxel with atoms.

grid: (required) *NXcg_grid*

dimensionality: (required) *NX_POSINT* <=

Obligatory value: `3`

cardinality: (required) *NX_POSINT* <=

origin: (required) *NX_NUMBER* (Rank: 1, Dimensions: [grid_dim]) <=

symmetry: (required) *NX_CHAR* <=

Obligatory value: `cubic`

cell_dimensions: (required) *NX_NUMBER* (Rank: 1, Dimensions: [grid_dim]) <=

extent: (required) *NX_POSINT* (Rank: 1, Dimensions: [grid_dim])

index_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

position: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_voxels, grid_dim]) {units=*NX_LENGTH*} <=

Position of each cell in Euclidean space.

coordinate: (required) *NX_INT* (Rank: 2, Dimensions: [n_voxels, grid_dim]) {units=*NX_DIMENSIONLESS*} <=

Discrete coordinate of each voxel.

indices_voxel: (required) *NX_INT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

For each ion, the identifier of the voxel into which the ion binned.

ionID: (required) *NXatom*

name: (required) *NX_CHAR* <=

Chemical symbol of the element from the periodic table.

weight: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_voxels]) {units=*NX_UNITLESS*}

Element-specific weight (counts for discretization with a rectangular transfer function) for the occupancy of each voxel with atoms of this element.

autophase: (optional) *NXprocess* <=

Optional step during which the subsequent segmentation step is prepared to improve the segmentation.

sequence_index: (required) *NX_POSINT* <=

Obligatory value: 2

result: (required) *NXdata* <=

@**signal**: (required) *NX_CHAR* <=

@**axes**: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

axis_feature_indices: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

Element identifier stored sorted in descending order of feature importance.

@**long_name**: (required) *NX_CHAR*

Axis caption

axis_feature_importance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

Element relative feature importance stored sorted in descending order of feature importance.

@**long_name**: (required) *NX_CHAR*

Axis caption

segmentation: (optional) *NXprocess* <=

Step during which the voxel set is segmented into voxel sets with different chemical composition. **pca**: (required) *NXprocess*

PCA in the chemical space (essentially composition correlation analyses).

sequence_index: (required) *NX_POSINT* <=

Any of these values: 2 | 3

result: (required) *NXdata* <=

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

axis_explained_variance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_DIMENSIONLESS*}

Explained variance values

axis_pca_dimension: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Elements identifier matching those from ENTRY/voxelization/ionID used during the principal component analysis.

ic_opt: (required) *NXprocess*

Information criterion minimization.

sequence_index: (required) *NX_POSINT* <=

Any of these values: `3 | 4`

cluster_analysisID: (required) *NXprocess*

Results of the Gaussian mixture analysis for n_components equal to n_ic_cluster.

n_ic_cluster: (required) *NX_UINT* {units=*NX_UNITLESS*}

n_components argument of the Gaussian mixture model.

y_pred: (required) *NX_UINT* (Rank: 1, Dimensions: [n_voxels]) {units=*NX_UNITLESS*}

y_pred return values of the computation.

result: (required) *NXdata* <=

Information criterion as a function of number of n_ic_cluster aka dimensions.

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

axis_aic: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_ANY*}

Akaike information criterion values

axis_bic: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Bayes information criterion values

axis_dimension: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Actual n_ic_cluster values used

clustering: (optional) *NXprocess* <=

Step during which the chemically segmented voxel sets are analyzed for their spatial organization into different spatial clusters of voxels in the same chemical set but representing individual objects. The objects are constructed from blobs of neighboring voxels. The objects are not necessarily watertight or topologically closed.

sequence_index: (required) *NX_POSINT* <=

Any of these values: `4 | 5`

ic_opt: (required) *NXprocess*

Respective DBScan clustering result for each segmentation/ic_opt case.

cluster_analysisID: (optional) *NXprocess*

dbscanID: (required) *NXprocess*

epsilon: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The maximum distance between voxel pairs in a neighborhood to be considered connected.

min_samples: (required) *NX_UINT* {units=*NX_UNITLESS*}

The number of voxels in a neighborhood for a voxel to be considered as a core point.

label: (required) *NX_INT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

Raw label return values

voxel: (required) *NX_UINT* (Rank: 1, Dimensions: [n_voxels]) {units=*NX_UNITLESS*}

Voxel identifier

Using these identifiers correlated element-wise with the values in the label array specifies for which voxel in the grid clusters from this process were found.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_compositionspace_results/ENTRY-group*
- */NXapm_compositionspace_results/ENTRY/autophase-group*
- */NXapm_compositionspace_results/ENTRY/autophase/result-group*
- */NXapm_compositionspace_results/ENTRY/autophase/result/axis_feature_importance-field*
- */NXapm_compositionspace_results/ENTRY/autophase/result/axis_feature_importance@long_name-attribute*
- */NXapm_compositionspace_results/ENTRY/autophase/result/axis_feature_indices-field*
- */NXapm_compositionspace_results/ENTRY/autophase/result/axis_feature_indices@long_name-attribute*

- /NXapm_compositionspace_results/ENTRY/autophase/result/title-field
- /NXapm_compositionspace_results/ENTRY/autophase/result@axes-attribute
- /NXapm_compositionspace_results/ENTRY/autophase/result@AXISNAME_indices-attribute
- /NXapm_compositionspace_results/ENTRY/autophase/result@signal-attribute
- /NXapm_compositionspace_results/ENTRY/autophase/sequence_index-field
- /NXapm_compositionspace_results/ENTRY/clustering-group
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt-group
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt/cluster_analysisID-group
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt/cluster_analysisID/dbscanID-group
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt/cluster_analysisID/dbscanID/epsilon-field
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt/cluster_analysisID/dbscanID/label-field
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt/cluster_analysisID/dbscanID/min_samples-field
- /NXapm_compositionspace_results/ENTRY/clustering/ic_opt/cluster_analysisID/dbscanID/voxel-field
- /NXapm_compositionspace_results/ENTRY/clustering/sequence_index-field
- /NXapm_compositionspace_results/ENTRY/config-group
- /NXapm_compositionspace_results/ENTRY/config/algorithm-field
- /NXapm_compositionspace_results/ENTRY/config/checksum-field
- /NXapm_compositionspace_results/ENTRY/config/file_name-field
- /NXapm_compositionspace_results/ENTRY/definition-field
- /NXapm_compositionspace_results/ENTRY/definition@version-attribute
- /NXapm_compositionspace_results/ENTRY/environment-group
- /NXapm_compositionspace_results/ENTRY/environment/PROGRAM-group
- /NXapm_compositionspace_results/ENTRY/environment/PROGRAM/program-field
- /NXapm_compositionspace_results/ENTRY/environment/PROGRAM/program@version-attribute
- /NXapm_compositionspace_results/ENTRY/identifier_analysis-field
- /NXapm_compositionspace_results/ENTRY/profiling-group
- /NXapm_compositionspace_results/ENTRY/profiling/current_working_directory-field
- /NXapm_compositionspace_results/ENTRY/profiling/end_time-field
- /NXapm_compositionspace_results/ENTRY/profiling/start_time-field
- /NXapm_compositionspace_results/ENTRY/profiling/total_elapsed_time-field
- /NXapm_compositionspace_results/ENTRY/program1-group
- /NXapm_compositionspace_results/ENTRY/program1/program-field
- /NXapm_compositionspace_results/ENTRY/program1/program@version-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation-group
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt-group
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/cluster_analysisID-group

- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/cluster_analysisID/n_ic_cluster-field
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/cluster_analysisID/y_pred-field
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result-group
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result/axis_aic-field
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result/axis_bic-field
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result/axis_dimension-field
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result/title-field
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result@axes-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result@AXISNAME_indices-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/result@signal-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation/ic_opt/sequence_index-field
- /NXapm_compositionspace_results/ENTRY/segmentation/pca-group
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result-group
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result/axis_explained_variance-field
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result/axis_pca_dimension-field
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result/title-field
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result@axes-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result@AXISNAME_indices-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/result@signal-attribute
- /NXapm_compositionspace_results/ENTRY/segmentation/pca/sequence_index-field
- /NXapm_compositionspace_results/ENTRY/specimen-group
- /NXapm_compositionspace_results/ENTRY/specimen/atom_types-field
- /NXapm_compositionspace_results/ENTRY/specimen/is_simulation-field
- /NXapm_compositionspace_results/ENTRY/userID-group
- /NXapm_compositionspace_results/ENTRY/voxelization-group
- /NXapm_compositionspace_results/ENTRY/voxelization/grid-group
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/cardinality-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/cell_dimensions-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/coordinate-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/dimensionality-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/extent-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/index_offset-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/indices_voxel-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/origin-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/position-field
- /NXapm_compositionspace_results/ENTRY/voxelization/grid/symmetry-field

- */NXapm_compositionspace_results/ENTRY/voxelization/ionID-group*
- */NXapm_compositionspace_results/ENTRY/voxelization/ionID/name-field*
- */NXapm_compositionspace_results/ENTRY/voxelization/ionID/weight-field*
- */NXapm_compositionspace_results/ENTRY/voxelization/sequence_index-field*
- */NXapm_compositionspace_results/ENTRY/voxelization/weight-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_compositionspace_results.nxdl.xml

NXapm_paraprobe_clusterer_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-clusterer tool.

The tool paraprobe-clusterer evaluates how points cluster in space.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ivec_max: 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_paraprobe_tool_parameters, *NXentry*, *NXnote*, *NXprocess*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_clusterer_config`

cameca_to_nexus: (optional) *NXapm_paraprobe_tool_parameters* <=

This process maps results from a cluster analysis made with IVAS / AP Suite into an interoperable representation. IVAS / AP Suite usually exports such results as a list of reconstructed ion positions with one cluster label per position. These labels are reported via the mass-to-charge-state-ratio column of what is effectively a binary file that is formatted like a POS file but cluster labels written out using floating point numbers.

recover_evaporation_id: (required) *NX_BOOLEAN*

Specifies if paraprobe-clusterer should use the evaporation_ids from reconstruction for recovering for each position in the *NXnote* results the closest matching position (within floating point accuracy). This can be useful when users wish to recover the original evaporation_id, which IVAS /AP Suite drops when writing their *.indexed.* cluster results POS files that is referred to results.

reconstruction: (required) *NXnote* <=

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

position: (required) *NX_CHAR* <=

mass_to_charge: (required) *NX_CHAR* <=

results: (required) *NXnote* <=

File with the results of the cluster analyses that was computed with IVAS / AP suite (e.g. maximum-separation method clustering algorithm J. Hyde et al.). The information is stored in an improper (.indexed.) POS file as a matrix of floating point quadruplets, one quadruplet for each ion. The first three values of each quadruplet encode the position of the ion. The fourth value is the integer identifier of the cluster encoded as a floating point number.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

cluster_analysisID: (optional) *NXapm_paraprobe_tool_parameters* <=

This process performs a cluster analysis on a reconstructed dataset or a ROI within it.

ion_type_filter: (required) *NX_CHAR*

How should iontypes be considered during the cluster analysis.

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 nuclide structure.

The value resolve_element will set an ion active and account as many times for it, as the (molecular) ion contains atoms of elements in the whitelist ion_query_nuclide_vector.

The value resolve_isotope will set an ion active and account as many times for it, as the (molecular) ion contains nuclides in the whitelist ion_query_nuclide_vector.

In effect, ion_query_nuclide_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 an ion of a molecular ion with more than one nuclide, say Ti O for example is counted potentially several times because at that position (reconstructed) position it has been assumed that there was a Ti and an O atom. This multiplicity affects the size of the feature and its chemical composition.

Obligatory value: `resolve_element`

ion_query_nuclide_vector: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ions, n_ivec_max]) {units=*NX_UNITLESS*}

Matrix of nuclide 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.

surface_distance: (optional) *NXnote* <=

Distance between each ion and triangulated surface mesh.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR* <=

dbscan: (required) *NXprocess*

Settings for DBScan clustering algorithm. For original details about the algorithm 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 a set of cluster analyses with different parameter is executed:

- For tuple as many runs are performed as parameter values have been defined.
- For combinatorics individual parameter arrays are looped over.

As an example we may provide ten entries for eps and three entries for min_pts. If high_throughput_method is set to tuple, the analysis is invalid because we have an insufficient number of min_pts values to pair them with our ten eps values. By contrast, if high_throughput_method is set to combinatorics, the tool will run three individual min_pts runs for each eps value, resulting in a total of 30 analyses.

A typical example from the literature M. Kühbach et al. can be instructed via setting eps to an array of values np.linspace(0.2, 5.0, nums=241, endpoint=True), one min_pts value that is equal to 1, 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.

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 with different parameter values are composed, following the explanation for high_throughput_method of dbscan.

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_clusterer_config/ENTRY-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/reconstruction-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/reconstruction/algorithm-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/reconstruction/checksum-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/reconstruction/file_name-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/reconstruction/mass_to_charge-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/reconstruction/position-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/recover_evaporation_id-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/results-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/results/algorithm-field*

- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/results/checksum-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cameca_to_nexus/results/file_name-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/dbscan-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/dbscan/eps-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/dbscan/high_throughput_method-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/dbscan/min_pts-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/hdbscan-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/hdbscan/alpha-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/hdbscan/cluster_selection_epsilon-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/hdbscan/high_throughput_method-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/hdbscan/min_cluster_size-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/hdbscan/min_samples-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/ion_query_nuclide_vector-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/ion_type_filter-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/surface_distance-group*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/surface_distance/algorithm-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/surface_distance/checksum-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/surface_distance/distance-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/cluster_analysisID/surface_distance/file_name-field*
- */NXapm_paraprobe_clusterer_config/ENTRY/definition-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_clusterer_config.nxdl.xml

NXapm_paraprobe_clusterer_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-clusterer tool.

The tool paraprobe-clusterer evaluates how points cluster in space.

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_feat: Number of clusters found.

Groups cited:
NXapm_paraprobe_tool_process, *NXentry*, *NXprocess*, *NXsimilarity_grouping*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_clusterer_results`

cameca_to_nexus: (optional) *NXapm_paraprobe_tool_process* <=

cluster_analysisID: (optional) *NXapm_paraprobe_tool_process* <=

dbscanID: (optional) *NXsimilarity_grouping*

Results of a DBScan clustering analysis.

eps: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The epsilon (eps) parameter used.

min_pts: (required) *NX_UINT* {units=*NX_UNITLESS*}

The minimum points (min_pts) parameter used.

cardinality: (required) *NX_POSINT* {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, i.e. typically the number of clusters which is usually not and for sure not necessarily the total number of ions in the dataset.

index_offset: (required) *NX_INT* {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: * index_offset - 1 indicates an object belongs to no cluster. * index_offset - 2 indicates an object belongs to the noise category.

Setting for instance index_offset to 1 recovers the commonly used case that objects of the noise category get the value of -1 and points of the unassigned category get the value 0.

targets: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The evaporation (sequence) id (aka evaporation_id) to figure out which ions from the reconstruction were considered targets. The length of this array is not necessarily n_ions. Instead, it is the value of cardinality.

number_of_solutions: (optional) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

The number of solutions found for each target. Typically, this value is 1 in which case the field can be omitted. Otherwise, this array is the concatenated set of values of solution tuples for each target that can be used to decode model_labels, core_sample_indices, and weight.

model_label: (optional) *NX_INT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

The raw labels from the DBScan clustering backend process. The length of this array is not necessarily n_ions. Instead, it is typically the value of cardinality provided that each target has only one associated cluster.

If targets are assigned to multiple cluster this array is as long as the total number of solutions found and

core_sample_indices: (optional) *NX_INT* (Rank: 1, Dimensions: [k]) {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: [k]) {units=*NX_UNITLESS*}

Numerical label for each target (member in the set) aka cluster identifier.

categorical_label: (optional) *NX_CHAR* (Rank: 1, Dimensions: [k]) <=

Categorical label(s) for each target (member in the set) aka cluster name(s).

weight: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

Weights for each target that specifies how probable the target is assigned to a specific cluster.

For the DBScan algorithm and atom probe tomography this value is 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 nuclides of requested type.

is_noise: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [k])

Are targets assigned to the noise category or not.

is_core: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [k])

Are targets assumed a core point.

statistics: (recommended) *NXprocess* <=

In addition to the detailed storage which members were grouped to which feature here summary statistics are stored that communicate e.g. how many cluster were found.

number_of_targets: (required) *NX_UINT* {units=*NX_UNITLESS*}

Total number of targets in the set, i.e. ions that were filtered and considered in this cluster analysis.

number_of_noise_members: (required) *NX_UINT* {units=*NX_UNITLESS*}

Total number of members in the set which are categorized as noise.

number_of_core_members: (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).

indices_feature: (required) *NX_INT* (Rank: 1, Dimensions: [n_feat]) {units=*NX_UNITLESS*}

Numerical identifier of each feature aka cluster_id.

number_of_members: (required) *NX_UINT* (Rank: 1, Dimensions: [n_feat]) {units=*NX_UNITLESS*}

Number of members for each feature.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_clusterer_results/ENTRY-group*
- */NXapm_paraprobe_clusterer_results/ENTRY/cameca_to_nexus-group*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID-group*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID-group*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/cardinality-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/categorical_label-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/core_sample_indices-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/eps-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/index_offset-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/is_core-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/is_noise-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/min_pts-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/model_label-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/number_of_solutions-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/numerical_label-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics-group*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics/indices_feature-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics/number_of_core_members-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics/number_of_features-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics/number_of_members-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics/number_of_noise_members-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/statistics/number_of_targets-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/targets-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/cluster_analysisID/dbscanID/weight-field*
- */NXapm_paraprobe_clusterer_results/ENTRY/definition-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_clusterer_results.nxdl.xml

NXapm_paraprobe_distancer_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-distancer tool.

The tool paraprobe-distancer tool evaluates exactly the shortest Euclidean distance for each member of a set of points against a set of triangles.

Triangles can represent for instance the facets of a triangulated surface mesh like those returned by paraprobe-surfacer or any other set of triangles. Triangles do not have to be connected.

Currently, paraprobe-distancer does not check if the respectively specified triangle sets are consistent, what their topology is, or whether or not these triangles are consistently oriented.

Symbols:

No symbol table

Groups cited:
NXapm_paraprobe_tool_parameters, *NXentry*, *NXmatch_filter*, *NXnote*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_distancer_config`

point_to_triangleID: (required) *NXapm_paraprobe_tool_parameters* <=

method: (required) *NX_CHAR*

Specifies for which point the tool will compute distances.

The value *default* configures that distances are computed for all points. The value *skin* configures that distances are computed only for those points which are not farther away located to a triangle than threshold_distance.

Any of these values: `default` | `skin`

threshold_distance: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Maximum distance for which distances are computed when *method* is *skin*.

number_of_triangle_sets: (required) *NX_UINT* {units=*NX_UNITLESS*}

How many triangle sets to consider. Multiple triangle sets can be defined which are composed into one joint triangle set for the analysis.

triangle_setID: (required) *NXnote* <=

Each triangle_set that is referred to here should be a face_list_data_structure, i.e. an array of (n_vertices, 3) of NX_FLOAT for vertex coordinates, an (n_facets, 3) array of NX_UINT incident vertices of each facet. Vertex indices are assumed to start at zero and must not exceed n_vertices - 1, i.e. the index_offset is 0. Facet normal have to be provided as an array of (n_facets, 3) of NX_FLOAT.

algorithm: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

file_name: (required) *NX_CHAR* <=

vertices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex positions for the triangles in that triangle_set.

indices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex indices for the triangles in that triangle_set.

vertex_normals: (optional) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex normal vectors for the triangles in that triangle_set.

face_normals: (optional) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of facet normal vectors for the triangles in that triangle_set.

indices_patch: (optional) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of identifier for the triangles in that triangle_set.

patch_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_distancer_config/ENTRY-group*
- */NXapm_paraprobe_distancer_config/ENTRY/definition-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID-group*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/method-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/number_of_triangle_sets-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/threshold_distance-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID-group*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/algorithm-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/checksum-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/face_normals-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/file_name-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/indices-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/indices_patch-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/patch_filter-group*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/patch_filter/match-field*

- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/patch_filter/method-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/vertex_normals-field*
- */NXapm_paraprobe_distancer_config/ENTRY/point_to_triangleID/triangle_setID/vertices-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_distancer_config.nxdl.xml

NXapm_paraprobe_distancer_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-distancer tool.

The tool paraprobe-distancer tool evaluates exactly the shortest Euclidean distance for each member of a set of points against a set of triangles.

Triangles can represent for instance the facets of a triangulated surface mesh like those returned by paraprobe-surfacer or any other set of triangles. Triangles do not have to be connected.

Currently, paraprobe-distancer does not check if the respectively specified triangle sets are consistent, what their topology is, or whether or not these triangles are consistently oriented.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of points, i.e. ions in the reconstruction.

n_tri: The total number of triangles in the set.

Groups cited:
NXapm_paraprobe_tool_process, *NXcs_filter_boolean_mask*, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_distancer_results`

point_to_triangleID: (required) *NXapm_paraprobe_tool_process* <=

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_LENGTH*}

The shortest analytical distance of each point to their respectively closest triangle from the joint triangle set.

indices_triangle: (optional) *NX_INT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

For each point the identifier of the triangle for which the shortest distance was found.

indices_point: (optional) *NX_INT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

A support field to enable the visualization of each point by an explicit identifier on the interval [0, n_ions - 1]. The field can be used to visualize the points as a function of their distance to the triangle set (e.g. via XDMF/Paraview).

sign_valid: (optional) *NXcs_filter_boolean_mask*

A bitmask that identifies which of the distance values is assumed to have a consistent sign because the closest triangle had a consistent outer unit normal defined.

For points whose bit is set to 0 the distance is correct but the sign is not reliable.

number_of_triangles: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of triangles covered by the mask.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Bitdepth of the elementary datatype that is used to store the information content of the mask (typically 8 bit, uint8).

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The content of the mask. Like for all masks used in the tools of the paraprobe-toolbox, padding is used when number_of_objects is not an integer multiple of bitdepth. If padding is used, padded bits are set to 0.

window_triangles: (optional) *NXcs_filter_boolean_mask*

A bitmask that identifies which of the triangles in the set were considered when certain triangles have been filtered out.

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_tri]) <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_distancer_results/ENTRY-group*
- */NXapm_paraprobe_distancer_results/ENTRY/definition-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID-group*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/distance-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/indices_point-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/indices_triangle-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/sign_valid-group*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/sign_valid/bitdepth-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/sign_valid/mask-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/sign_valid/number_of_triangles-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/window_triangles-group*

- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/window_triangles/bitdepth-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/window_triangles/mask-field*
- */NXapm_paraprobe_distancer_results/ENTRY/point_to_triangleID/window_triangles/number_of_objects-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_distancer_results.nxdl.xml

NXapm_paraprobe_intersector_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-intersector tool.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_variable: Number of entries

Groups cited:
NXapm_paraprobe_tool_parameters, *NXentry*, *NXnote*, *NXparameters*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_intersector_config`

v_v_spatial_correlationID: (required) *NXapm_paraprobe_tool_parameters* <=

Tracking volume_volume_spatial_correlations (v_v) is the process of building logical relations between objects, their proximity and eventual volumetric intersections. Here, objects are assumed to be represented as a set of triangulated surface meshes.

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.

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., it is assumed by default 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 tetrahedralization library used in the tests (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. For practical reasons we have thus deactivated the functionality of tetrahedra-tetrahedron intersections in paraprobe-intersector.

Obligatory value: `shared_ion`

analyze_intersection: (required) *NX_BOOLEAN*

Specifies if the tool evaluates if objects intersect volumetrically.

analyze_proximity: (required) *NX_BOOLEAN*

Specifies if the tool evaluates if objects lay closer to one another than threshold_proximity.

analyze_coprecipitation: (required) *NX_BOOLEAN*

Specifies if the tool evaluates, provided that all (preprocessing tasks were successful), how intersecting or proximity related objects build sub-graphs. This is the feature that was used in M. Kühbach et al. for the high-throughput analyses of how many objects are coprecipitates in the sense that they are single, duplet, triplet, or high-order local groups.

threshold_proximity: (required) *NX_FLOAT* {units=*NX_LENGTH*}

The maximum Euclidean distance between two objects below which they are 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) *NXparameters* <=

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 can be interpreted as the time/iteration (i.e. k) step when the current set was taken (see M. Kühbach et al. 2022).

number_of_feature_types: (required) *NX_UINT* {units=*NX_UNITLESS*}

The total number of distinguished feature sets featureID. It is assumed that the members within all these featureID 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.

objectID: (required) *NXnote* <=

Name of the (NeXus)/HDF5 file which contains triangulated surface meshes of the members of the set as instances of NXcg_polyhedron.

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`
- `other`

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

geometry: (required) *NX_CHAR*

Absolute path to the group with geometry data in the HDF5 file referred to by path.

indices_feature: (required) *NX_INT* (Rank: 1, Dimensions: [n_variable]) {units=*NX_UNITLESS*}

Array of identifier whereby the path to the geometry data can be inferred automatically.

next_set: (required) *NXparameters* <=

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 current set. The label effectively can be interpreted as the time/iteration (i.e. $k+1$) step when the current set was taken (see M. Kühbach et al. 2022).

number_of_feature_types: (required) *NX_UINT* {units=*NX_UNITLESS*}

The total number of distinguished feature sets featureID. It is assumed that the members within all these featureID 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.

objectID: (required) *NXnote* <=

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`

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

geometry: (required) *NX_CHAR*

Absolute path to the group with geometry data in the HDF5 file referred to by path.

indices_feature: (required) *NX_INT* (Rank: 1, Dimensions: [n_variable]) {units=*NX_UNITLESS*}

Array of identifier whereby the path to the geometry data can be inferred automatically.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_intersector_config/ENTRY-group*
- */NXapm_paraprobe_intersector_config/ENTRY/definition-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID-group*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/analyze_coprecipitation-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/analyze_intersection-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/analyze_proximity-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set-group*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/number_of_feature_types-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID-group*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID/algorithm-field*

- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID/checksum-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID/feature_type-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID/file_name-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID/geometry-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/objectID/indices_feature-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/current_set/set_identifier-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/has_current_to_next_links-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/has_next_to_current_links-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/intersection_detection_method-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set-group*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/number_of_feature_types-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID-group*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID/algorithm-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID/checksum-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID/feature_type-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID/file_name-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID/geometry-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/objectID/indices_feature-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/next_set/set_identifier-field*
- */NXapm_paraprobe_intersector_config/ENTRY/v_v_spatial_correlationID/threshold_proximity-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_intersector_config.nxdl.xml

NXapm_paraprobe_intersector_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for results files of the paraprobe-intersector tool.

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 statistics.

Groups cited:
NXapm_paraprobe_tool_process, *NXentry*, *NXprocess*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_intersector_results`

v_v_spatial_correlationID: (optional) *NXapm_paraprobe_tool_process* <=

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 indices_feature that 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 volumetric overlap (0x00 == 0), proximity (0x01 == 1), or something else unknown (0xFF == 255).

next_to_current_link: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_n2c, 2]) {units=*NX_UNITLESS*}

A matrix of indices_feature 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 are defined is symmetric it holds that next_to_current maps the links for current_to_next in just 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: [n_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, pair relations are analyzed with respect to which objects with indices_feature cluster in identifier space. Thereby, a logical connection (link) is established between the features in the current_set and the next_set. Recall that these two sets 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 for precipitates B.

Coprecipitation analysis now logically connects these independent characterization results to establish spatial correlations of e.g. the precipitates' spatial arrangement.

current_set_feature_to_cluster: (required) *NX_UINT* (Rank: 2, Dimensions: [n_features_curr, 2]) {units=*NX_UNITLESS*}

Matrix of indices_feature and cluster_id pairs which encodes the cluster to which each indices_feature 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 indices_feature and cluster_id pairs which encodes the cluster to which each indices_feature was assigned. Here for features of the next_set.

cluster_id: (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 third 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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_intersector_results/ENTRY-group*
- */NXapm_paraprobe_intersector_results/ENTRY/definition-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID-group*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/coprecipitation_analysis-group*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/coprecipitation_analysis/cluster_composition-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/coprecipitation_analysis/cluster_id-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/coprecipitation_analysis/cluster_statistics-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/coprecipitation_analysis/current_set_feature_to_clus field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/coprecipitation_analysis/next_set_feature_to_cluster-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/current_to_next_link-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/current_to_next_link_type-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/intersection_volume-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/next_to_current_link-field*
- */NXapm_paraprobe_intersector_results/ENTRY/v_v_spatial_correlationID/next_to_current_link_type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_intersector_results.nxdl.xml

NXapm_paraprobe_nanochem_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-nanochem tool.

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_grid: How many grid_resolutions values.

n_var: How many kernel_variance values.

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_max: Maximum number of atoms per molecular ion.

n_rois: Number of cylinder ROIs to place for oned_profile if no feature mesh is used.

Groups cited:
NXapm_paraprobe_tool_parameters, *NXcg_cylinder*, *NXentry*, *NXmatch_filter*, *NXnote*, *NXprocess*, *NXroi_process*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_nanochem_config`

delocalizationID: (optional) *NXapm_paraprobe_tool_parameters* <=

Discretization and distributing of the ion point cloud on a 3D grid to enable analyses at the continuum scale.

By default, the tool computes a full kernel density estimation of decomposed ions to create one discretized field for each element.

One delocalization task configures a parameter sweep with at least one delocalization. The total number of runs depends on the number of grid_resolution and kernel_variance values. For example, setting two grid_resolutions and three kernel_variance will compute six runs. Two sets of three with the first set using the first grid_resolutions and in sequence the kernel_variance respectively.

method: (required) *NX_CHAR*

Compute delocalization or load an existent one from input.

Any of these values: `compute | load_existent`

nuclide_whitelist: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ityp_deloc_cand, n_ivec_max]) {units=*NX_UNITLESS*}

Matrix of nuclides representing how iontypes should be accounted for during the delocalization. This is the most general approach to define if and how many times an ion is to be counted. The tool performs a so-called atomic decomposition of all iontypes, i.e. the tool analyses from how many atoms of each nuclide or element respectively an (molecular) ion is built from.

Taking the hydroxonium H3O+ molecular ion as an example: It contains hydrogen and oxygen atoms. The multiplicity of hydrogen is three whereas that of oxygen is one. Therefore, the respective atomic decomposition analysis prior to the iso-surface computation adds three hydrogen counts for each H3O+ ion.

This is a practical solution which accepts that on the one hand not every bond is broken during an atom probe experiment but also that ions may react further during their flight to the detector. The exact details depend on the local field conditions, quantum mechanics of possible electron transfer and thus the detailed trajectory of the system and its electronic state.

The detection of molecular ions instead of always single atom ions only is the reason that an atom probe experiment tells much about field evaporation physics but also faces an inherent loss of information with respect to the detailed spatial arrangement that is independent of other imprecisions such as effect of limited accuracy of reconstruction protocols and their parameterization.

Unused values in each row of the matrix are nullified. Nuclides are identified as hashed nuclide (see *NXatom*) for further details.

grid_resolution: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_grid]) {units=*NX_LENGTH*}

Array of edge lengths of the cubic cells used for discretizing the reconstructed dataset on a cuboidal 3D grid (*NXcg_grid*). The tool performs as many delocalization computations as values are specified in grid_resolution.

kernel_size: (required) *NX_UINT* {units=*NX_UNITLESS*}

Half the width of a $(2 \cdot n + 1)^3$ cubic kernel of cubic voxel beyond which the Gaussian Ansatz function will be truncated. Intensity outside the kernel is factorized into the kernel via a normalization procedure.

kernel_variance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_var]) {units=*NX_LENGTH*}

Array of variance values σ of the Gaussian Ansatz kernel ($\sigma_x := \sigma$, $\sigma_x = \sigma_y = 2 \cdot \sigma_z$). The tool performs as many delocalization computations as values are specified in kernel_variance.

normalization: (required) *NX_CHAR*

How should the results of the kernel-density estimation be normalized 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 nuclide_whitelist.

Any of these values: `none` | `composition` | `concentration`

has_scalar_fields: (required) *NX_BOOLEAN*

Specifies if the tool should report the delocalization 3D field values.

surface: (required) *NXnote* <=

A precomputed 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.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

vertices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex positions for the triangles in that triangle_set.

indices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex indices for the triangles in that triangle_set.

surface_distance: (recommended) *NXnote* <=

Distance between each ion and triangulated surface mesh.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR* <=

decomposition: (required) *NXmatch_filter*

Configuration for the algorithm that defines the multiplicity of each reconstructed position during the delocalization.

method: (required) *NX_CHAR* <=

The multiplicity of an ion at a reconstructed position is defined as follows:

- resolve_unknown, multiplicity equals 1 for all ions of the unknown_type This mode is useful for segmenting regions with poor ranging.
- resolve_point, multiplicity equals 1 for all ions This mode is useful for segmenting point density.
- resolve_atom, multiplicity equals the number of atoms per ion This mode is useful for segmenting atomic density.
- resolve_element, multiplicity equals the number of elements in the whitelist per ion This mode is useful for segmenting regions of specific elemental composition (ignoring nuclids)
- resolve_element_charge, ???multiplicity like resolve_element when charge is met
- resolve_isotope, multiplicity equals the number of nuclides in the whitelist per ion This mode is useful for segmenting regions of specific isotopic composition
- resolve_isotope_charge, ???

Other multiplicities are 0.

Any of these values:

- `resolve_unknown`
- `resolve_point`
- `resolve_atom`
- `resolve_element`
- `resolve_element_charge`
- `resolve_isotope`
- `resolve_isotope_charge`

nuclide_whitelist: (required) *NX_UINT* {units=*NX_UNITLESS*}

TODO

charge_state_whitelist: (required) *NX_INT* {units=*NX_DIMENSIONLESS*}

TODO

input: (required) *NXnote* <=

Serialized result of an already computed delocalization which is for performance reasons here just loaded and not computed again.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

results: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the group within which individual delocalization results are stored.

isosurfacing: (optional) *NXprocess*

Configuration of the set of iso-surfaces to compute using that delocalization. Such iso-surfaces are the starting point for a reconstruction of so-called objects or (microstructural) features. Examples of scientific relevant are (line features e.g. dislocations poles, surface features such as interfaces, i.e. phase and grain boundaries, or volumetric features such as precipitates. Users should be aware that reconstructed datasets in atom probe are a model and may face inaccuracies and artifacts that can be mistaken incorrectly as microstructural features.

edge_method: (required) *NX_CHAR*

As it is detailed in M. Kühbach et al., the handling of triangles at the surface (edge) of the dataset requires special attention especially for composition-normalized delocalization. Here, it is possible that the composition increases towards the edge of the dataset because the quotient of two numbers that 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, keep_edge_triangles instructs the tool to not remove triangles at the edge of the dataset at the cost of bias. When using this keep_edge_triangles users should understand that all features in contact with the edge of the dataset get usually artificial enlarged. Consequently, triangulated surface meshes of these objects are closed during the marching. However, this closure is artificial and can bias shape analyses for those objects. This also holds for such practices that are offered in proprietary software like IVAS / AP Suite. The situation is comparable to analyses of grain shapes via orientation microscopy from electron microscopy or X-ray diffraction tomography. Features at the edge of the dataset may have not been captured fully.

Thanks to collaboration with V. V. Rielli and S. Primig from the Sydney atom probe group, paraprobe-nanochem implements a complete pipeline to process features at the edge of the dataset. Specifically, these

are modelled and replaced with closed polyhedral objects using an iterative mesh and hole-filling procedures with fairing operations.

The tool bookkeeps such objects separately to lead the decision whether or not to consider these objects to the user. 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. Furthermore, users should be careful with overestimating the statistical significance of their dataset especially when using atom probe when they use their atom probe result to compare different descriptors. Even though a dataset may deliver statistically significant results for compositions, this does not necessarily mean that same dataset will also yield statistically significant and insignificantly biased results for 3D 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-surface distance that is used in the analyses of features to identify whether these are laying inside the dataset or close to the surface (edge) of the dataset.

If an object has at least one ion with an ion-to-surface-distance below this threshold, the object is considered close to the edge of the dataset. The tool uses a distance-based approach to solve the in general complicated and involved treatment of computing volumetric intersections between closed 2-manifolds that are not necessarily convex. The main practical reason is that such computational geometry analyses face numerical robustness issues as a consequence of which a mesh can be detected as being completely inside another mesh although in reality it is only ϵ-close only, i.e. almost touching only the edge (e.g. from inside).

Practically, humans would likely still 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 flexible.

phi: (required) *NX_FLOAT* {units=*NX_ANY*}

Iso-contour values. For each value, the tool computes an iso-surface and performs subsequent analyses for each iso-surface. 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/nm^3

has_triangle_soup: (required) *NX_BOOLEAN*

Specifies if the tool should report the triangle soup which represents each triangle of the iso-surface complex. The resulting set of triangles is colloquially referred to as a soup because different sub-set may not be connected.

Each triangle is reported with an ID specifying to which triangle cluster (with IDs starting at zero) the triangle belongs. The clustering of triangles within the soup 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 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 scandium-rich precipitates in aluminum alloys which in reconstructions may appear as ellipsoids although they should be indeed almost spherical provided their size is larger than the atomic length scale.

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 tiny sub-complexes (e.g. small isolated tetrahedra).

These can be small tetrahedra/polyhedra about the center of a voxel of the support grid on which marching cubes operates. Such objects may not contain an 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 because 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 a volume that contains no 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).

Users should be aware that the choice of the algorithm to compute the bounding box can have an effect on aspect ratio statistics. It is known

that computing the true optimal bounding box of in 3D is an $\mathcal{O}^3$-time-complex task. The tool uses well-established approximate algorithms of the Computational Geometry Algorithms Library (CGAL).

has_object_ions: (required) *NX_BOOLEAN*

Specifies if the tool should report for each closed polyhedron all evaporation IDs of those ions which lay inside or on the boundary of the polyhedron. This information is used by the paraprobe-intersector tool to infer if two objects share common ions, which is then understood as 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 and flat face(t)s. 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 reconstructed ion positions. Consequently, sharing ions is a sufficient but not a necessary condition for interpreting (volumetric) intersections between objects.

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 surface aka edge of the dataset. For this the tool currently inspects if the shortest distance between the set of triangles of the triangulated surface mesh and the triangles of the edge model is larger than 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 shape is likely affected by the edge.

has_proxy: (required) *NX_BOOLEAN*

Specifies if the tool should analyze a closed polyhedron (aka proxy) for each cluster of triangles whose combinatorial analysis according to has_object returned 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 feature. In most cases objects, precipitates in atom probe end up as open objects because they have been clipped by the edge of the dataset. Using a proxy is in this case a strategy to still be able to account for these objects. However, users should make themselves familiar with the consequences and potential bias 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_proxigram: (required) *NX_BOOLEAN*

Specifies if the tool should report for each closed object a (cylindrical) region-of-interest (ROI) that gets placed, centered, and aligned with the local normal for each triangle of the object.

has_object_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 with the edge the dataset. Currently, the tool supports cylindrical ROIs. A computational geometry test is performed to check for a possible intersection of each ROI with the triangulated surface mesh that is defined via surface. Results of this cylinder-set-of-triangles intersection are interpreted as follows: If the cylinder intersects with at least one triangle of the surface (mesh) the ROI is assumed to make edge contact. Otherwise, the ROI is assumed to make no edge contact.

Users should note that this approach does not work if the ROI is laying completely outside the dataset as also in this case the cylinder intersects with any triangle. However, for atom probe this case is practically irrelevant provided constructions such as alpha shapes or alpha wrappings (such as paraprobe-surfacer does) about the ions of the entire reconstructed volume are used.

interface_meshingID: (optional) *NXapm_paraprobe_tool_parameters* <=

Use a principle component analysis (PCA) to mesh a single free-standing interface patch within the reconstructed volume that is decorated by ions of specific iontypes (e.g. solute atoms).

Interface_meshing is a typical starting point for the quantification of Gibbsian interfacial excess in cases when closed objects constructed from patches e.g. iso-surfaces are not available or when there is no substantial or consistently oriented concentration gradients across an interface patch. The functionality can also be useful when the amount of latent crystallographic information within the point cloud is insufficient or when combined with interface_meshing based on ion density traces in field-desorption maps (see Y. Wei et al. and A. Breen et al. for details).

Noteworthy to mention is that the method used is conceptually similar to the work of Z. Peng et al. and related work (DCOM algorithm) by P. Felfer et al.. Compared to these implementations paraprobe-nanochem uses inspection functionalities which detect potential geometric inconsistencies or self-interactions of the evolved DCOM mesh.

initialization: (required) *NX_CHAR*

How is the PCA initialized:

- default, means based on segregated solutes in the ROI
- control_point_file, means based on reading an external list of control points, currently coming from the Leoben APT_Analyzer.

The control_point_file is currently expected with a specific format. The Leoben group lead by L. Romaner has developed a GUI tool A. Reichmann et al. creates a control_point_file that can be parsed by paraprobe-parmsetup-nanochem to match the here required formatting in control_points.

Any of these values: `default | 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.

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 might be relocated.

The larger it is the DCOM radius in relation to the target_edge_length the more likely it becomes that vertices will be relocated so substantially that triangle self-intersections may occur. The tool detects these and stops in a controlled manner so that the user can repeat the analyses with using a different parameterization.

target_smoothing_step: (required) *NX_UINT* (Rank: 1, Dimensions: [n_fct_iterations]) {units=*NX_UNITLESS*}

Array of integers which specify for each DCOM step how many times the mesh should be iteratively smoothened. Users should be aware that all three arrays 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 respective parameter 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.

surface: (optional) *NXnote* <=

A precomputed 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.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

vertices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex positions for the triangles in that triangle_set.

indices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex indices for the triangles in that triangle_set.

control_point: (required) *NXnote* <=

Details about the control point file used.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

control_points: (required) *NX_CHAR*

X, Y, Z position matrix of disjoint control points.

decoration_filter: (required) *NXmatch_filter*

Specify those nuclides which the tool should inspect iontypes for if they contain such nuclides. If this is the case ions of such type are taken with the number of nuclides of this multiplicity found. The atoms of these ions are assumed to serve as useful markers for locating the interface and refining the interface mesh.

method: (required) *NX_CHAR* <=

Obligatory value: `whitelist`

match: (required) *NX_UINT* (Rank: 2, Dimensions: [n_fct_filter_cand, n_ivec_max]) {units=*NX_UNITLESS*}

Array of nuclide iontypes to filter.

oned_profileID: (optional) *NXapm_paraprobe_tool_parameters* <=

Analysis of one-dimensional profiles in ROIs placed in the dataset. Such analyses are useful for quantifying interfacial excess or for performing classical composition analyses.

The tool will test for each ROIs if it is completely embedded in the dataset. Specifically, each such test evaluates if the ROI cuts at least one triangle of the triangulated surface mesh that is referred to by surface. If this is the case the ROI is marked as one close to the surface and not analyzed further. Otherwise, the ROI is marked as one far from the surface and processed further.

For each ROI the tool computes atomically decomposed profiles. This means, molecular ions are split into nuclides as many times as their respective multiplicity. For each processed ROI the tool stores a sorted list of signed distance values to enable post-processing with other software like e.g. reporter to perform classical Krakauer/Seidman-style interfacial excess analyses.

Users should be aware that the latter intersection analysis is not a volumetric intersection analysis. Given that the triangulated mesh referred to in surface is not required to mesh neither a watertight nor convex polyhedron a rigorous testing of volumetric intersection is much more involved. If the mesh is watertight one could use split the task in first tessellating the mesh into convex polyhedra (e.g. tetrahedra and apply a volumetric intersection method like the Gilbert-Johnson-Keerthi algorithm (GJK). In cases when the mesh is not even watertight distance-based segmentation in combination with again intersection of triangles and convex polyhedra is a robust but currently not implemented method to quantify intersections.

distancing_model: (required) *NX_CHAR*

Which type of distance should be reported for the profile.

Obligatory value: `project_to_triangle_plane`

roi_orientation: (required) *NX_CHAR*

For each ROI, along which direction should the cylindrical ROI be oriented if ROIs are placed at triangles of the feature mesh.

Obligatory value: `triangle_outer_unit_normal`

roi_cylinder_height: (required) *NX_FLOAT* {units=*NX_LENGTH*}

For each ROI, how high (projected onto the cylinder axis) should the cylindrical ROI be if ROIs are placed at triangles of the feature mesh.

roi_cylinder_radius: (required) *NX_FLOAT* {units=*NX_LENGTH*}

For each ROI, how wide (in radius) should the cylindrical ROI be if ROIs are placed at triangles of the feature mesh.

surface: (optional) *NXnote* <=

A precomputed 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.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

vertices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex positions for the triangles in that triangle_set.

indices: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the array of vertex indices for the triangles in that triangle_set.

surface_distance: (recommended) *NXnote* <=

Distance between each ion and triangulated surface mesh.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR* <=

Absolute path in the (HDF5) file that points to the distance values. The tool assumes that the values are stored in the same order as points (ions).

feature: (optional) *NXnote* <=

A precomputed triangulated mesh of the feature representing a model of the interface at which to place ROIs to profile. This can be the mesh of an interface as returned e.g. by a previous interface_meshing task or the mesh of an iso-surface from a previous delocalization task.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

vertices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset that specifies the array of vertex positions.

indices: (required) *NX_CHAR*

Absolute HDF5 path to the dataset that specifies the array of facet indices which refer to vertices.

facet_normals: (required) *NX_CHAR*

Absolute HDF5 path to the dataset that specifies the array of facet signed unit normals.

vertex_normals: (required) *NX_CHAR*

Absolute HDF5 path to the dataset that specifies the array of vertex signed unit normals.

patch_filter: (optional) *NXmatch_filter*

If interface_model is isosurface this filter can be used to restrict the analysis to specific patches of an iso-surface.

method: (required) *NX_CHAR* <=

match: (required) *NX_NUMBER* <=

feature_distance: (optional) *NXnote* <=

To enable an additional filtration of specific parts of the feature mesh it is recommended to feed precomputed distances of each ion to the triangles of the feature mesh.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR*

Absolute path in the (HDF5) file that points to the distance values. The tool assumes that the values are stored in the same order as points (ions).

user_defined_roi: (optional) *NXroi_process*

As an alternative mode the tool can be instructed to place ROIs at specific locations into the dataset. This is the programmatic equivalent to the classical

approach in atom probe to place ROIs for composition analyses via positioning and rotating them via a graphical user interface (such as in IVAS / AP Suite). **cylinder_set**: (required) *NXcg_cylinder*

index_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_rois, 3]) <=

height: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_rois, 3]) <=

radii: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_rois]) <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_nanochem_config/ENTRY-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/definition-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/decomposition-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/decomposition/charge_state_whitelist-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/decomposition/method-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/decomposition/nuclide_whitelist-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/grid_resolution-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/has_scalar_fields-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/input-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/input/algorithm-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/input/checksum-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/input/file_name-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/input/results-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/edge_method-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/edge_threshold-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_edge_contact-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_geometry-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_ions-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_obb-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_properties-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_proxigram-field*

- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_object_proxigram_edge_contact-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_proxy-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_proxy_edge_contact-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_proxy_geometry-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_proxy_ions-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_proxy_obb-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_proxy_properties-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/has_triangle_soup-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/isosurfacing/phi-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/kernel_size-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/kernel_variance-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/method-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/normalization-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/nuclide_whitelist-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface-group
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface/algorithm-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface/checksum-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface/file_name-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface/indices-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface/vertices-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface_distance-group
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface_distance/algorithm-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface_distance/checksum-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface_distance/distance-field
- /NXapm_paraprobe_nanochem_config/ENTRY/delocalizationID/surface_distance/file_name-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID-group
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/control_point-group
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/control_point/algorithm-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/control_point/checksum-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/control_point/control_points-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/control_point/file_name-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/decoration_filter-group
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/decoration_filter/match-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/decoration_filter/method-field
- /NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/initialization-field

- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/method-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/number_of_iterations-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/surface-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/surface/algorithm-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/surface/checksum-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/surface/file_name-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/surface/indices-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/surface/vertices-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/target_dcom_radius-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/target_edge_length-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/interface_meshingID/target_smoothing_step-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/distancing_model-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/algorithm-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/checksum-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/facet_normals-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/file_name-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/indices-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/patch_filter-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/patch_filter/match-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/patch_filter/method-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/vertex_normals-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature/vertices-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature_distance-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature_distance/algorithm-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature_distance/checksum-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature_distance/distance-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/feature_distance/file_name-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/roi_cylinder_height-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/roi_cylinder_radius-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/roi_orientation-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface/algorithm-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface/checksum-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface/file_name-field*

- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface/indices-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface/vertices-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface_distance-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface_distance/algorithm-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface_distance/checksum-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface_distance/distance-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/surface_distance/file_name-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/user_defined_roi-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/user_defined_roi/cylinder_set-group*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/user_defined_roi/cylinder_set/center-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/user_defined_roi/cylinder_set/height-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/user_defined_roi/cylinder_set/index_offset-field*
- */NXapm_paraprobe_nanochem_config/ENTRY/oned_profileID/user_defined_roi/cylinder_set/radii-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_nanochem_config.nxdl.xml

NXapm_paraprobe_nanochem_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-nanochem tool.

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: The total number of faces of triangles.

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_v_feat: The total number of volumetric features.

n_speci: The total number of member distinguished when reporting composition.

n_rois: The total number of ROIs placed in a oned_profile task.

Groups cited:

NXapm_paraprobe_tool_process, *NXatom*, *NXcg_face_list_data_structure*, *NXcg_grid*, *NXcg_hexahedron*, *NXcg_polyhedron*, *NXcg_roi*, *NXcg_triangle*, *NXcg_unit_normal*, *NXchemical_composition*, *NXcs_filter_boolean_mask*, *NXdata*, *NXdelocalization*, *NXentry*, *NXisocontour*, *NXprocess*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_nanochem_results`

delocalizationID: (optional) *NXdelocalization*

window: (required) *NXcs_filter_boolean_mask*

number_of_ions: (required) *NX_UINT*

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

grid: (required) *NXcg_grid* <=

The discretized domain/grid on which the delocalization is applied.

dimensionality: (required) *NX_POSINT* <=

Any of these values: `1 | 2 | 3`

cardinality: (required) *NX_POSINT* <=

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])

Number of unit cells along each of the d-dimensional base 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.

index_offset: (required) *NX_INT* {units=*NX_UNITLESS*} <=

Integer which specifies the first index to be used for distinguishing identifiers for cells. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval $[identifier_offset, identifier_offset + c - 1]$. For explicit indexing the identifier array has to be defined.

kernel_size: (required) *NX_POSINT* (Rank: 1, Dimensions: [3]) {units=*NX_DIMENSIONLESS*}

Halfwidth of the kernel about the central voxel. The shape of the kernel is that of a cuboid of extent 2*kernel_extent[i] + 1 in each dimension i.

kernel_type: (required) *NX_CHAR*

Functional form of the kernel (Ansatz function).

Obligatory value: `gaussian`

kernel_sigma: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Standard deviation σ_i 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 μ_i of the kernel in each dimension in the paraprobe coordinate_system with i = 0 is x, i = 1 is y, i = 2 is z.

normalization: (required) *NX_CHAR*

How were results of the kernel-density estimation normalized:

- total, the total number (intensity) of ions or elements.
- candidates, the total number (intensity) of ions matching weighting_model
- composition, the value for candidates divided by the value for total,
- concentration, the value for candidates divided by the volume of the cell.

Any of these values:

- `total`
- `candidates`
- `composition`
- `concentration`

bounding_box: (required) *NXcg_hexahedron*

A tight axis-aligned bounding box about the grid.

is_axis_aligned: (required) *NX_BOOLEAN* <=

For atom probe should be set to true.

index_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.

hexahedron: (required) *NXcg_face_list_data_structure* <=

vertex_index_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_index_offset: (required) *NX_INT* {units=*NX_UNITLESS*}

Integer which specifies the first index to be used for distinguishing identifiers for faces. Identifiers are defined either implicitly or explicitly. For implicit indexing the identifiers are defined on the interval $[identifier_offset, identifier_offset + c - 1]$. For explicit indexing the identifier array has to be defined.

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [8, 3]) {units=*NX_LENGTH*} <=

Positions of the vertices. Users are encouraged to reduce the vertices to unique set of positions and vertices as this supports a more efficient storage of the geometry data. It is also possible though to store the vertex positions naively in which case vertices_are_unique is likely False. Naively here means that one for example stores each vertex of a triangle mesh even though many vertices are shared between triangles and thus the positions of these vertices do not have to be duplicated.

faces: (required) *NX_NUMBER* (Rank: 2, Dimensions: [6, 4]) {units=*NX_UNITLESS*}

Array of identifiers from vertices which describe each face.

The first entry is the identifier of the start vertex of the first face, followed by the second vertex of the first face, until the last vertex of the first face. Thereafter, the start vertex of the second face, the second vertex of the second face, and so on and so forth.

Therefore, summating over the number_of_vertices, allows to extract the vertex identifiers for the i-th face on the following index interval of the faces array: $[\sum_{i=0}^{i=n-1}, \sum_{i=0}^{i=n}]$.

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [36]) {units=*NX_UNITLESS*}

Six equally formatted sextets chained together. For each sextett the first entry is an XDMF primitive topology key (here 5 for polygon), the second entry the XDMF primitive count value (here 4 because each face is a quad). The remaining four values are the vertex indices.

number_of_boundaries: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

How many distinct boundaries are distinguished? Most grids discretize a cubic or cuboidal region. In this case six sides can be distinguished, each making an own boundary.

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

scalar_field_magn_SUFFIX: (optional) *NXdata* <=

The result of the delocalization $\Phi = f(x, y, z)$ based on which subsequent iso-surfaces will be computed. In commercial software so far there is no possibility to export this information.

If the intensity for all matches of the weighting_model are summarized, name this NXdata instance scalar_field_magn_total.

If the intensity is reported for each iontype, one can avoid many subsequent computations as individual intensities can be reinterpreted using a different weighting_model in down-stream usage of the here reported values (e.g. with Python scripting). In this case name the individual NXdata instances scalar_field_magn_ionID using the ID of the ion as per the configuration of the ranging definitions used.

xdmf_intensity: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [n_xyz]) {units=*NX_ANY*}

Intensity of the field at given point

xdmf_xyz: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_xyz, 3]) {units=*NX_LENGTH*}

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: [i]) {units=*NX_UNITLESS*}

XDMF topology for rendering in combination with xdmf_xyz the scalar field via XDMF in e.g. Paraview.

scalar_field_grad_SUFFIX: (optional) *NXdata* <=

The three-dimensional gradient $\nabla\Phi$. Follow the naming convention of scalar_field_magn_SUFFIX to report parallel structures.

xdmf_gradient: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_xyz, 3]) {units=*NX_ANY*}

The gradient vector formatted for direct visualization via XDMF in e.g. Paraview.

xdmf_xyz: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [n_xyz, 3]) {units=*NX_LENGTH*}

Center of mass positions of each voxel for rendering the scalar field gradient via XDMF in e.g. Paraview.

xdmf_topology: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

XDMF topology for rendering in combination with xdmf_xyz the scalar field via XDMF in e.g. Paraview.

iso_surfaceID: (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 φ.

For applications in atom probe microscopy, the location and shape of such a boundary (set) is typically approximated by discretization - triangulation to be specific.

This yields a complex of not necessarily connected geometric primitives. Paraprobe-nanochem approximates this complex with a soup of triangles.

dimensionality: (required) *NX_POSINT* <=

isovalue: (required) *NX_NUMBER* {units=*NX_ANY*} <=

The threshold or iso-contour value φ.

marching_cubes: (optional) *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*

The resulting triangle soup computed via marching cubes.

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

index_offset: (required) *NX_INT* <=

triangles: (required) *NXcg_face_list_data_structure* <=

number_of_vertices: (required) *NX_POSINT*

number_of_faces: (required) *NX_POSINT*

vertex_index_offset: (required) *NX_INT*

face_index_offset: (required) *NX_INT*

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} <=

Positions of the vertices.

Users are encouraged to reduce the vertices to a unique set as this may result in 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 each vertex is stored even though many share the same positions.

faces: (required) *NX_INT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*} <=

Array of identifiers from vertices which describe each face.

The first entry is the identifier of the start vertex of the first face, followed by the second vertex of the first face, until the last vertex of the first face. Thereafter, the start vertex of the second face, the second vertex of the second face, and so on and so forth.

Therefore, summating over the number_of_vertices, allows to extract the vertex identifiers for the i-th face on the following index interval of the faces array: $[\sum_{i=0}^{i=n-1}, \sum_{i=0}^{i=n}]$.

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_tri_xdmf]) {units=*NX_UNITLESS*}

A list of as many tuples of XDMF topology key, XDMF number of vertices and a triple of vertex indices specifying each triangle. The total number of entries is n_f_tri * (1+1+3).

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [j]) {units=*NX_AREA*} <=

edge_length: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [k, 3]) {units=*NX_LENGTH*}

Array of edge length values. For each triangle the edge length is reported for the edges traversed according to the sequence in which vertices are indexed in triangles.

interior_angle: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [j, 4]) {units=*NX_ANGLE*}

Array of interior angle values. For each triangle the angle is reported for the angle opposite to the edges which are traversed according to the sequence in which vertices are indexed in triangles.

center: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*} <=

The center of mass of each triangle.

vertex_normal: (optional) *NXcg_unit_normal* <=

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* <=

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 the projection is almost perpendicular.

volumetric_features: (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 representation of such iso-surface. This may yield a set of connected features whose individual surfaces are discretized by a triangulated mesh each. Such volumetric features can be processed further using paraprobe-nanochem using a workflow with at most two steps.

In the first step, the tool distinguishes three types of (v) i.e. volumetric features:

1. **So-called objects, i.e. necessarily watertight features represented by polyhedra.**
   These objects were already watertight within the triangulated iso-surface.
2. **So-called proxies, i.e. features that were not necessarily watertight within the triangulated**
   iso-surface but were subsequently replaced by a watertight mesh using polyhedral mesh processing

operations (hole filling, refinement, fairing operations).

3. **Remaining triangle surface meshes or parts of these of arbitrary shape and cardinality**
   that are not transformable into proxies or for which no transformation into proxies was instructed.

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.

In the second step, the tool can be used to analyze the proximity of these objects to a model of the surface (edge) of the dataset.

indices_triangle_cluster: (required) *NX_UINT* (Rank: 1, Dimensions: [n_v_feat]) {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: [n_v_feat]) {units=*NX_UNITLESS*}

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.

indices_feature: (required) *NX_INT* (Rank: 1, Dimensions: [n_v_feat]) {units=*NX_UNITLESS*}

The explicit identifier of features.

FEATURE: (optional) *NXprocess*

In all situations instances of the parent NXprocess group are returned with a very similar information structuring and thus we here replace the template name FEATURE with one of the following types feature-specific group names:

- objects, objects, irrespective their distance to the surface
- objects_close_to_edge, sub-set of v_feature_object close surface

- objects_far_from_edge, sub-set of v_feature_object not close to the surface
- proxies, proxies, irrespective their distance to the surface
- proxies_close_to_edge, sub-set of v_feature_proxies, close to surface
- proxies_far_from_edge, sub-set of v_feature_proxies, not close to surface

indices_feature: (required) *NX_INT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Explicit identifier of the feature a sub-set of the indices_feature in the parent group.

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [i]) {units=*NX_VOLUME*}

Volume of the feature. NaN for non-watertight objects.

obb: (optional) *NXcg_hexahedron*

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 or second-largest over largest and smallest over second largest.

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: (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.

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [k, 3]) {units=*NX_LENGTH*} <=

xdmf_topology: (required) *NX_INT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

indices_feature_xdmf: (required) *NX_INT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

objectID: (optional) *NXcg_polyhedron*

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])

face_normals: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_f, 3]) {units=*NX_LENGTH*}

xdmf_topology: (recommended) *NX_INT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

xdmf_indices_feature: (recommended) *NX_INT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

ion_id: (optional) *NX_UINT* (Rank: 1, Dimensions: [m]) {units=*NX_UNITLESS*}

Array of evaporation_id / identifier_ion which details which ions lie inside or on the surface of the feature.

composition: (optional) *NXchemical_composition*

total: (required) *NX_NUMBER* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*} <=

Total (count) of ions inside or on the surface of the feature relevant for normalization. NaN for non watertight objects.

ATOM: (optional) *NXatom* <=

charge_state: (required) *NX_INT*

nuclide_hash: (required) *NX_UINT* <=

nuclide_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, nuclide, or (molecular) ion within the volume of the feature/object.

interface_meshingID: (optional) *NXapm_paraprobe_tool_process* <=

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_stateID: (optional) *NXcg_triangle*

The triangle surface mesh representing the interface model. Exported at state before or after the next DCOM step.

state: (required) *NX_CHAR*

Was this state exported before or after the next DCOM step.

Any of these values: `before` | `after`

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

index_offset: (required) *NX_INT* <=

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* <=

number_of_vertices: (required) *NX_UINT* <=

number_of_faces: (required) *NX_UINT* <=

index_offset_vertex: (required) *NX_INT* <=

index_offset_edge: (required) *NX_INT* <=

index_offset_face: (required) *NX_INT* <=

indices_face: (required) *NX_INT* (Rank: 1, Dimensions: [j]) <=

vertices: (required) *NX_NUMBER* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*} <=

vertex_normal: (required) *NX_FLOAT* (Rank: 2, Dimensions: [i, 3]) {units=*NX_LENGTH*}

Direction of each vertex normal.

vertex_normal_orientation: (required) *NX_UINT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Qualifier which details how specifically oriented the face normal is with respect to its primitive (triangle):

- 0 - undefined
- 1 - outer
- 2 - inner

faces: (required) *NX_UINT* (Rank: 2, Dimensions: [j, 3])

face_normal: (required) *NX_FLOAT* (Rank: 2, Dimensions: [j, 3]) {units=*NX_LENGTH*}

Direction of each face normal.

face_normal_orientation: (required) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

Qualifier which details how specifically oriented the face normal is with respect to its primitive (triangle):

- 0 - undefined
- 1 - outer
- 2 - inner

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

oned_profileID: (optional) *NXapm_paraprobe_tool_process* <=

xdmf_cylinder: (required) *NXcg_polyhedron*

The ROIs are defined as cylinders for the computations. To visualize these we discretize them into regular n-gons. Using for instance 360-gons, i.e. a regular n-gon with 360 edges, resolves the lateral surface of each cylinder such that their renditions are smooth in visualization software like Paraview.

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

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.

orientation: (required) *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.

roi_id: (optional) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

XDMF support to enable coloring each ROI by its identifier.

edge_contact: (optional) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

XDMF support to enable coloring each ROI whether it has edge contact or not.

number_of_atoms: (optional) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

XDMF support to enable coloring each ROI by its number of atoms.

number_of_ions: (optional) *NX_UINT* (Rank: 1, Dimensions: [j]) {units=*NX_UNITLESS*}

XDMF support to enable coloring each ROI by its number of ions.

rois_far_from_edge: (required) *NXprocess*

Distance and iontype-specific processed data for each ROI. Arrays signed_distance and nuclide_hash are sorted by increasing distance. Array nuclide_hash reports one hash for each atom of each isotope. Effectively, this can yield to groups of values on signed_distance with the same distance value as molecular ions are reported decomposed into their atoms. Therefore, the XDMF support fields number_of_atoms and number_of_ions are only expected to display pairwise the same values respectively, if all ions are built from a single atom only. **roi**ID: (optional) *NXcg_roi*

signed_distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [k]) {units=*NX_LENGTH*}

Sorted in increasing order projected along the positive direction of the ROI as defined by orientation in the parent group.

nuclide_hash: (required) *NX_UINT* (Rank: 1, Dimensions: [k]) {units=*NX_UNITLESS*}

Hashvalue as defined in *NXatom*.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_nanochem_results/ENTRY-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/definition-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/boundaries-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/boundary_conditions-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/face_index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/faces-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/number_of_boundaries-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/vertex_index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/vertices-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/hexahedron/xdmf_topology-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/bounding_box/is_axis_aligned-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/cardinality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/cell_dimensions-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/dimensionality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/extent-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/dimensionality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/isovalue-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/marching_cubes-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/cardinality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/dimensionality-field*

- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/area-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/center-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/edge_length-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/face_index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/face_normal-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/face_normal/gradien field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/face_normal/gradien field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/face_normal/normal field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/face_normal/orienta field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/faces-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/interior_angle-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/number_of_faces-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/number_of_vertices-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/vertex_index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/vertex_normal-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/vertex_normal/norm field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/vertex_normal/orien field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/vertices-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features-group*

- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field*

- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/volumetric_features/ field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/iso_surfaceID/triangle_soup/triangles/xdmf_topology-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/kernel_mu-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/kernel_sigma-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/kernel_size-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/kernel_type-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/normalization-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/origin-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_grad_SUFFIX-group
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_grad_SUFFIX/xdmf_gradient-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_grad_SUFFIX/xdmf_topology-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_grad_SUFFIX/xdmf_xyz-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_magn_SUFFIX-group
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_magn_SUFFIX/xdmf_intensity-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_magn_SUFFIX/xdmf_topology-field
- /NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/scalar_field_magn_SUFFIX/xdmf_xyz-field

- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/grid/symmetry-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/window-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/window/bitdepth-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/window/mask-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/delocalizationID/window/number_of_ions-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/decorator_multiplicity-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/initial_interface-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/initial_interface/point_normal_form-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/ion_multiplicity-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/area-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/cardinality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/dimensionality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/edge_length-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/index_offset-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/interior_angle-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/state-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/dimensionality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/face_normal-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/face_normal_orientation-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/faces-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/index_offset_edge-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/index_offset_face-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/index_offset_vertex-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/indices_face-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/number_of_faces-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/number_of_vertices-field*

- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/vertex_normal-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/vertex_normal_orientation-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/vertices-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/interface_meshingID/mesh_stateID/triangles/xdmf_topology-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/cardinality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/center-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/dimensionality-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/edge_contact-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/number_of_atoms-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/number_of_ions-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/orientation-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/roi_id-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/rois_far_from_edge-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/rois_far_from_edge/roiID-group*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/rois_far_from_edge/roiID/nuclide_hash-field*
- */NXapm_paraprobe_nanochem_results/ENTRY/oned_profileID/xdmf_cylinder/rois_far_from_edge/roiID/signed_distance-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_nanochem_results.nxdl.xml

NXapm_paraprobe_ranger_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-ranger tool.

The tool paraprobe-ranger evaluates how mass-to-charge-state-ratio values map on (molecular) ion types.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_nuclides: 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_paraprobe_tool_parameters, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_ranger_config`

rangeID: (required) *NXapm_paraprobe_tool_parameters* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_ranger_config/ENTRY-group*
- */NXapm_paraprobe_ranger_config/ENTRY/definition-field*
- */NXapm_paraprobe_ranger_config/ENTRY/rangeID-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_ranger_config.nxdl.xml

NXapm_paraprobe_ranger_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for results files of the paraprobe-ranger tool.

The tool paraprobe-ranger evaluates how mass-to-charge-state-ratio values map on (molecular) ion types.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ions: The total number of ions in the reconstructed volume.

Groups cited:
NXapm_paraprobe_tool_process, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_ranger_results`

iontypesID: (required) *NXapm_paraprobe_tool_process* <=

The tool loads ranging definitions from the configuration file and evaluates for each ion to which iontype it matches. If an ion matches on no type, the ion is assume of the default *unknown_type*. In this case, the value *iontypes* is 0. In other cases the value is larger than 0.

iontypes: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

The iontype (identifier) for each ion that was best matching, stored in the order of the evaporation sequence ID.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_ranger_results/ENTRY-group*
- */NXapm_paraprobe_ranger_results/ENTRY/definition-field*
- */NXapm_paraprobe_ranger_results/ENTRY/iontypesID-group*
- */NXapm_paraprobe_ranger_results/ENTRY/iontypesID/iontypes-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_ranger_results.nxdl.xml

NXapm_paraprobe_selector_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-selector tool.

Symbols:

No symbol table

Groups cited:
NXapm_paraprobe_tool_parameters, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_selector_config`

selectID: (required) *NXapm_paraprobe_tool_parameters* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_selector_config/ENTRY-group*
- */NXapm_paraprobe_selector_config/ENTRY/definition-field*
- */NXapm_paraprobe_selector_config/ENTRY/selectID-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_selector_config.nxdl.xml

NXapm_paraprobe_selector_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-selector tool.

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:
NXapm_paraprobe_tool_process, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_selector_results`

roiID: (required) *NXapm_paraprobe_tool_process* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_selector_results/ENTRY-group*
- */NXapm_paraprobe_selector_results/ENTRY/definition-field*
- */NXapm_paraprobe_selector_results/ENTRY/roiID-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_selector_results.nxdl.xml

NXapm_paraprobe_spatstat_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-spatstat tool.

The tool paraprobe-spatstat evaluates spatial distribution functions.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_ivec_max: Maximum number of atoms per molecular ion. Should be 32 for paraprobe.

n_ion_source: Number of different source iontypes to distinguish.

n_ion_target: Number of different target iontypes to distinguish.

Groups cited:
NXapm_paraprobe_tool_parameters, *NXcs_prng*, *NXentry*, *NXnote*, *NXprocess*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_spatstat_config`

spatial_statisticsID: (required) *NXapm_paraprobe_tool_parameters* <=

randomize_iontypes: (required) *NX_BOOLEAN*

Specifies, if the iontypes are randomized for the point cloud or not. Internally, paraprobe uses a sequentially executed deterministic MT19987 (MersenneTwister) pseudo-random number generator to shuffle the iontypes randomly across the entire set of ions. That is the total number of ions of either type remain the same but the information about their location is randomized.

ion_query_type_source: (required) *NX_CHAR*

How should the iontype be interpreted on the source-side, i.e. all these ion positions where a regions-of-interest (ROI) around so-called source ions will be placed. Different options exist how iontypes are interpreted given an iontype represents in general a (molecular) ion with different isotopes that have individually different multiplicity.

The value resolve_all will set an ion active in the analysis regardless of which iontype it is. Each active ion is accounted for once.

The value resolve_unknown will set an ion active when the ion is of the UNKNOWNTYPE type. Each active ion is accounted for once.

The value resolve_ion will set an ion active if it is of the specific iontype, irregardless of its elemental or isotopic details. Each active ion is counted once.

The value resolve_element will set an ion active, and most importantly, account for each as many times as the (molecular) ion contains atoms of elements in the whitelist ion_query_isotope_vector.

The value resolve_isotope will set an ion active, and most importantly, account for each as many times as the (molecular) ion contains isotopes in the whitelist ion_query_isotope_vector.

In effect, ion_query_isotope_vector acts as a whitelist to filter which ions are considered as source ions of the correlation statistics and how the multiplicity of each ion will be factorized, i.e. how often it is accounted for.

Any of these values:

- `resolve_all`
- `resolve_unknown`
- `resolve_ion`
- `resolve_element`
- `resolve_isotope`

ion_query_nuclide_source: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ion_source, n_ivec_max]) {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_nuclide_target: (required) *NX_UINT* (Rank: 2, Dimensions: [n_ion_target, n_ivec_max]) {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.

surface_distance: (optional) *NXnote* <=

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR* <=

edge_distance: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Threshold to define how far an ion has to lay at least from the edge of the dataset so that the ion can act as a source. This means that an ROI is placed at the location of the ion and its neighbors are analyzed how they contribute to the computed statistics.

The edge_distance threshold can be combined with the feature_distance threshold. This threshold defines defines up to which distance to a microstructural feature an ROI is placed.

The threshold is useful to process the dataset such that ROIs do not protrude out of the dataset as this would add bias.

feature_distance: (optional) *NXnote* <=

Distance between each ion and triangulated mesh of microstructural features. 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 the feature_distance threshold value.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR*

Absolute path in the (HDF5) file which points to the distance of each ion to the closest feature.

feature_distance: (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 this feature_distance threshold is used in combination with the edge_distance threshold when placing ROI about source ions.

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.

kth: (required) *NX_UINT* {units=*NX_UNITLESS*}

Order k.

min: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Minimum value of the histogram binning.

increment: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Increment of the histogram binning.

max: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Maximum value of the histogram binning.

rdf: (optional) *NXprocess*

Compute radial distribution function.

min: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Minimum value of the histogram binning.

increment: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Increment value of the histogram binning.

max: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Maximum value of the histogram binning.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_spatstat_config/ENTRY-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/definition-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/feature_distance-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/feature_distance/algorithm-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/feature_distance/checksum-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/feature_distance/distance-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/feature_distance/feature_distance-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/feature_distance/file_name-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/ion_query_nuclide_source-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/ion_query_nuclide_target-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/ion_query_type_source-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/ion_query_type_target-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/random_number_generator-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/random_number_generator/seed-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/random_number_generator/type-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/random_number_generator/warmup-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/randomize_iontypes-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/knn-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/knn/increment-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/knn/kth-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/knn/max-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/knn/min-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/rdf-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/rdf/increment-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/rdf/max-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/statistics/rdf/min-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/surface_distance-group*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/surface_distance/algorithm-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/surface_distance/checksum-field*

- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/surface_distance/distance-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/surface_distance/edge_distance-field*
- */NXapm_paraprobe_spatstat_config/ENTRY/spatial_statisticsID/surface_distance/file_name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_spatstat_config.nxdl.xml

NXapm_paraprobe_spatstat_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-spatstat tool.

The tool paraprobe-spatstat evaluates spatial distribution functions.

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_knn: The total number of bins in the histogram for the k-th nearest neighbor.

n_rdf: The total number of bins in the histogram for the radial distribution function.

Groups cited:
NXapm_paraprobe_tool_process, *NXentry*, *NXprocess*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_spatstat_results`

spatial_statisticsID: (required) *NXapm_paraprobe_tool_process* <=

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 remain the same.

The order matches the iontypes array from a given ranging results as it is specified in the configuration settings inside the specific config_filename that was used for this paraprobe-spatstat analysis.

knn: (optional) *NXprocess*

K-nearest neighbor statistics.

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_knn]) {units=*NX_LENGTH*}

Right boundary of the binning.

probability_mass: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_knn]) {units=*NX_DIMENSIONLESS*}

cumulated: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_knn]) {units=*NX_UNITLESS*}

Cumulated not normalized by total counts.

cumulated_normalized: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_knn]) {units=*NX_DIMENSIONLESS*}

Cumulated and normalized by total counts.

rdf: (optional) *NXprocess*

Radial distribution statistics.

distance: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_rdf]) {units=*NX_LENGTH*}

Right boundary of the binning.

probability_mass: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_rdf]) {units=*NX_DIMENSIONLESS*}

cumulated: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_rdf]) {units=*NX_UNITLESS*}

Cumulated not normalized by total counts.

cumulated_normalized: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_rdf]) {units=*NX_DIMENSIONLESS*}

Cumulated and normalized by total counts.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_spatstat_results/ENTRY-group*
- */NXapm_paraprobe_spatstat_results/ENTRY/definition-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID-group*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/iontypes_randomized-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/knn-group*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/knn/cumulated-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/knn/cumulated_normalized-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/knn/distance-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/knn/probability_mass-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/rdf-group*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/rdf/cumulated-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/rdf/cumulated_normalized-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/rdf/distance-field*
- */NXapm_paraprobe_spatstat_results/ENTRY/spatial_statisticsID/rdf/probability_mass-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_spatstat_results.nxdl.xml

NXapm_paraprobe_surfacer_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-surfacer tool.

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_paraprobe_tool_parameters, *NXentry*, *NXparameters*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_surfacer_config`

surface_meshingID: (required) *NXapm_paraprobe_tool_parameters* <=

alpha_value_choice: (required) *NX_CHAR*

Specifies which method to use to define the alpha value.

The value *convex_hull_naive* is the default. The setting 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* instructs to tool to refine the quality of the mesh resulting from *convex_hull_naive* via triangle flipping and splitting.

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 CGAL library to choose a value which the library considers as to be an optimal value. Details are defined in the respective section of the CGAL library on 3D alpha shapes.

The value *set_of_values* instructs the tool to compute a list collection of alpha-shapes for the specified alpha-values.

The value *set_of_alpha_wrappings* instructs the tool to generate a set of so-called alpha wrappings. These are similar to alpha-shapes but provide additional guarantees (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).

preprocessing: (required) *NXparameters* <=

method: (required) *NX_CHAR*

Specifies the method that is used to preprocess the point cloud prior to the alpha-shape computation.

The option *default* specifies that no such filtering is applied. The option *percolation* specifies that a Hoshen-Kopelman percolation analysis is used to identify points that lie closer to the edge of the dataset. Details about the methods are reported in M. Kühbach et al..

Any of these values: `default|percolation`

kernel_width: (required) *NX_UINT* {units=*NX_UNITLESS*}

When using the *percolation* preprocessing, this is the width of the kernel for identifying which ions are in voxels close to the edge of the point cloud.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_surfacer_config/ENTRY-group*
- */NXapm_paraprobe_surfacer_config/ENTRY/definition-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID-group*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/alpha_value_choice-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/alpha_values-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/has_closure-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/has_exterior_facets-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/has_interior_tetrahedra-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/offset_values-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/preprocessing-group*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/preprocessing/kernel_width-field*
- */NXapm_paraprobe_surfacer_config/ENTRY/surface_meshingID/preprocessing/method-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_surfacer_config.nxdl.xml

NXapm_paraprobe_surfacer_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-surfacer tool.

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:
NXapm_paraprobe_tool_process, *NXcg_alpha_complex*, *NXcg_face_list_data_structure*, *NXcg_tetrahedron*, *NXcg_triangle*, *NXcs_filter_boolean_mask*, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_surfacer_results`

point_set_wrappingID: (required) *NXapm_paraprobe_tool_process* <=

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**ID: (optional) *NXcg_alpha_complex*

dimensionality: (required) *NX_POSINT* <=

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_ANY*} <=

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 that alpha_complex instance was computed.

This window can be different to the window of the *point_set_wrapping* parent group because irrelevant ions might have been filtered out in addition to the window defined in *point_set_wrapping* to reduce e.g. computational costs of the alpha complex computation.

number_of_ions: (required) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask.

bitdepth: (required) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype.

mask: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The bitfield of the mask. See *NXcs_filter_boolean_mask* for how this bitfield is to be interpreted.

triangle_set: (optional) *NXcg_triangle* <=

The set of triangles in the coordinate system paraprobe which discretizes the exterior surface of the alpha complex.

index_offset: (required) *NX_INT* <=

triangles: (required) *NXcg_face_list_data_structure* <=

dimensionality: (required) *NX_POSINT* <=

number_of_vertices: (required) *NX_UINT* <=

number_of_faces: (required) *NX_UINT* <=

indices_offset_vertex: (required) *NX_INT*

indices_offset_face: (required) *NX_INT*

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])

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_tri_xdmf]) {units=*NX_UNITLESS*}

A list of as many tuples of XDMF topology key, XDMF number of vertices and a triple of vertex indices specifying each triangle. The total number of entries is n_f_tri * (1+1+3).

is_watertight: (optional) *NX_BOOLEAN*

Do the triangles define a triangulated surface mesh that is watertight?

volume: (optional) *NX_FLOAT* {units=*NX_VOLUME*}

The volume which the triangulated surface mesh encloses if that mesh is watertight.

interior_tetrahedra: (optional) *NXcg_tetrahedron* <=

The set of tetrahedra which represent the interior volume of the complex if that is a closed two-manifold.

index_offset: (required) *NX_INT* <=

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_UINT* <=

number_of_faces: (required) *NX_UINT* <=

indices_offset_vertex: (required) *NX_INT*

indices_offset_face: (required) *NX_INT*

vertices: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_v_tet, 3]) {units=*NX_LENGTH*}

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_tet_xdmf]) {units=*NX_UNITLESS*}

A list of as many tuples of XDMF topology key, XDMF number of vertices and a triple of vertex indices specifying each triangle. The total number of entries is n_f_tet * (1+1+4).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_surfacer_results/ENTRY-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/definition-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/alpha-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/dimensionality-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/index_offset-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra/indices_of field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra/indices_of field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra/number_o field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra/number_o field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra/vertices-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/tetrahedra/xdmf_topo field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/interior_tetrahedra/volume-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/mode-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/offset-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/index_offset-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/dimensionality-field*

- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/faces-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/indices_offset_face-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/indices_offset_verte-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/is_watertight-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/number_of_faces-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/number_of_vertices-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/vertices-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/volume-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/triangle_set/triangles/xdmf_topology-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/type-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/window-group*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/window/bitdepth-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/window/mask-field*
- */NXapm_paraprobe_surfacer_results/ENTRY/point_set_wrappingID/alpha_complexID/window/number_of_ions-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_surfacer_results.nxdl.xml

NXapm_paraprobe_tessellator_config

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_config*

Description:

Application definition for a configuration file of the paraprobe-tessellator tool.

The tool paraprobe-tessellator computes a tessellation of the reconstructed positions.

Symbols:

No symbol table

Groups cited:
NXapm_paraprobe_tool_parameters, *NXentry*, *NXnote*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_tessellator_config`

tessellateID: (required) *NXapm_paraprobe_tool_parameters* <=

method: (required) *NX_CHAR*

The method used to compute the tessellation. The value *default* configures the computation of the Voronoi tessellation.

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 for each cell if it 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 likely 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 anymore by the boundary. This is valuable information to judge about the significance of finite size effects.

surface_distance: (optional) *NXnote* <=

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

distance: (required) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tessellator_config/ENTRY-group*
- */NXapm_paraprobe_tessellator_config/ENTRY/definition-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID-group*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/has_cell_edge_detection-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/has_cell_geometry-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/has_cell_neighbors-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/has_cell_volume-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/method-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/surface_distance-group*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/surface_distance/algorithm-field*

- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/surface_distance/checksum-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/surface_distance/distance-field*
- */NXapm_paraprobe_tessellator_config/ENTRY/tessellateID/surface_distance/file_name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tessellator_config.nxdl.xml

NXapm_paraprobe_tessellator_results

Status:

application definition (contribution), extends *NXapm_paraprobe_tool_results*

Description:

Application definition for a results file of the paraprobe-tessellator tool.

The tool paraprobe-tessellator computes a tessellation of the reconstructed positions.

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: The total number of values required to represent all faces of each cell.

n_f_xdmf: The total number of values required to represent all faces of each cell (polyhedron) using XDMF.

Groups cited:
NXapm_paraprobe_tool_process, *NXcg_face_list_data_structure*, *NXcg_hexahedron*, *NXcg_polyhedron*, *NXcs_filter_boolean_mask*, *NXentry*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXapm_paraprobe_tessellator_results`

tessellationID: (required) *NXapm_paraprobe_tool_process* <=

The tool can be used to compute a Voronoi tessellation the entire or of a sub-set of the reconstructed volume. Each point (ion) is wrapped in one (Voronoi) cell. The point cloud in the ROI is wrapped into an axis-aligned bounding box (AABB) that is tight. This means points at the edge of the point cloud can lay on the surface of the bounding box. The tool detects if cells make contact with the walls of this bounding box. The tessellation is computed without periodic boundary conditions. **wall**: (recommended) *NXcg_hexahedron*

The (tight) axis-aligned bounding box about the point cloud.

closest_corner: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Coordinate triplet of the corner that lays closest to the origin of the *paraprobe* coordinate system.

farthest_corner: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Coordinate triplet of the corner that lays farthest away from the origin of the *paraprobe* coordinate system.

voronoi_cells: (optional) *NXcg_polyhedron*

dimensionality: (required) *NX_POSINT* <=

Obligatory value: 3

cardinality: (required) *NX_POSINT* <=

The number of points (and thus cells).

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_VOLUME*}

Volume of each Voronoi cell.

process_id: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

Which MPI process computed which Voronoi cell.

thread_id: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*}

Which OpenMP thread computed which Voronoi cell.

number_of_faces: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) {units=*NX_UNITLESS*} <=

The number of faces for each cell. Faces of adjoining polyhedra are counted for each polyhedron. This field can be used to interpret the concatenated vector with the individual values for the area of each face.

index_offset: (required) *NX_INT* <=

xdmf_topology: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f_xdmf]) {units=*NX_UNITLESS*}

Sequence of tuples, concatenated in the order of the Voronoi cells. Each tuple contains encodes information to visualize using XDMF: Firstly, an XDMF geometric primitive type key. Secondly, the number of vertices of the polygon. Third, the sequence of indices_vertex which define the facet. Tuples encode faces faster than cells.

xdmf_cell_id: (required) *NX_UINT* (Rank: 1, Dimensions: [n_f]) {units=*NX_UNITLESS*}

Sequence of cell identifier, concatenated such that each face is associated with its cell. Given that paraprobe-tessellator assigns each cell the evaporation_id of the ion that the cell wraps this information enables the segmentation of the tessellation and thus correlate per-ion properties with the volume that each cell represents.

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 purposes.

number_of_vertices: (required) *NX_UINT* <=

number_of_faces: (required) *NX_UINT* <=

indices_offset_vertex: (required) *NX_INT*

indices_offset_face: (required) *NX_INT*

vertices: (required) *NX_FLOAT*

wall_contact_global: (recommended) *NXcs_filter_boolean_mask*

A bitmask that documents which of the cells are likely truncated because they share at least one face with the *aabb* of the point cloud. This field encodes the result of the boolean or operator applied to the value of all six wall_contact groups that document contact in specific outer unit normal directions of the *aabb*.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

wall_contact_left: (recommended) *NXcs_filter_boolean_mask*

In the spirit of wall_contact_global, the left face of *aabb*. Its outer unit normal points in the opposite direction of the x-axis of the *paraprobe* coordinate system.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

wall_contact_right: (recommended) *NXcs_filter_boolean_mask*

In the spirit of wall_contact_global, the right face of *aabb*. Its outer unit normal points in the direction of the x-axis of the *paraprobe* coordinate system.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

wall_contact_front: (recommended) *NXcs_filter_boolean_mask*

In the spirit of wall_contact_global, the front face of *aabb*. Its outer unit normal points in the opposite direction of the y-axis of the *paraprobe* coordinate system.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

wall_contact_rear: (recommended) *NXcs_filter_boolean_mask*

In the spirit of wall_contact_global, the rear face of *aabb*. Its outer unit normal points in the direction of the y-axis of the *paraprobe* coordinate system.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

wall_contact_bottom: (recommended) *NXcs_filter_boolean_mask*

In the spirit of wall_contact_global, the front face of *aabb*. Its outer unit normal points in the opposite direction of the z-axis of the *paraprobe* coordinate system.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

wall_contact_top: (recommended) *NXcs_filter_boolean_mask*

In the spirit of wall_contact_global, the front face of *aabb*. Its outer unit normal points in the direction of the z-axis of the *paraprobe* coordinate system.

number_of_objects: (required) *NX_UINT* (Rank: 1, Dimensions: [n_ions]) <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tessellator_results/ENTRY-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/definition-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/cardinality-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/dimensionality-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/index_offset-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/number_of_faces-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/polyhedra-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/polyhedra/indices_offset_face-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/polyhedra/indices_offset_vertex-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/polyhedra/number_of_faces-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/polyhedra/number_of_vertices-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/polyhedra/vertices-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/process_id-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/thread_id-field*

- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/volume-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/xdmf_cell_id-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/voronoi_cells/xdmf_topology-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall/closest_corner-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall/farthest_corner-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_bottom-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_bottom/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_bottom/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_bottom/number_of_objects-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_front-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_front/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_front/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_front/number_of_objects-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_global-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_global/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_global/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_global/number_of_objects-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_left-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_left/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_left/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_left/number_of_objects-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_rear-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_rear/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_rear/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_rear/number_of_objects-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_right-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_right/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_right/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_right/number_of_objects-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_top-group*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_top/bitdepth-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_top/mask-field*
- */NXapm_paraprobe_tessellator_results/ENTRY/tessellationID/wall_contact_top/number_of_objects-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tessellator_results.nxdl.xml

NXapm_paraprobe_tool_common

Status:

base class (contribution), extends *NXobject*

Description:

Base class documenting organizational metadata used by all tools of the paraprobe-toolbox.

Symbols:

No symbol table

Groups cited:
NXcoordinate_system, *NXcs_profiling*, *NXnote*, *NXprogram*, *NXuser*

Structure:

status: (optional) *NX_CHAR*

A statement whether the tool executable managed to process the analysis or whether this failed. Status is written to the results file after the end_time beyond which point in time the tool must no longer compute any further analysis results but exit.

Only when this status message is present and its value is *success*, one should consider the results of the tool. In all other cases it might be that the tool has terminated prematurely or another error occurred.

Any of these values: `success | failure`

identifier_analysis: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Internal identifier used by the tool to refer to an analysis. Simulation ID is an alias.

config: (optional) *NXnote* <=

The configuration file that was used to parameterize the algorithms that this tool has executed.

programID: (optional) *NXprogram*

profiling: (optional) *NXcs_profiling*

start_time: (optional) *NX_DATE_TIME* <=

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis in this results file was started, i.e. when the respective executable/tool was started as a process.

end_time: (optional) *NX_DATE_TIME* <=

ISO 8601 formatted time code with local time zone offset to UTC information included when the analysis in this results file were completed and the respective process of the tool exited.

total_elapsed_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

Wall-clock time.

userID: (optional) *NXuser*

NAMED_**reference_frame**ID: (optional) *NXcoordinate_system*

Details about coordinate systems (reference frames) used. In atom probe several coordinate systems have to be distinguished. Names of instances of such *NXcoordinate_system* should be documented explicitly and doing so by picking from the following controlled set of names:

- paraprobe_reference_frame
- lab_reference_frame
- specimen_reference_frame
- laser_reference_frame
- instrument_reference_frame
- detector_reference_frame
- reconstruction_reference_frame

The aim of this convention is to support users with contextualizing which reference frame each instance (coordinate system) is. If needed, instances of *NXtransformations* are used to detail the explicit affine transformations whereby one can convert representations between different reference frames. Inspect *NXtransformations* for further details.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tool_common/config-group*
- */NXapm_paraprobe_tool_common/identifier_analysis-field*
- */NXapm_paraprobe_tool_common/NAMED_reference_frameID-group*
- */NXapm_paraprobe_tool_common/profiling-group*
- */NXapm_paraprobe_tool_common/profiling/end_time-field*
- */NXapm_paraprobe_tool_common/profiling/start_time-field*
- */NXapm_paraprobe_tool_common/profiling/total_elapsed_time-field*
- */NXapm_paraprobe_tool_common/programID-group*
- */NXapm_paraprobe_tool_common/status-field*
- */NXapm_paraprobe_tool_common/userID-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tool_common.nxdl.xml

NXapm_paraprobe_tool_config

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for a (configuration) file of a tool from the paraprobe-toolbox.

The paraprobe-toolbox is a collection of open-source tools for performing efficient analyses of point cloud data where each point can represent atoms or (molecular) ions. A key application of the toolbox has been for research in the field of Atom Probe Tomography (APT) and related Field Ion Microscopy (FIM):

- paraprobe-toolbox
- M. Kühbach et al.

The toolbox does not replace but complements existent software tools in this research field. Given its capabilities of handling points as objects with properties and enabling analyses of the spatial arrangement of and inter- sections between geometric primitives, the software can equally be used for analyzing data in Materials Science and Materials Engineering.

Symbols:

No symbol table

Groups cited:

NXapm_paraprobe_tool_common, *NXapm_paraprobe_tool_parameters*, *NXcg_cylinder*, *NXcg_ellipsoid*, *NXcg_face_list_data_structure*, *NXcg_hexahedron*, *NXcg_polyhedron*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXmatch_filter*, *NXnote*, *NXprogram*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

TASKCONFIG: (required) *NXapm_paraprobe_tool_parameters*

A specific configuration to achieve a processing result

identifier_analysis: (recommended) *NX_UINT* <=

reconstruction: (required) *NXnote* <=

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

position: (required) *NX_CHAR* <=

mass_to_charge: (required) *NX_CHAR* <=

ranging: (required) *NXnote* <=

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

ranging_definitions: (required) *NX_CHAR* <=

spatial_filter: (required) *NXspatial_filter* <=

windowing_method: (required) *NX_CHAR* <=

hexahedron_set: (optional) *NXcg_hexahedron* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

index_offset: (required) *NX_INT* <=

hexahedra: (required) *NXcg_face_list_data_structure* <=

vertices: (required) *NX_UINT*

cylinder_set: (optional) *NXcg_cylinder* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

index_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

height: (required) *NX_NUMBER* <=

radii: (required) *NX_NUMBER* <=

ellipsoid_set: (optional) *NXcg_ellipsoid* <=

dimensionality: (required) *NX_POSINT* <=

cardinality: (required) *NX_POSINT* <=

index_offset: (required) *NX_INT* <=

center: (required) *NX_NUMBER* <=

half_axes_radii: (required) *NX_NUMBER*

orientation: (required) *NX_NUMBER* <=

polyhedron_set: (optional) *NXcg_polyhedron* <=

bitmask: (optional) *NXcs_filter_boolean_mask* <=

number_of_objects: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

evaporation_id_filter: (optional) *NXsubsampling_filter* <=

min: (required) *NX_INT* <=

increment: (required) *NX_INT* <=

max: (required) *NX_INT* <=

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* <=

common: (required) *NXapm_paraprobe_tool_common*

status: (required) *NX_CHAR* <=

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@**version**: (required) *NX_CHAR* <=

profiling: (recommended) *NXcs_profiling* <=

start_time: (required) *NX_DATE_TIME* <=

end_time: (required) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_FLOAT* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tool_config/ENTRY-group*
- */NXapm_paraprobe_tool_config/ENTRY/common-group*
- */NXapm_paraprobe_tool_config/ENTRY/common/profiling-group*
- */NXapm_paraprobe_tool_config/ENTRY/common/profiling/end_time-field*
- */NXapm_paraprobe_tool_config/ENTRY/common/profiling/start_time-field*
- */NXapm_paraprobe_tool_config/ENTRY/common/profiling/total_elapsed_time-field*
- */NXapm_paraprobe_tool_config/ENTRY/common/programID-group*
- */NXapm_paraprobe_tool_config/ENTRY/common/programID/program-field*
- */NXapm_paraprobe_tool_config/ENTRY/common/programID/program@version-attribute*
- */NXapm_paraprobe_tool_config/ENTRY/common/status-field*
- */NXapm_paraprobe_tool_config/ENTRY/definition-field*
- */NXapm_paraprobe_tool_config/ENTRY/definition@version-attribute*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/evaporation_id_filter-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/evaporation_id_filter/increment-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/evaporation_id_filter/max-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/evaporation_id_filter/min-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/hit_multiplicity_filter-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/hit_multiplicity_filter/match-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/hit_multiplicity_filter/method-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/identifier_analysis-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/iontype_filter-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/iontype_filter/match-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/iontype_filter/method-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/ranging-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/ranging/algorithm-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/ranging/checksum-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/ranging/file_name-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/ranging/ranging_definitions-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/reconstruction-group*

- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/reconstruction/algorithm-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/reconstruction/checksum-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/reconstruction/file_name-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/reconstruction/mass_to_charge-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/reconstruction/position-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/bitmask-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/bitmask/bitdepth-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/bitmask/mask-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/bitmask/number_of_objects-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set/cardinality-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set/center-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set/dimensionality-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set/height-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set/index_offset-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/cylinder_set/radii-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set/cardinality-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set/center-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set/dimensionality-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set/half_axes_radii-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set/index_offset-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/ellipsoid_set/orientation-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/hexahedron_set-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/hexahedron_set/cardinality-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/hexahedron_set/dimensionality-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/hexahedron_set/hexahedra-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/hexahedron_set/hexahedra/vertices-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/hexahedron_set/index_offset-field*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/polyhedron_set-group*
- */NXapm_paraprobe_tool_config/ENTRY/TASKCONFIG/spatial_filter/windowing_method-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tool_config.nxdl.xml

NXapm_paraprobe_tool_parameters

Status:

base class (contribution), extends *NXparameters*

Description:

Base class documenting parameters for processing used by all tools of the paraprobe-toolbox.

Symbols:

No symbol table

Groups cited:
NXmatch_filter, *NXnote*, *NXspatial_filter*, *NXsubsampling_filter*

Structure:

identifier_analysis: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Internal identifier used by the tool to refer to an analysis. Simulation ID an alias.

description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

Although offered here for convenience, we strongly encourage to parameterize such descriptions as much as possible to support reusage and clearer communication.

reconstruction: (optional) *NXnote* <=

Specification of the tomographic reconstruction to use for this analysis.

Typically, reconstructions in the field of atom probe tomography are communicated via files which store at least reconstructed ion positions and mass-to-charge-state-ratio values. Container files like HDF5 though can store multiple reconstructions. Therefore, the position and mass_to_charge concepts point to specific instances to use for this analysis.

position: (optional) *NX_CHAR*

Name of the node which resolves the reconstructed ion position values to use for this analysis.

mass_to_charge: (optional) *NX_CHAR*

Name of the node which resolves the mass-to-charge-state-ratio values to use for this analysis.

ranging: (optional) *NXnote* <=

Specification of the ranging definitions to use for this analysis.

Ranging is the process of labeling time-of-flight data with so-called iontypes (aka ion species). Ideally, iontypes specify the most likely (molecular) ion that is assumed to have been evaporated given that its mass-to-charge-state ratio lies within the specific mass-to-charge-state-ratio value interval of the iontype.

The so-called unknown_type iontype represents the null model of an ion that has not been ranged (for whatever reasons) or is not rangeable. The identifier of this special iontype is always the reserved value 0.

ranging_definitions: (optional) *NX_CHAR*

Name of the (parent) node directly below which (in the hierarchy) the ranging definition for (molecular) ions are stored.

surface: (optional) *NXnote* <=

Specification of the triangulated surface mesh to use for this analysis.

Such a surface mesh can be used to define the edge of the reconstructed volume to account for finite size effects.

surface_distance: (optional) *NXnote* <=

Specification of the point-to-triangulated-surface-mesh distances to use for this analysis.

distance: (optional) *NX_CHAR*

Absolute path in the (HDF5) file that points to the distance values. The tool assumes that the values are stored in the same order as points (ions).

spatial_filter: (optional) *NXspatial_filter*

evaporation_id_filter: (optional) *NXsubsampling_filter*

iontype_filter: (optional) *NXmatch_filter*

hit_multiplicity_filter: (optional) *NXmatch_filter*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tool_parameters/description-field*
- */NXapm_paraprobe_tool_parameters/evaporation_id_filter-group*
- */NXapm_paraprobe_tool_parameters/hit_multiplicity_filter-group*
- */NXapm_paraprobe_tool_parameters/identifier_analysis-field*
- */NXapm_paraprobe_tool_parameters/iontype_filter-group*
- */NXapm_paraprobe_tool_parameters/ranging-group*
- */NXapm_paraprobe_tool_parameters/ranging/ranging_definitions-field*
- */NXapm_paraprobe_tool_parameters/reconstruction-group*
- */NXapm_paraprobe_tool_parameters/reconstruction/mass_to_charge-field*
- */NXapm_paraprobe_tool_parameters/reconstruction/position-field*
- */NXapm_paraprobe_tool_parameters/spatial_filter-group*
- */NXapm_paraprobe_tool_parameters/surface-group*
- */NXapm_paraprobe_tool_parameters/surface_distance-group*
- */NXapm_paraprobe_tool_parameters/surface_distance/distance-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tool_parameters.nxdl.xml

NXapm_paraprobe_tool_process

Status:

base class (contribution), extends *NXprocess*

Description:

Base class documenting a processing step within a tool of the paraprobe-toolbox.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_mask: The number of entries in the mask.

Groups cited:
NXcs_filter_boolean_mask

Structure:

description: (optional) *NX_CHAR*

Possibility for leaving a free-text description about this analysis.

window: (optional) *NXcs_filter_boolean_mask*

A bitmask which identifies all ions considered in the analysis.

number_of_ions: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of ions covered by the mask. By default, the total number of ions in the dataset.

bitdepth: (optional) *NX_UINT* {units=*NX_UNITLESS*} <=

Number of bits assumed matching on a default datatype.

mask: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_mask]) {units=*NX_UNITLESS*} <=

The mask. The length of the mask is an integer multiple of bitdepth. In such case, padded bits are set to 0.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tool_process/description-field*
- */NXapm_paraprobe_tool_process/window-group*
- */NXapm_paraprobe_tool_process/window/bitdepth-field*
- */NXapm_paraprobe_tool_process/window/mask-field*
- */NXapm_paraprobe_tool_process/window/number_of_ions-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tool_process.nxdl.xml

NXapm_paraprobe_tool_results

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for storing processing results of a tool from the paraprobe-toolbox.

The paraprobe-toolbox is a collection of open-source tools for performing efficient analyses of point cloud data where each point can represent atoms or (molecular) ions. A key application of the toolbox has been for research in the field of Atom Probe Tomography (APT) and related Field Ion Microscopy (FIM):

- paraprobe-toolbox
- M. Kühbach et al.

The toolbox does not replace but complements existent software tools in this research field. Given its capabilities of handling points as objects with properties and enabling analyses of the spatial arrangement of and inter- sections between geometric primitives, the software can equally be used for analyzing data in Materials Science and Materials Engineering.

Symbols:

No symbol table

Groups cited:

NXapm_paraprobe_tool_common, *NXapm_paraprobe_tool_process*, *NXcoordinate_system*, *NXcs_filter_boolean_mask*, *NXcs_profiling*, *NXentry*, *NXnote*, *NXprogram*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

TASKPROCESSED: (required) *NXapm_paraprobe_tool_process*

A specific processing result

window: (required) *NXcs_filter_boolean_mask* <=

number_of_ions: (required) *NX_UINT* <=

bitdepth: (required) *NX_UINT* <=

mask: (required) *NX_UINT* <=

common: (required) *NXapm_paraprobe_tool_common*

status: (required) *NX_CHAR* <=

identifier_analysis: (required) *NX_UINT* <=

config: (required) *NXnote* <=

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

programID: (required) *NXprogram* <=

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

profiling: (recommended) *NXcs_profiling* <=

start_time: (required) *NX_DATE_TIME* <=

end_time: (required) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_FLOAT* <=

max_processes: (required) *NX_UINT* <=

max_threads: (required) *NX_UINT* <=

max_gpus: (required) *NX_UINT* <=

userID: (optional) *NXuser* <=

If used, metadata of at least the person who performed this analysis.

name: (required) *NX_CHAR* <=

paraprobe_reference_frame: (required) *NXcoordinate_system* <=

type: (required) *NX_CHAR* <=

x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*} <=

x_alias: (required) *NX_CHAR* <=

y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*} <=

y_alias: (required) *NX_CHAR* <=

z: (required) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*} <=

z_alias: (required) *NX_CHAR* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXapm_paraprobe_tool_results/ENTRY-group*
- */NXapm_paraprobe_tool_results/ENTRY/common-group*
- */NXapm_paraprobe_tool_results/ENTRY/common/config-group*
- */NXapm_paraprobe_tool_results/ENTRY/common/config/algorithm-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/config/checksum-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/config/file_name-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/identifier_analysis-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame-group*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/type-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/x-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/x_alias-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/y-field*

- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/y_alias-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/z-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/paraprobe_reference_frame/z_alias-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling-group*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling/end_time-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling/max_gpus-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling/max_processes-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling/max_threads-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling/start_time-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/profiling/total_elapsed_time-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/programID-group*
- */NXapm_paraprobe_tool_results/ENTRY/common/programID/program-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/programID/program@version-attribute*
- */NXapm_paraprobe_tool_results/ENTRY/common/status-field*
- */NXapm_paraprobe_tool_results/ENTRY/common/userID-group*
- */NXapm_paraprobe_tool_results/ENTRY/common/userID/name-field*
- */NXapm_paraprobe_tool_results/ENTRY/definition-field*
- */NXapm_paraprobe_tool_results/ENTRY/definition@version-attribute*
- */NXapm_paraprobe_tool_results/ENTRY/TASKPROCESSED-group*
- */NXapm_paraprobe_tool_results/ENTRY/TASKPROCESSED/window-group*
- */NXapm_paraprobe_tool_results/ENTRY/TASKPROCESSED/window/bitdepth-field*
- */NXapm_paraprobe_tool_results/ENTRY/TASKPROCESSED/window/mask-field*
- */NXapm_paraprobe_tool_results/ENTRY/TASKPROCESSED/window/number_of_ions-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXapm_paraprobe_tool_results.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 instances of NXbeam 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`

polarizing: (optional) *NX_BOOLEAN*

Is the beam splitter polarizing?

multiple_outputs: (optional) *NX_BOOLEAN*

Does the beam splitter have multiple outputs (diffractive optical element), i.e. more than two outputs?

splitting_ratio: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_outputs]) {units=*NX_UNITLESS*}

Beam splitting ratio(s) for the various outputs (i.e. the paths of the beam after being split by the beam splitter). The order of the ratios must be consistent with the labels 1, 2, … N_outputs defined by the sketch in 'SHAPE/sketch', starting with 1.

clear_aperture: (optional) *NX_FLOAT* {units=*NX_UNITLESS*}

Clear aperture of the device (e.g. 90% of diameter for a disc, or 90% of length and height for square geometry).

wavelength_range: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_WAVELENGTH*}

Wavelength range for which the beam splitter is designed. Enter the minimum and maximum values of the wavelength range. Alternatively, or additionally, you may define the wavelength range for the coating in coating/wavelength_range_coating.

optical_loss: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_outputs]) {units=*NX_UNITLESS*}

Optical loss of the beam splitter for the various outputs (i.e. the paths of the beam after being split by the beam splitter). The order of the ratios must be consistent with the labels 1, 2, … N_outputs defined by the sketch in 'SHAPE/sketch', starting with 1.

incident_angle: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Optimized angle of incidence for the desired splitting ratio.

deflection_angle: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of deflection corresponding to the optimized angle of incidence defined in incident_angle.

AOI_**range**: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_ANGLE*}

Range of the angles of incidence (AOI) for which the beam splitter can be operated. Specify the minimum and maximum angles of the range.

reflectance: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Reflectance of the beam splitter at given spectral values.

transmission: (optional) *NX_FLOAT* (Rank: 2, Dimensions: [N_outputs, N_spectrum_RT]) {units=*NX_UNITLESS*}

Transmission at given spectral values for the various outputs (i.e. the paths of the beam after being split by the beam splitter). The order of the ratios must be consistent with the labels 1, 2, … N_outputs defined by the sketch in 'SHAPE/sketch', starting with 1.

SHAPE: (recommended) *NXshape*

Describe the geometry (shape, dimension etc.) of the beam splitter. Specify the dimensions in 'SHAPE/size'. A sketch of the device should be provided in the 'sketch(NXdata)' field to clarify (i) the shape and dimensions of the device, and (ii) the input and outputs (i.e. the direction of the incoming and outcoming (split) beams).

shape: (optional) *NX_CHAR* <=

Describe the shape (plate, cube, wedged, prism etc.).

Any of these values:

- `cube`
- `cylinder`
- `plate`
- `prism`
- `wedged`
- `other`

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 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/polarizing-field*
- */NXbeam_splitter/reflectance-field*
- */NXbeam_splitter/SHAPE-group*
- */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

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:

Fig. 16: A hypothetical capillary furnace. The beam passes from left to right (blue dashes), passing through window 1, then window 2, before passing through the downstream wall of the capillary. It is then scattered by the sample with scattered beams passing through the upstream wall of the capillary, then windows 4 and 5. As part of the corrections for a PDF experiment it is necessary to subtract the PDF of the empty container (i.e. each of the windows and the capillary). To calculate the PDF of the empty container it is necessary to have the measured scattering data and to know the nature (e.g. density, elemental composition, etc.) of the portion of the container which the beam passed through.

This class encodes the position of the container with respect to the sample and allows the calculation of the beampath through the container. It also includes sufficient data to model beam absorption of the container and a link to a dataset containing a measurement of the container with nothing inside, to allow data corrections (at a specific beam energy/measurement time) to be made.

Symbols:

No symbol table

Groups cited:
NXbeam, *NXshape*, *NXtransformations*

Structure:

Fig. 17: 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

NXcsg

Status:

base class (contribution), extends *NXobject*

Description:

Constructive Solid Geometry (CSG) base class.

Offers concepts for combining the definitions of leaf and branching nodes of a CSG tree.

Symbols:

No symbol table

Groups cited:
NXcsg

Structure:

operation: (optional) *NX_CHAR*

One of the standard construction solid geometry set operations, or statement IS_QUADRIC or IS_MESH if the CSG is a pointer to an instance of a geometry class. Takes values:

Any of these values:

- `UNION`
- `INTERSECTION`
- `DIFFERENCE`
- `COMPLEMENT`
- `IS_QUADRIC`
- `IS_MESH`

geometry: (optional) *NX_CHAR*

Path to a field that is an instance of one of several possible geometry classes: Specifically, *NXquadric* if 'operation' is IS_QUADRIC, *NXoff_geometry*, or other primitive based base classes if 'operation' is IS_MESH.

The instance defines the surface making up the constructive solid geometry component. This field is compulsory if 'operation' is IS_QUADRIC or IS_MESH.

a: (optional) *NXcsg*

The first operand of constructive solid geometry operation. Compulsory if 'operation' is UNION, INTERSECTION, DIFFERENCE or COMPLEMENT.

b: (optional) *NXcsg*

The second operand of constructive solid geometry operation. Compulsory if 'operation' is UNION, INTERSECTION or DIFFERENCE.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcsg/a-group*
- */NXcsg/b-group*
- */NXcsg/geometry-field*
- */NXcsg/operation-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcsg.nxdl.xml

NXcxi_ptycho

Status:

application definition (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 to have shape (npts_x*npts_y, frame_size_x, frame_size_y). For nexus this is only true for arbitrary scan paths with raster format scans taking shape (npts_x, npts_y, frame_size_x, frame_size_y).

Symbols:

These symbols will be used below to coordinate the shapes of the datasets.

npts_x: The number of points in the x direction

npts_y: Number of points in the y direction.

frame_size_x: Number of detector pixels in x

frame_size_y: Number of detector pixels in y

Groups cited:
NXbeam, *NXcollection*, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXmonitor*, *NXsample*, *NXsource*, *NXtransformations*

Structure:

entry_1: (required) *NXentry*

title: (optional) *NX_CHAR* <=

start_time: (optional) *NX_DATE_TIME* <=

end_time: (optional) *NX_DATE_TIME* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXcxi_ptycho`

instrument_1: (required) *NXinstrument* <=

source_1: (required) *NXsource* <=

name: (required) *NX_CHAR* <=

energy: (required) *NX_FLOAT* <=

This is the energy of the machine, not the beamline.

probe: (required) *NX_FLOAT*

type: (required) *NX_FLOAT*

beam_1: (required) *NXbeam* <=

energy: (required) *NX_FLOAT*

@units: (required) *NX_CHAR*

extent: (optional) *NX_FLOAT* <=

@units: (required) *NX_CHAR*

incident_beam_divergence: (optional) *NX_FLOAT* <=

@units: (required) *NX_CHAR*

incident_beam_energy: (required) *NX_FLOAT*

@units: (required) *NX_CHAR*

incident_energy_spread: (required) *NX_FLOAT*

@units: (required) *NX_CHAR*

detector_1: (required) *NXdetector* <=

@axes: (required) *NX_CHAR*

should have value "[, data]"

@signal: (required) *NX_CHAR*

should have value "data"

translation: (required) *NX_FLOAT* {units=*NX_LENGTH*}

This is an array of shape (npts_x*npts_y, 3) and can be a Virtual Dataset of x and y

@units: (required) *NX_CHAR*

@axes: (required) *NX_CHAR*

this should take the value "translation:$slowaxisname:$fastaxisname"

@interpretation: (required) *NX_CHAR*

This should be "image"

data: (required) *NX_INT* (Rank: 3 for arbitrary scan, 4 for raster, Dimensions: [npts_x, npts_y, frame_size_x, frame_size_y])

x_pixel_size: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

@units: (required) *NX_CHAR*

y_pixel_size: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

@units: (required) *NX_CHAR*

distance: (required) *NX_FLOAT* {units=*NX_LENGTH*} <=

The distance between the detector and the sample

@units: (required) *NX_CHAR*

beam_center_x: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

@units: (required) *NX_CHAR*

beam_center_y: (optional) *NX_FLOAT* {units=*NX_LENGTH*} <=

@units: (required) *NX_CHAR*

transformations: (required) *NXtransformations* <=

vector: (required) *NX_NUMBER* <=

data_1: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

This data must always have shape (npts_x*npts_y, frame_size_x, frame_size_y) regardless of the scan pattern. Use hdf5 virtual dataset to achieve this.

MONITOR: (optional) *NXmonitor*

data: (required) *NX_FLOAT* (Rank: 1 for arbitrary scan, 2 for raster, Dimensions: [npts_x, npts_y])

DATA: (required) *NXdata* <=

@axes: (required) *NX_CHAR* <=

This should be "[x,.]" for arbitrary scanning patterns, and "[x,.,.]" for raster

@signal: (required) *NX_CHAR* <=

This should be "data"

x_indices: (required) *NX_CHAR*

y_indices: (required) *NX_CHAR*

data: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

x: *link* (suggested target: `/NXentry/NXsample/NXtransformations/x`)

y: *link* (suggested target: `/NXentry/NXsample/NXtransformations/y`)

data_1: (required) *NXcollection* <=

To ensure CXI compatibility the data in this group must always have shape that is (npts_x*npts_y, frame_size_x, frame_size_y). For nexus-style raster scans it is proposed that hdf5 virtual dataset is used. **data**: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/data`)

translation: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/translation`)

sample_1: (required) *NXsample*

name: (optional) *NX_CHAR* <=

transformations: (required) *NXtransformations* <=

This must contain two fields with the x and y motors that are linked via the dependency tree according to the real-life motor layout dependency. For raster scans x and y will have shape (npts_x, npts_y) For arbitrary scans x and y will be (npts_x*npts_y,) An attribute with the units for each motor is required.

@vector: (required) *NX_NUMBER*

geometry_1: (required) *NXcollection* <=

translation: *link* (suggested target: `/NXentry/NXinstrument/NXdetector/translation`)

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXcxi_ptycho/DATA-group*
- */NXcxi_ptycho/DATA/data-link*
- */NXcxi_ptycho/DATA/x-link*
- */NXcxi_ptycho/DATA/x_indices-field*
- */NXcxi_ptycho/DATA/y-link*
- */NXcxi_ptycho/DATA/y_indices-field*
- */NXcxi_ptycho/DATA@axes-attribute*
- */NXcxi_ptycho/DATA@signal-attribute*
- */NXcxi_ptycho/data_1-group*
- */NXcxi_ptycho/data_1/data-link*
- */NXcxi_ptycho/data_1/translation-link*
- */NXcxi_ptycho/entry_1-group*
- */NXcxi_ptycho/entry_1/definition-field*
- */NXcxi_ptycho/entry_1/end_time-field*
- */NXcxi_ptycho/entry_1/instrument_1-group*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1-group*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/energy-field*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/energy@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/extent-field*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/extent@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_divergence-field*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_divergence@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_energy-field*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_beam_energy@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_energy_spread-field*
- */NXcxi_ptycho/entry_1/instrument_1/beam_1/incident_energy_spread@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1-group*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_x-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_x@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_y-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/beam_center_y@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/data-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/data_1-link*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/distance-field*

- */NXcxi_ptycho/entry_1/instrument_1/detector_1/distance@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/transformations-group*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/transformations/vector-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation@axes-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation@interpretation-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/translation@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/x_pixel_size-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/x_pixel_size@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/y_pixel_size-field*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1/y_pixel_size@units-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1@axes-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/detector_1@signal-attribute*
- */NXcxi_ptycho/entry_1/instrument_1/MONITOR-group*
- */NXcxi_ptycho/entry_1/instrument_1/MONITOR/data-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1-group*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/energy-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/name-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/probe-field*
- */NXcxi_ptycho/entry_1/instrument_1/source_1/type-field*
- */NXcxi_ptycho/entry_1/start_time-field*
- */NXcxi_ptycho/entry_1/title-field*
- */NXcxi_ptycho/sample_1-group*
- */NXcxi_ptycho/sample_1/geometry_1-group*
- */NXcxi_ptycho/sample_1/geometry_1/translation-link*
- */NXcxi_ptycho/sample_1/name-field*
- */NXcxi_ptycho/sample_1/transformations-group*
- */NXcxi_ptycho/sample_1/transformations@vector-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXcxi_ptycho.nxdl.xml

NXdelocalization

Status:

base class (contribution), extends *NXobject*

Description:

Base class of the configuration and results of a delocalization algorithm.

Delocalization is used to distribute point-like objects on a grid to obtain e.g. smoother count, composition, or concentration values of 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_nuclides: Number of isotopes in the whitelist.

Groups cited:
NXcg_grid, *NXmatch_filter*

Structure:

grid: (optional) *NXcg_grid*

Details about the grid on which the delocalization is applied.

weighting_model: (optional) *NXmatch_filter*

The weighting model specifies how mark data are mapped to a weight per point/object.

weighting_method: (optional) *NX_CHAR*

As an example from the research field of atom probe points/objects are (molecular) ions. Different methods are used for weighting ions:

- **default, points get all the same weight 1., which for atom probe is equivalent** to (molecular) iontype-based delocalization.
- element, points get as much weight as they have atoms representing a nuclide with a proton number that is matching to a respective entry in whitelist. In atom probe jargon, this means atomic_decomposition.
- isotope, points get as much weight as they have atoms representing a nuclides from a respective entry in whitelist. In atom probe jargon, this means isotope_decomposition.

Any of these values: `default` | `element` | `isotope`

method: (optional) *NX_CHAR* <=

Obligatory value: `whitelist`

match: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_nuclides]) {units=*NX_UNITLESS*}

A list of nuclides based on which to evaluate the weight. Nuclides need to exist in the nuclide table. Values are nuclide (isotope) hash values using the hashing rule defined in *NXatom*.

mark: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_p, n_m]) {units=*NX_UNITLESS*}

Attribute data for each member of the point cloud. For APM these are the iontypes generated via ranging. The number of mark data per point is 1 in the case for atom probe.

weight: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_p]) {units=*NX_UNITLESS*}

Weighting factor with which the integrated intensity per grid cell is multiplied specifically for each point/object. For APM the weight are positive integer values, specifically the multiplicity of the ion, according to the details of the weighting_method.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXdelocalization/grid-group*
- */NXdelocalization/weighting_model-group*
- */NXdelocalization/weighting_model/mark-field*
- */NXdelocalization/weighting_model/match-field*
- */NXdelocalization/weighting_model/method-field*
- */NXdelocalization/weighting_model/weight-field*
- */NXdelocalization/weighting_model/weighting_method-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:

An application definition for describing a dispersive material.

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* <=

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

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 or a custom value (if you use a custom value, also set @custom=True): `gas` | `liquid` | `solid`

material_phase_comment: (optional) *NX_CHAR*

Additional information about the phase if the material phase is not from the enumerated list.

additional_phase_information: (recommended) *NX_CHAR*

This field may be used to denote additional phase information, such as crystalline 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/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_REPEATED_PARAMETER-group
- /NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/name-field
- /NXdispersive_material/ENTRY/dispersion_y/DISPERSION_FUNCTION/DISPERSION_REPEATED_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_REPEATED_PARAMETER-group
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/name-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_REPEATED_PARAMETER/values-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER-group
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/name-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/DISPERSION_SINGLE_PARAMETER/value-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/energy_identifier-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/energy_unit-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/formula-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/model_name-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/representation-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/wavelength_identifier-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_FUNCTION/wavelength_unit-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE-group
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/convention-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/dielectric_function-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/model_name-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/refractive_index-field
- /NXdispersive_material/ENTRY/dispersion_z/DISPERSION_TABLE/wavelength-field
- /NXdispersive_material/ENTRY/dispersion_z/model_name-field

- */NXdispersive_material/ENTRY/dispersion_z/plot-group*
- */NXdispersive_material/ENTRY/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*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXdispersive_material.nxdl.xml

NXelectrostatic_kicker

Status:

base class (contribution), extends *NXcomponent*

Description:

Base class for an 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

NXem_calorimetry

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for minimal example in-situ calorimetry.

TODO:

- What is the technique about.
- General context.
- Literature references.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_p: Number of diffraction pattern.

n_f: Number of radial integration bins.

n_i: Number of coordinates along i axis.

n_j: Number of coordinates along j axis.

Groups cited:
NXcite, *NXcollection*, *NXcoordinate_system*, *NXcs_profiling*, *NXdata*, *NXentry*, *NXnote*, *NXprocess*, *NXprogram*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXem_calorimetry`

identifier_analysis: (optional) *NX_CHAR* <=

profiling: (optional) *NXcs_profiling*

Details about performance, profiling, etc.

start_time: (recommended) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

total_elapsed_time: (required) *NX_NUMBER* <=

program1: (recommended) *NXprogram*

Name of the program whereby this config file was created.

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

environment: (recommended) *NXcollection* <=

Programs and libraries representing the computational environment

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

userID: (optional) *NXuser* <=

sample: (recommended) *NXsample* <=

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).

atom_types: (required) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the specimen. If the specimen 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.

citeID: (optional) *NXcite*

diffraction_space: (optional) *NXcoordinate_system*

diffraction: (required) *NXnote* <=

Reference to the resource which stores acquired pattern from the experiment or simulation that are analyzed in this workflow.

Can refer to the original EMD or MRC files or the parsed NXem in RDM e.g. NOMAD OASIS.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

actuator: (required) *NXnote* <=

Reference to the resource which stores actuator log file from the experiment.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

config: (required) *NXnote* <=

Configuration file that was used for parametrizing this analysis workflow.

file_name: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

synchronization: (required) *NXprocess* <=

Assumptions and computations whereby timestamping data from the detector and actuator (e.g. heating chip) were synchronized.

sequence_index: (required) *NX_POSINT* <=

start_time: (required) *NX_DATE_TIME*

ISO8601 with local time zone reference timestamp that tells with which delta_time can be converted in timestamp. The reference timestamp is defined as the time when the actuator started acting on the sample.

Time differences to this timestamp when correlated signals such as diffraction pattern matching with a specific state of the sample (e.g. obtained temperature via the actuator) are reported through delta_time.

indices_pattern: (required) *NX_INT* (Rank: 1, Dimensions: [n_p]) {units=*NX_UNITLESS*}

delta_time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_p]) {units=*NX_TIME*}

Time difference to start_time.

Collecting diffraction pattern also takes some time. It is assumed that the acquisition time for each pattern is substantial shorter than the time it takes the actuator to cause a change in stimulus (e.g. temperature).

pattern_center: (required) *NXprocess* <=

Computation of the center for each pattern using e.g. a Circular Hough Transformation.

sequence_index: (required) *NX_POSINT* <=

position: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_p, 2]) {units=*NX_LENGTH*}

Computed center for each pattern.

distortion_correction: (optional) *NXprocess* <=

Elliptical distortion correction as a step when computing the center for patterns.

sequence_index: (required) *NX_POSINT* <=

center: (required) *NX_NUMBER* (Rank: 2, Dimensions: [n_p, 2]) {units=*NX_LENGTH*}

Computed center for each pattern.

integration: (required) *NXprocess* <=

Integrated diffraction pattern intensity as a function of radial distance from the center azimuthally integrated as a function of time.

sequence_index: (required) *NX_POSINT* <=

resultBACKGROUND: (optional) *NXdata* <=

The integrated intensities:

- result_with_background
- result_without_background

@**signal**: (required) *NX_CHAR* <=

@**axes**: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (required) *NX_CHAR* <=

intensity: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_p, n_f]) {units=*NX_UNITLESS*}

Integrated intensity as a function of time and the radial distance from the pattern center.

@**long_name**: (required) *NX_CHAR*

indices_pattern: (optional) *NX_INT* (Rank: 1, Dimensions: [n_p]) {units=*NX_UNITLESS*}

Identifier for each pattern.

@**long_name**: (required) *NX_CHAR*

s: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_f]) {units=*NX_ANY*}

Positions in reciprocal space.

@**long_name**: (required) *NX_CHAR*

time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_p]) {units=*NX_TIME*}

Time since start of the in-situ experiment

background_subtraction: (optional) *NXprocess* <=

sequence_index: (required) *NX_POSINT* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXem_calorimetry/ENTRY-group*
- */NXem_calorimetry/ENTRY/actuator-group*
- */NXem_calorimetry/ENTRY/actuator/algorithm-field*
- */NXem_calorimetry/ENTRY/actuator/checksum-field*
- */NXem_calorimetry/ENTRY/actuator/file_name-field*
- */NXem_calorimetry/ENTRY/background_subtraction-group*
- */NXem_calorimetry/ENTRY/background_subtraction/sequence_index-field*
- */NXem_calorimetry/ENTRY/citeID-group*
- */NXem_calorimetry/ENTRY/config-group*
- */NXem_calorimetry/ENTRY/config/algorithm-field*
- */NXem_calorimetry/ENTRY/config/checksum-field*
- */NXem_calorimetry/ENTRY/config/file_name-field*
- */NXem_calorimetry/ENTRY/definition-field*
- */NXem_calorimetry/ENTRY/diffraction-group*
- */NXem_calorimetry/ENTRY/diffraction/algorithm-field*
- */NXem_calorimetry/ENTRY/diffraction/checksum-field*
- */NXem_calorimetry/ENTRY/diffraction/file_name-field*
- */NXem_calorimetry/ENTRY/diffraction_space-group*
- */NXem_calorimetry/ENTRY/distortion_correction-group*
- */NXem_calorimetry/ENTRY/distortion_correction/center-field*
- */NXem_calorimetry/ENTRY/distortion_correction/sequence_index-field*
- */NXem_calorimetry/ENTRY/environment-group*
- */NXem_calorimetry/ENTRY/environment/programID-group*
- */NXem_calorimetry/ENTRY/environment/programID/program-field*
- */NXem_calorimetry/ENTRY/environment/programID/program@version-attribute*
- */NXem_calorimetry/ENTRY/identifier_analysis-field*
- */NXem_calorimetry/ENTRY/integration-group*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND-group*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/indices_pattern-field*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/indices_pattern@long_name-attribute*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/intensity-field*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/intensity@long_name-attribute*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/s-field*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/s@long_name-attribute*

- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/time-field*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND/title-field*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND@axes-attribute*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND@AXISNAME_indices-attribute*
- */NXem_calorimetry/ENTRY/integration/resultBACKGROUND@signal-attribute*
- */NXem_calorimetry/ENTRY/integration/sequence_index-field*
- */NXem_calorimetry/ENTRY/pattern_center-group*
- */NXem_calorimetry/ENTRY/pattern_center/position-field*
- */NXem_calorimetry/ENTRY/pattern_center/sequence_index-field*
- */NXem_calorimetry/ENTRY/profiling-group*
- */NXem_calorimetry/ENTRY/profiling/end_time-field*
- */NXem_calorimetry/ENTRY/profiling/start_time-field*
- */NXem_calorimetry/ENTRY/profiling/total_elapsed_time-field*
- */NXem_calorimetry/ENTRY/program1-group*
- */NXem_calorimetry/ENTRY/program1/program-field*
- */NXem_calorimetry/ENTRY/program1/program@version-attribute*
- */NXem_calorimetry/ENTRY/sample-group*
- */NXem_calorimetry/ENTRY/sample/atom_types-field*
- */NXem_calorimetry/ENTRY/sample/is_simulation-field*
- */NXem_calorimetry/ENTRY/synchronization-group*
- */NXem_calorimetry/ENTRY/synchronization/delta_time-field*
- */NXem_calorimetry/ENTRY/synchronization/indices_pattern-field*
- */NXem_calorimetry/ENTRY/synchronization/sequence_index-field*
- */NXem_calorimetry/ENTRY/synchronization/start_time-field*
- */NXem_calorimetry/ENTRY/userID-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXem_calorimetry.nxdl.xml

NXisocontour

Status:

base class (contribution), extends *NXobject*

Description:

Base class for describing isocontouring/phase-fields in Euclidean space.

Iso-contouring algorithms such as Marching Cubes and others are frequently used to segment d-dimensional regions at crossings of a threshold value, the so-called isovalue.

In Computational Materials Science phase-field methods are frequently used. Phase-field variables are discretized frequently using regular grids.

Isocontour algorithms are often used in such context to pinpoint the locations of microstructural features from this implicit phase-field- variable-value-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 (and other) simulation data can take advantage of NeXus base class to improve interoperability.

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*}

The dimensionality of the space in which the isocontour is embedded.

Any of these values: `1 | 2 | 3`

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*, *NXsample*, *NXsensor*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Obligatory value: `NXiv_temp`

SAMPLE: (recommended) *NXsample* <=

name: (required) *NX_CHAR* <=

Descriptive name or ideally (globally) unique persistent identifier.

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.

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*
- */NXiv_temp/ENTRY/SAMPLE-group*
- */NXiv_temp/ENTRY/SAMPLE/atom_types-field*
- */NXiv_temp/ENTRY/SAMPLE/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXiv_temp.nxdl.xml

NXlockin

Status:

base class (contribution), extends *NXobject*

Description:

A base class definition for a lock-in amplifier.

The lock-in amplifier information: the device is being used to extract a (potentially) very weak input signal buried in the noisy background, where the input signal has the same frequency (or its harmonic) as carrier signal or reference signal, using heterodyne detection.

This method extracts the amplitude and phase shift between input signal and reference signal.

In single phase lock-in amplifiers used in high signal-to-noise ratio applications, only the amplitude is measured and phase difference is set to zero.

In two phase lock-in amplifiers used in low signal-to-noise ratio applications, both the amplitude and phase difference are measured.

Fields with partial names like low_passN, high_passN, etc., can be repeated for each channel of the lockin amplifier. It is envisioned that these fields are named low_pass_0, low_pass_1, and so on.

Symbols:

No symbol table

Groups cited:
NXfabrication

Structure:

bias_divider: (optional) *NX_CHAR*

Bias divider for lock-in channel. math: *Bias divider $= frac{V_{mathrm{ref}}}{V_{mathrm{ref}} + V_{mathrm{input}}}$*

modulation_status: (optional) *NX_BOOLEAN*

Switch the lock-in modulation on or off.

modulation_signal: (optional) *NX_CHAR*

Type of the modulation or reference signal, voltage | current | bias. The name of the current or voltage signal can be also specified according to their purpose (e.g., *bias* voltage).

Any of these values or a custom value (if you use a custom value, also set @custom=True): `voltage` | `current` | `bias`

modulation_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The frequency of the sine modulation that is used to modulate the signal in lock-in.

reference_amplitude: (optional) *NX_NUMBER* {units=*NX_ANY*}

Amplitude of the reference signal for the lock-in amplifier. Unit could be NX_VOLTAGE or NX_CURRENT depending on the type of the reference signal.

reference_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

Frequency of the reference signal for the lock-in amplifier.

reference_phase: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Phase of the reference signal set in the lock-in amplifier.

phase_difference: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Phase difference between the input signal and the reference signal. This is used in two phase lock-in amplifiers. In single phase lock-in amplifiers, this value is set to zero.

demodulated_signal: (optional) *NX_CHAR*

Type of the demodulated signal, current | voltage | bias. The name of the current or voltage signal can be also specified according to their purpose (e.g., *bias* voltage).

Any of these values or a custom value (if you use a custom value, also set @custom=True): `current` | `voltage` | `bias`

demodulated_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The frequency of the demodulated signal.

frequency_modulation_bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The bandwidth of the modulating signal.

phase_modulation_bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The bandwidth of the modulating signal over which modulated signal spreads.

amplitude_modulation_bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The bandwidth of the modulating signal over which modulated signal spreads.

demodulated_amplitude: (optional) *NX_NUMBER* {units=*NX_ANY*}

The amplitude of the demodulated signal. Unit could be NX_VOLTAGE or NX_CURRENT depending on the type of the demodulated signal.

demodulated_phase: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The phase of the demodulated signal.

demodulator_channels: (optional) *NX_CHAR*

Comma separated list of the demodulator channels.

low_passN: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

Frequency of the low-pass filter or cut-off frequency. Only signals below this frequency are passed through the filter. N is envisioned to represent the channel number e.g., low_pass1.

high_passN: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

Frequency of the high-pass filter or cut-off frequency. Only signals above this frequency are passed through the filter. N is envisioned to represent the channel number e.g., high_pass1.

lp_filter_orderN: (optional) *NX_NUMBER*

Order of the low-pass filter applied on the input or demodulated signals (X, Y). Reducing the bandwidth or increasing the order reduces noise on the demodulated signals, but increases settling and measurement times. N is envisioned to represent the channel number e.g., lp_filter_order1.

hp_filter_orderN: (optional) *NX_NUMBER*

Order of the high-pass filter applied on the input or demodulated signal. This is used mainly to suppress a DC component of the input signal noise. N is envisioned to represent the channel number e.g., hp_filter_order1.

ref_offset_phaseN: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

An extra phase offset added to the reference signal in modulation step. N is envisioned to represent the channel number e.g., hp_filter_order1.

harmonic_orderN: (optional) *NX_NUMBER*

The reference signal can be a higher harmonic of the modulation signal. Here the order of the harmonic is stored. N is envisioned to represent the channel number e.g., harmonic_order1.

sensitivity_factor: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Ratio of output signal amplitude to input signal amplitude (V/V).

dc_offset_valueN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The DC offset of the demodulated signal. This is used to remove the DC component from the demodulated signal.

The same DC offset might be applied to the input signal in the modulation process. N is envisioned to represent the channel number e.g., dc_offset_value1.

hardware: (optional) *NXfabrication*

Hardware manufacturers and type (product number) of lock-in amplifier.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXlockin/amplitude_modulation_bandwidth-field*
- */NXlockin/bias_divider-field*
- */NXlockin/dc_offset_valueN-field*
- */NXlockin/demodulated_amplitude-field*
- */NXlockin/demodulated_frequency-field*
- */NXlockin/demodulated_phase-field*
- */NXlockin/demodulated_signal-field*
- */NXlockin/demodulator_channels-field*
- */NXlockin/frequency_modulation_bandwidth-field*
- */NXlockin/hardware-group*
- */NXlockin/harmonic_orderN-field*
- */NXlockin/high_passN-field*
- */NXlockin/hp_filter_orderN-field*
- */NXlockin/low_passN-field*
- */NXlockin/lp_filter_orderN-field*
- */NXlockin/modulation_frequency-field*
- */NXlockin/modulation_signal-field*
- */NXlockin/modulation_status-field*
- */NXlockin/phase_difference-field*
- */NXlockin/phase_modulation_bandwidth-field*
- */NXlockin/ref_offset_phaseN-field*
- */NXlockin/reference_amplitude-field*
- */NXlockin/reference_frequency-field*
- */NXlockin/reference_phase-field*
- */NXlockin/sensitivity_factor-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXlockin.nxdl.xml

NXmagnetic_kicker

Status:

base class (contribution), extends *NXcomponent*

Description:

Base class 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 *NXparameters*

Description:

Base class of a filter to select members of a set based on their identifier.

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*

Definition of the logic what the filter yields:

- Whitelist specifies which entries with said value to include. Entries with all other values will be excluded.
- 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. If the match e.g. specifies [1, 5, 6] and method is set to whitelist, only entries with values matching 1, 5 or 6 will be processed. All other entries will be excluded.

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

NXmicrostructure

Status:

base class (contribution), extends *NXobject*

Description:

Base class to describe a microstructure, its structural aspects, associated descriptors, properties.

Whether one uses a continuum or atomic scale description of materials, these are always a model only of the true internal structure of a material. Such models are useful as they enable a coarse-graining and categorizing of properties and representational aspects from which measured or simulated descriptions can be correlated with properties, i.e. descriptor values. The base class here can be used to describe the structural aspect of a region-of-interest for a specimen that was investigated or a computer simulation that was performed for a virtual specimen.

Specimens experience thermo-chemo-mechanical processing (steps) before characterization. Therefore, the characterized microstructure may not turn out to be the same structure as of the untreated sample from which the region-of-interests on the specimen were sampled.

Fields such as time and increment enable a quantification of the spatiotemporal evolution of a materials' structure by using multiple instances of NXmicrostructure. Both measurements and simulation virtually always sample this evolution. Most microscopy techniques characterize only a two-dimensional representation (projection) of the characterized material volume. Often materials are characterized only for specific states or via sampling coarsely in time relative to the timescale at which the physical phenomena take place. This is typically a compromise between the research questions and technical surplus practical limitations.

The term microstructural feature covers here crystals and all sorts of crystal defects within the material (interfaces, triple junctions, dislocations, pores, etc.). A key challenge with the description of representations and properties of such microstructural features is that they can be represented and view as features with different dimensionality. Furthermore, combinations of features of different dimensionality are frequently expected to be documented with intuitive naming conventions when flat property lists are used. For these key-value dictionaries often folksonomies are used. These can be based on ad hoc documentation of such dictionaries in the literature and the metadata section of public data repositories.

NXmicrostructure is an attempt to standardize these descriptions stronger.

For crystals the number of typically used technical terms are smaller than for interfaces or line like defects and junctions of different types of crystal defects. The term grain describes a contiguous region of material that is delineated by interfaces (phase or grain boundaries). With its origin motivated by light optical microscopy though a grain is not necessarily a single crystal but can have an internal structure of defect such as dislocations. In this base class we use the term and respective group crystals though for single crystals and grains. The reason why this is possible is that when e.g. materials engineers talk about grains they inherently assume that the internal structure of these grains can be described with homogenized effective properties. If alternatively the individual structural crystalline or features of this grain should be distinguished it is useful to instantiate these as individual instances of crystals.

Grain boundaries and phase boundaries are two main categories of interfaces. A grain boundary delineates two regions with similar crystal structure and phase but different orientation. A grain boundary is thus a homophase interface. By contrast, a heterophase boundary delineates two regions with typically but not necessarily dissimilar crystal structure but a different atomic occupation that justifies to distinguish two phases. There is a substantial variety of interfaces whose distinction was classically based on geometrical arguments but considers that atomic segregation is an equally important structural aspect to consider when classifying grain boundaries. A concise overview on theoretical aspect of and the semantics for characterizing interfaces and their properties is provided in e.g. W. Bollmann and A. Sutton and R. W. Baluffi, Interfaces in Crystalline Materials, Clarendon Press, ISBN 9780198500612.

Also for junctions between crystal defects there is a considerable variety of terms. Junctions are features in three-dimensional Euclidean space even if they are formed maybe only through a monolayer or a pearl chain of atoms. Either way their local atomic and electronic environment is different compared to the situation of an ideal crystal, or the adjoining defects, which gives typically rise to a plethora of configurations of which some yield useful material properties or affect material properties.

Like crystals and interfaces, junctions are assumed to represent groups of atoms that have specific descriptor values which are different to other features. Taking an example, a triple junction is practically a three-dimensional defect as its atoms are arranged in three-dimensional space but the characteristics of that defect can often be reduced to a lower-dimensional description such as a triple line or a triple point as the projection of a line. Therefore, different representations can be used to describe the location, shape, and structure of such defect.

This base class provides definitions for crystals, grains, interfaces, triple junctions, and quadruple junctions thus covering, volumetric, patch, line, and point like features that can serve as examples for future extension.

As different types of crystal defects can interact, there is a substantial number of in principle characterizable and representable objects. Take again a triple line as an example. It is a tubular feature built from three adjoining interfaces. However, dislocations as line defects can interact with triple lines. Therefore, one can also argue that along a triple line there exist dislocation-line- triple-line junctions, likewise dislocations form own junctions.

The description took inspiration from E. E. Underwood and E. E. Underwood's book on Quantitative Stereology published in 1970 to categorize features based on their dimensionality.

Indices can be defined either implicitly or explicitly. Indices for implicit indexing are defined on the interval $[index_offset, index_offset + cardinality - 1]$. Indices can be used as identifiers for distinguishing instances, i.e. indices are equivalent to instance names of individual crystals.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_c: The number of crystals or their projections

n_i: The number of interfaces or their projections

n_tj: The number of triple junctions or their projections

n_qj: The number of quadruple junctions or their projections

n_c_one: The number of one-dimensional crystal projections

n_i_one: The number of one-dimensional interface projections

n_c_two: The number of two-dimensional crystal projections

n_i_two: The number of two-dimensional interface projections

n_tj_two: The number of two-dimensional triple line projections

n_ld_two: The number of two-dimensional line defect projections

n_c_three: The number of crystals (grain and sub-grain are exact synonyms for crystal).

n_i_three: The number of interfaces (grain boundary and phase boundary are subclasses of interfaces).

n_tj_three: The number of triple junctions (triple line is a exact synonym for triple junction, triple point is projection of a triple junction).

n_qj_three: The number of quadruple junctions.

d: The dimensionality of the representation that needs to match the value for configuration/dimensionality

Groups cited:

NXcg_grid, *NXcg_point*, *NXcg_polygon*, *NXcg_polyhedron*, *NXcg_polyline*, *NXcg_triangle*, *NXchemical_composition*, *NXmicrostructure_feature*, *NXphase*, *NXprocess*, *NXprogram*, *NXrotations*

Structure:

comment: (optional) *NX_CHAR*

Discouraged free-text field for leaving comments

date: (optional) *NX_DATE_TIME*

ISO8601 with offset to local time zone included when a timestamp is required.

time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Measured or simulated physical time stamp for this microstructure snapshot. Not to be confused with wall-clock timing or profiling data.

iteration: (optional) *NX_INT* {units=*NX_UNITLESS*}

Iteration or increment counter.

configuration: (optional) *NXprocess*

Group where to store details about the configuration and parameterization of algorithms used whereby microstructural features were identified.

dimensionality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Dimensionality of Euclidean space in which the analysis is performed.

This field can be used e.g. by a research data management system to identify if the description specifies one-, two-, or three-dimensional microstructural representations.

Any of these values: `1 | 2 | 3`

algorithm: (optional) *NX_CHAR*

Algorithm whereby interfaces between crystals were reconstructed.

- Disorientation clustering groups nearby material points based on their crystallographic disorientation
- Fast multiscale clustering based on D. Kushnir et al.
- Markov chain clustering F. Niessen et al.

Any of these values:

- `unknown`
- `disorientation_clustering`
- `fast_multiscale_clustering`
- `markov_chain_clustering`

disorientation_threshold: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Threshold to define at which disorientation angle to assume two crystalline regions have a significant orientation difference that warrants to assume that there exists an interface between the two regions.

PROGRAM: (optional) *NXprogram*

The program with which the microstructure was reconstructed.

CG_GRID: (optional) *NXcg_grid*

CG_POINT: (optional) *NXcg_point*

CG_POLYLINE: (optional) *NXcg_polyline*

CG_TRIANGLE: (optional) *NXcg_triangle*

CG_POLYGON: (optional) *NXcg_polygon*

CG_POLYHEDRON: (optional) *NXcg_polyhedron*

chemical_composition: (optional) *NXchemical_composition*

The chemical composition of this microstructure (region).

phases: (optional) *NXmicrostructure_feature*

Different (thermodynamic) phases can be distinguished for the region-of- interest.

index_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

First identifier whereby to identify phases implicitly.

PHASE: (optional) *NXphase*

crystals: (optional) *NXmicrostructure_feature*

One- or two-dimensional projections, or three-dimensional representations of crystals.

An example for a volume bounded by other crystal defects. Crystals can be grains of different phases, precipitates, dispersoids; there are many terms used specifically in the materials engineering community.

Typically, crystals are measured on the surface of a sample via optical or electron microscopy. Using X-ray diffraction methods crystals can be observed in bulk specimens.

Crystals are represented by a set of pixel, voxel, or polygons and their polyline boundaries. In rare cases the volume bounded gets represented using constructive solid geometry approaches.

representation: (optional) *NX_CHAR*

Reference to an instance of:

- *NXcg_polyline* for a one- or two-dimensional representation as only a projection is available (like in linear intercept analysis)
- *NXcg_polygon*, *NXcg_triangle*, or *NXcg_polyhedron* for a two- or three-dimensional representation as only a projection is available (like in most experiments)
- *NXcg_grid* for regularly pixelated (in 1D, 2D) or voxelated representations (in 3D)

which represent the geometrical entities of the discretization.

number_of_crystals: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many crystals are distinguished.

Crystals are listed irrespective of the phase to which these are assigned.

number_of_phases: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many phases are distinguished.

Phases are typically distinguished based on statistical thermodynamics argument and crystal structure.

index_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

First identifier whereby to identify crystals implicitly.

indices_crystal: (optional) *NX_INT* (Rank: 1, Dimensions: [n_c]) {units=*NX_UNITLESS*}

Identifier whereby to identify each crystal explicitly.

indices_phase: (optional) *NX_INT* (Rank: 1, Dimensions: [n_c]) {units=*NX_UNITLESS*}

Identifier whereby to identify phase for each crystal explicitly.

boundary_contact: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_c])

True, if the feature makes contact with the edge of the ROI. False, if the feature does not make contact with the edge of the ROI.

orientation_spread: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_c]) {units=*NX_ANGLE*}

Average disorientation angle for each crystal between individual orientations of that crystal evaluated as a summary statistic for all probed positions vs the average disorientation of that crystal.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_c]) {units=*NX_LENGTH*}

Length of each crystal

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_c]) {units=*NX_AREA*}

Area of each crystal.

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_c]) {units=*NX_VOLUME*}

Volume of each crystal

orientation: (optional) *NXrotations*

Possibility to store the mean orientation of the grain.

interfaces: (optional) *NXmicrostructure_feature*

One- or two-dimensional projections or three-dimensional representation of interfaces between crystals as topological entities equivalent to dual_junctions.

An example for a surface defect. Most important are interfaces such as grain and phase boundaries but factually interfaces also exist between the environment and crystals exposed at the surface of the specimen or internal surfaces like between crystals, cracks, or pores.

Interfaces are typically reported as discretized features. For interface projections on the 2D plane these are most frequently polyline segments. For interface patches in 3D these are most frequently triangulations. Descriptions with continuous functions are seldom used unless simplified configurations are studied in modeling and theoretical studies.

When using discretizations the individual interface segments need to be distinguished from the interfaces themselves. Consequently, there are two sets of indices.

representation: (optional) *NX_CHAR*

Reference to an instance of:

- *NXcg_point* for a one-dimensional representation as only a projection is available (as in linear intercept analyses)
- *NXcg_polyline* or *NXcg_polygon* for a two-dimensional representation as only a projection is available (like in most experiments)

- *NXcg_grid* for regularly pixelated (in 1D, 2D) or voxelated representations (in 3D) using (boolean) masks (like in computer simulations or 3D experiments)

which represent the geometrical entities of the discretization.

number_of_interfaces: (optional) *NX_UINT* {units=*NX_UNITLESS*}

How many interfaces are distinguished.

index_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

First identifier whereby to identify interfaces implicitly.

indices_interface: (optional) *NX_INT* (Rank: 1, Dimensions: [n_i]) {units=*NX_UNITLESS*}

Identifier whereby to identify each interface explicitly.

An array with as many entries as interfaces or their projections.

indices_crystal: (optional) *NX_INT* (Rank: 2, Dimensions: [n_i, 2]) {units=*NX_UNITLESS*}

Set of pairs of indices_crystal values, for each interface one value pair.

An array with as many pairs as interfaces or their projections.

@use_these: (optional) *NX_CHAR*

The specific identifiers whereby to resolve ambiguities.

indices_phase: (optional) *NX_INT* (Rank: 2, Dimensions: [n_i, 2]) {units=*NX_UNITLESS*}

Set of pairs of indices_phase values, for each interface one value pair.

An array with as many pairs as interfaces or their projections.

@use_these: (optional) *NX_CHAR*

The specific identifiers whereby to resolve ambiguities.

number_of_triple_junctions: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_i]) {units=*NX_UNITLESS*}

Interfaces can be the physical three-dimensional surfaces or two- or one-dimensional projections. The latter situation applies typically for characterization with electron microscopy.

In the case of a two-dimensional projection interfaces are interface traces. These have two terminating junctions. In three dimensions though the interface is a surface patch that is bounded by multiple triple lines.

Number of triple_junctions adjoining each interface. This array resolves the number of values along the second dimension for the field indices_triple_junctions.

indices_triple_junction: (optional) *NX_INT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Set of pairs of indices_triple_junction for each interface.

An array with as many tuples of pairs to describe all junctions about all interfaces.

@use_these: (optional) *NX_CHAR*

The specific identifiers whereby to resolve ambiguities.

boundary_contact: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_i])

True, if the interface makes contact with the edge of the ROI. False, if the interface does not make contact with the edge of the ROI.

surface_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Gibbs free surface energy for each interface.

mobility: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_ANY*}

Non-intrinsic mobility of each interface.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_LENGTH*}

The length of each interface if only projections are available.

This is not necessarily the same as the length of the individual polyline segments whereby the interface is discretized.

area: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_i]) {units=*NX_AREA*}

The surface area of all interfaces.

triple_junctions: (optional) *NXmicrostructure_feature*

Projections or representations of junctions at which three interfaces meet.

An example for a line defect. Triple junctions are characterized as triple lines or triple points as their projections, or junctions observed between crystals (at the specimen surface exposed to an environment) (including wetting phenomena) or inside the specimen (crack, pores).

representation: (optional) *NX_CHAR*

Reference to an instance of:

- *NXcg_point* for a one-dimensional representation as only a projection is available (like in most experiments)
- *NXcg_polyline* for a two-dimensional representation as only a projection is available
- *NXcg_polygon* for a two-dimensional representation in the (seldom) case of sufficient spatial resolution and the line in the projection plane or cases where triple junction locations are approximated e.g. using a set of triangles
- *NXcg_polyhedron* for a three-dimensional representation via e.g. a representation of Voronoi cells about atoms
- *NXcg_grid* for regularly pixelated or voxelated representation in one, two, or three dimensions using (boolean) masks

which represent the geometrical entities of the discretization.

number_of_junctions: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of triple junctions.

index_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

First identifier to identify triple junctions implicitly.

indices_triple_junction: (optional) *NX_INT* (Rank: 1, Dimensions: [n_tj]) {units=*NX_UNITLESS*}

Identifier to identify each triple junction explicitly.

location: (optional) *NX_INT* (Rank: 1, Dimensions: [n_tj]) {units=*NX_UNITLESS*}

Set of identifier for positions whereby to identify the location of each junction.

@use_these: (optional) *NX_CHAR*

The specific identifiers whereby to resolve ambiguities.

position: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_tj, d]) {units=*NX_LENGTH*}

Explicit positions.

indices_crystal: (optional) *NX_INT* (Rank: 2, Dimensions: [n_tj, 3]) {units=*NX_UNITLESS*}

Set of tuples of identifier of crystals connected to the junction for each triple junction.

indices_interface: (optional) *NX_INT* (Rank: 2, Dimensions: [n_tj, 3]) {units=*NX_UNITLESS*}

Set of tuples of identifier of interfaces connected to the junction for each triple junction.

@use_these: (optional) *NX_CHAR*

The specific interface identifiers whereby to resolve ambiguities.

indices_polyline: (optional) *NX_INT* (Rank: 2, Dimensions: [n_tj, 3]) {units=*NX_UNITLESS*}

Set of tuples of identifier for polyline segments connected to the junction for each triple junction.

@use_these: (optional) *NX_CHAR*

The specific indices_polyline whereby to resolve ambiguities.

boundary_contact: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_tj])

True, if the triple line makes contact with the edge of the ROI. False, if the triple line does not make contact with the edge of the ROI.

line_energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_tj]) {units=*NX_ANY*}

Specific line energy of each triple junction

mobility: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_tj]) {units=*NX_ANY*}

Non-intrinsic mobility of each triple junction.

length: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_tj]) {units=*NX_LENGTH*}

The length of each triple junction.

This is not necessarily the same as the length of the individual polyline segments whereby the junction is discretized.

volume: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_tj]) {units=*NX_VOLUME*}

The volume about each triple junction.

Respective cut-off criteria need to be specified.

quadruple_junctions: (optional) *NXmicrostructure_feature*

Quadruple junctions as a region where four crystals meet.

An example for a point (like) defect.

Thermodynamically such junctions can be unstable. Specifically when discretizations are used in simulations that do not address the thermodynamics of and splitting characteristics of junctions in cases when more than four crystals meet, it is possible that so-called higher-order junctions are observed.

representation: (optional) *NX_CHAR*

Reference to an instance of:

- *NXcg_point*
- *NXcg_grid* for regularly pixelated (in 1D, 2D) or voxelated representations (in 3D) using (boolean) masks

which represent the geometrical entities of the discretization.

number_of_junctions: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of quadruple junctions.

index_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

First identifier to identify quadruple junctions implicitly.

indices_quadruple_junction: (optional) *NX_INT* (Rank: 1, Dimensions: [i]) {units=*NX_UNITLESS*}

Identifier to identify each quadruple junction explicitly.

location: (optional) *NX_INT* (Rank: 1, Dimensions: [n_qj]) {units=*NX_UNITLESS*}

Set of identifier for positions whereby to identify the location of each junction.

@use_these: (optional) *NX_CHAR*

The specific point identifier whereby to resolve ambiguities.

position: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_qj, d]) {units=*NX_LENGTH*}

Explicit positions.

indices_crystal: (optional) *NX_INT* (Rank: 2, Dimensions: [n_qj, 4]) {units=*NX_UNITLESS*}

Set of tuples of identifier of crystals connected to the junction for each junction.

@use_these: (optional) *NX_CHAR*

The specific identifier to instances of crystal identifiers whereby to resolve ambiguities.

indices_interface: (optional) *NX_INT* (Rank: 2, Dimensions: [n_qj, 4]) {units=*NX_UNITLESS*}

Set of tuples of identifier of interfaces connected to the junction for each junction.

@use_these: (optional) *NX_CHAR*

The specific identifier to instances of interface identifiers whereby to resolve ambiguities.

indices_triple_junction: (optional) *NX_INT* (Rank: 2, Dimensions: [n_qj, 3]) {units=*NX_UNITLESS*}

Set of tuples of identifier for triple junctions connected to the junction for each quadruple junction.

@use_these: (optional) *NX_CHAR*

The specific identifier to instances of triple junction identifiers whereby to resolve ambiguities.

indices_phase: (optional) *NX_INT* (Rank: 2, Dimensions: [n_qj, 4]) {units=*NX_UNITLESS*}

Set of tuples of identifier for phases of crystals connected to the junction for each quadruple junction.

@use_these: (optional) *NX_CHAR*

The specific identifier to instances of phase identifier whereby to resolve ambiguities.

boundary_contact: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_qj])

True, if the junction makes contact with the edge of the ROI. True, if the junction does not make contact with the edge of the ROI.

energy: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_qj]) {units=*NX_ANY*}

Energy of the quadruple_junction as a defect.

mobility: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_qj]) {units=*NX_ANY*}

Non-intrinsic mobility of each quadruple_junction.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure/CG_GRID-group*
- */NXmicrostructure/CG_POINT-group*
- */NXmicrostructure/CG_POLYGON-group*
- */NXmicrostructure/CG_POLYHEDRON-group*
- */NXmicrostructure/CG_POLYLINE-group*
- */NXmicrostructure/CG_TRIANGLE-group*
- */NXmicrostructure/chemical_composition-group*
- */NXmicrostructure/comment-field*
- */NXmicrostructure/configuration-group*
- */NXmicrostructure/configuration/algorithm-field*
- */NXmicrostructure/configuration/dimensionality-field*
- */NXmicrostructure/configuration/disorientation_threshold-field*
- */NXmicrostructure/configuration/PROGRAM-group*
- */NXmicrostructure/crystals-group*
- */NXmicrostructure/crystals/area-field*
- */NXmicrostructure/crystals/boundary_contact-field*
- */NXmicrostructure/crystals/index_offset-field*
- */NXmicrostructure/crystals/indices_crystal-field*
- */NXmicrostructure/crystals/indices_phase-field*
- */NXmicrostructure/crystals/length-field*
- */NXmicrostructure/crystals/number_of_crystals-field*
- */NXmicrostructure/crystals/number_of_phases-field*

- */NXmicrostructure/crystals/orientation-group*
- */NXmicrostructure/crystals/orientation_spread-field*
- */NXmicrostructure/crystals/representation-field*
- */NXmicrostructure/crystals/volume-field*
- */NXmicrostructure/date-field*
- */NXmicrostructure/interfaces-group*
- */NXmicrostructure/interfaces/area-field*
- */NXmicrostructure/interfaces/boundary_contact-field*
- */NXmicrostructure/interfaces/index_offset-field*
- */NXmicrostructure/interfaces/indices_crystal-field*
- */NXmicrostructure/interfaces/indices_crystal@use_these-attribute*
- */NXmicrostructure/interfaces/indices_interface-field*
- */NXmicrostructure/interfaces/indices_phase-field*
- */NXmicrostructure/interfaces/indices_phase@use_these-attribute*
- */NXmicrostructure/interfaces/indices_triple_junction-field*
- */NXmicrostructure/interfaces/indices_triple_junction@use_these-attribute*
- */NXmicrostructure/interfaces/length-field*
- */NXmicrostructure/interfaces/mobility-field*
- */NXmicrostructure/interfaces/number_of_interfaces-field*
- */NXmicrostructure/interfaces/number_of_triple_junctions-field*
- */NXmicrostructure/interfaces/representation-field*
- */NXmicrostructure/interfaces/surface_energy-field*
- */NXmicrostructure/iteration-field*
- */NXmicrostructure/phases-group*
- */NXmicrostructure/phases/index_offset-field*
- */NXmicrostructure/phases/PHASE-group*
- */NXmicrostructure/quadruple_junctions-group*
- */NXmicrostructure/quadruple_junctions/boundary_contact-field*
- */NXmicrostructure/quadruple_junctions/energy-field*
- */NXmicrostructure/quadruple_junctions/index_offset-field*
- */NXmicrostructure/quadruple_junctions/indices_crystal-field*
- */NXmicrostructure/quadruple_junctions/indices_crystal@use_these-attribute*
- */NXmicrostructure/quadruple_junctions/indices_interface-field*
- */NXmicrostructure/quadruple_junctions/indices_interface@use_these-attribute*
- */NXmicrostructure/quadruple_junctions/indices_phase-field*
- */NXmicrostructure/quadruple_junctions/indices_phase@use_these-attribute*

- */NXmicrostructure/quadruple_junctions/indices_quadruple_junction-field*
- */NXmicrostructure/quadruple_junctions/indices_triple_junction-field*
- */NXmicrostructure/quadruple_junctions/indices_triple_junction@use_these-attribute*
- */NXmicrostructure/quadruple_junctions/location-field*
- */NXmicrostructure/quadruple_junctions/location@use_these-attribute*
- */NXmicrostructure/quadruple_junctions/mobility-field*
- */NXmicrostructure/quadruple_junctions/number_of_junctions-field*
- */NXmicrostructure/quadruple_junctions/position-field*
- */NXmicrostructure/quadruple_junctions/representation-field*
- */NXmicrostructure/time-field*
- */NXmicrostructure/triple_junctions-group*
- */NXmicrostructure/triple_junctions/boundary_contact-field*
- */NXmicrostructure/triple_junctions/index_offset-field*
- */NXmicrostructure/triple_junctions/indices_crystal-field*
- */NXmicrostructure/triple_junctions/indices_interface-field*
- */NXmicrostructure/triple_junctions/indices_interface@use_these-attribute*
- */NXmicrostructure/triple_junctions/indices_polyline-field*
- */NXmicrostructure/triple_junctions/indices_polyline@use_these-attribute*
- */NXmicrostructure/triple_junctions/indices_triple_junction-field*
- */NXmicrostructure/triple_junctions/length-field*
- */NXmicrostructure/triple_junctions/line_energy-field*
- */NXmicrostructure/triple_junctions/location-field*
- */NXmicrostructure/triple_junctions/location@use_these-attribute*
- */NXmicrostructure/triple_junctions/mobility-field*
- */NXmicrostructure/triple_junctions/number_of_junctions-field*
- */NXmicrostructure/triple_junctions/position-field*
- */NXmicrostructure/triple_junctions/representation-field*
- */NXmicrostructure/triple_junctions/volume-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure.nxdl.xml

NXmicrostructure_feature

Status:

base class (contribution), extends *NXobject*

Description:

Base class for documenting structuring features of a microstructure.

Instances of the class enable sub-grouping of microstructural features as the abstract base class NXobject should not be used for this purpose.

Symbols:

No symbol table

Groups cited:
NXchemical_composition

Structure:

chemical_composition: (optional) *NXchemical_composition*

The chemical composition of this microstructural feature or set of such features.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_feature/chemical_composition-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_feature.nxdl.xml

NXmicrostructure_ipf

Status:

base class (contribution), extends *NXprocess*

Description:

Base class to store an inverse pole figure (IPF) mapping (IPF map).

Symbols:

v: Number of pixel along the slow direction used for the IPF color key.

u: Number of pixel along the fast direction used for the IPF color key.

n_z: Number of pixel along the slowest direction, typically labeled z or k.

n_y: Number of pixel along the slow direction, typically labeled y or j.

n_x: Number of pixel along the fast direction, typically labeled x or i.

n_rgb: Number of RGB values along the fastest direction, always three.

Groups cited:
NXcg_grid, *NXdata*

Structure:

depends_on: (optional) *NX_CHAR*

Reference to an instance of *NXcoordinate_system* in which the axes axis_z, axis_y, and axis_x are defined.

color_model: (optional) *NX_CHAR*

The algorithm whereby orientations are colored.

Any of these values: `tsl|mtex`

projection_direction: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [3]) {units=*NX_UNITLESS*}

The direction normal vector along which orientations are projected.

@depends_on: (optional) *NX_CHAR*

Reference to an instance of *NXcoordinate_system* in which the projection_direction is defined.

If the field depends_on is not provided but parents of the instance of this base class or its specializations define an instance of *NXcoordinate_system*, projection_direction is defined in this coordinate system.

If nothing is provided, it is assumed that projection_direction is defined in the McStas coordinate system.

interpolation: (optional) *NX_CHAR*

How where orientation values at positions of input_grid computed to values on output_grid.

Nearest neighbour means the orientation of the closed (Euclidean distance) grid point of the input_grid was taken.

Obligatory value: `nearest_neighbour`

input_grid: (optional) *NXcg_grid*

Details about the original grid, i.e. the grid for which the IPF map was computed when that IPF map was exported from the tech partner's file format representation.

output_grid: (optional) *NXcg_grid*

Details about the grid onto which the IPF is recomputed.

Rescaling the visualization of the IPF map may be needed to enable visualization in specific software tools like H5Web.

map: (optional) *NXdata* <=

Inverse pole figure mapping.

Instances named phase0 should by definition refer to the null phase notIndexed. Inspect the definition of *NXphase* and its field phase_id for further details.

Details about possible regridding and associated interpolation during the computation of the IPF map visualization can be stored using the input_grid, output_grid, and interpolation fields.

The main purpose of this map is to offer a normalized default representation of the IPF map for consumption by a research data management system (RDMS).

data: (optional) *NX_NUMBER* {units=*NX_UNITLESS*} <=

Inverse pole figure color code for each map coordinate.

Different types of AXISNAME dimensional scale axes are found in practice. A few examples:

- No scaling, e.g. pixel position values like 0, 1, 2, 3 pixel. Pixels on the map can be distinguished but that map is disconnected from any sample surface context and eventually physical scaling
- Scaling but no offset, e.g. calibrated pixel position 0., 0.5, 1.0, 1.5 micron. Pixels on the map can be compared for their distance to obtain e.g. size of features but the position of the map relative to the e.g. the sample surface is unclear. For IPF maps this is the most frequently reported situation.
- Scaling and offset, which resolves also the absolute position of the map in relation to the sample surface. This is useful information for stitching multiple mappings together and other processing where precise and accurate position data are relevant e.g. for correlative materials characterization.

Three types of dimensional constraints for maps are possible:

- (n_x, 3), a one-dimensional map, typically used for coarse sampling and crystal size statistics.
- (n_y, n_x, 3), a two-dimensional map, the most frequently found reported
- (n_z, n_y, n_x, 3), a three-dimensional map, these are commonly generated using computational methods, or in cases multiple EBSD maps have been stitched/reconstructed into a three-dimensional map.

axis_z: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_z]) {units=*NX_LENGTH*}

Pixel center coordinate calibrated for step size along the z axis of the map.

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Pixel center coordinate calibrated for step size along the y axis of the map.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Pixel center coordinate calibrated for step size along the x axis of the map.

legend: (optional) *NXdata* <=

The color code which maps color to orientation in the fundamental zone.

For each stereographic standard triangle (SST), i.e. a rendering of the fundamental zone of the crystal-symmetry-reduced orientation space SO3, it is possible to define a color model which assigns a color to each point in the fundamental zone.

Different mapping models are used. These implement (slightly) different scaling relations. Differences exist across representations of tech partners.

Differences are which base colors of the RGB color model are placed in which extremal position of the SST and where the white point is located.

For further details see:

- [G. Nolze et al.](https://doi.org/10.1107/S1600576716012942)
- [S. Patala et al.](https://doi.org/10.1016/j.pmatsci.2012.04.002).

Details are implementation-specific and not standardized yet.

data: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [v, u, 3]) {units=*NX_ANY*} <=

Inverse pole figure color code for each map coordinate.

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [v]) {units=*NX_UNITLESS*}

Pixel along the y-axis.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [u]) {units=*NX_UNITLESS*}

Pixel along the x-axis.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_ipf/color_model-field*
- */NXmicrostructure_ipf/depends_on-field*
- */NXmicrostructure_ipf/input_grid-group*
- */NXmicrostructure_ipf/interpolation-field*
- */NXmicrostructure_ipf/legend-group*
- */NXmicrostructure_ipf/legend/axis_x-field*
- */NXmicrostructure_ipf/legend/axis_y-field*
- */NXmicrostructure_ipf/legend/data-field*
- */NXmicrostructure_ipf/map-group*
- */NXmicrostructure_ipf/map/axis_x-field*
- */NXmicrostructure_ipf/map/axis_y-field*
- */NXmicrostructure_ipf/map/axis_z-field*
- */NXmicrostructure_ipf/map/data-field*
- */NXmicrostructure_ipf/output_grid-group*
- */NXmicrostructure_ipf/projection_direction-field*
- */NXmicrostructure_ipf/projection_direction@depends_on-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_ipf.nxdl.xml

NXmicrostructure_kanapy_results

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for the microstructure generator kanapy from ICAMS Bochum.

- A. Hartmeier et al.

A draft application definition to support discussion within the infrastructure use case IUC07 of the NFDI-MatWerk consortium of the German NFDI working on a data model for documenting simulations of spatiotemporal microstructure evolution with scientific software from this community.

Symbols:

n_z: Number of material points along the z axis of the domain.

n_y: Number of material points along the y axis of the domain.

n_x: Number of material points along the x axis of the domain.

c: Number of crystals.

Groups cited:
NXcg_grid, *NXcollection*, *NXcs_profiling*, *NXdata*, *NXentry*, *NXmicrostructure_feature*, *NXmicrostructure*, *NXprogram*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXmicrostructure_kanapy_results`

description: (required) *NX_CHAR*

Discouraged free-text field to add further details to the computation.

start_time: (required) *NX_DATE_TIME* <=

end_time: (recommended) *NX_DATE_TIME* <=

profiling: (optional) *NXcs_profiling*

USER: (optional) *NXuser* <=

program1: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@**version**: (required) *NX_CHAR* <=

@**url**: (recommended) *NX_CHAR* <=

program2: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@**version**: (required) *NX_CHAR* <=

@**url**: (recommended) *NX_CHAR* <=

environment: (optional) *NXcollection* <=

Programs and libraries representing the computational environment

PROGRAM: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@**version**: (required) *NX_CHAR* <=

microstructureID: (required) *NXmicrostructure*

grid: (required) *NXcg_grid* <=

extent: (required) *NX_UINT*

cell_dimensions: (required) *NX_NUMBER* <=

origin: (required) *NX_NUMBER* <=

symmetry: (required) *NX_CHAR* <=

structure: (required) *NXdata* <=

Default plot showing the grid.

@**signal**: (required) *NX_CHAR* <=

@**axes**: (required) *NX_CHAR* <=

@AXISNAME_**indices**: (required) *NX_UINT*

title: (required) *NX_CHAR* <=

indices_crystal: (required) *NX_INT* {units=*NX_UNITLESS*}

Crystal identifier that was assigned to each material point.

z: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Material point barycenter coordinate along z direction.

@**long_name**: (required) *NX_CHAR*

Coordinate along z direction.

y: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_LENGTH*}

Material point barycenter coordinate along y direction.

@**long_name**: (required) *NX_CHAR*

Coordinate along y direction.

x: (required) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_LENGTH*}

Material point barycenter coordinate along x direction.

@**long_name**: (required) *NX_CHAR*

Coordinate along x direction.

crystals: (required) *NXmicrostructure_feature* <=

reference: (required) *NX_CHAR*

number_of_crystals: (required) *NX_UINT* <=

number_of_phases: (required) *NX_UINT* <=

indices_crystal: (required) *NX_INT* (Rank: 1, Dimensions: [c]) <=

indices_phase: (required) *NX_INT* (Rank: 1, Dimensions: [c]) <=

area: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [c]) <=

volume: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [c]) <=

bunge_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [c, 3]) {units=*NX_ANGLE*}

Bunge-Euler angle orientation of each crystal.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_kanapy_results/ENTRY-group*
- */NXmicrostructure_kanapy_results/ENTRY/definition-field*
- */NXmicrostructure_kanapy_results/ENTRY/description-field*
- */NXmicrostructure_kanapy_results/ENTRY/end_time-field*
- */NXmicrostructure_kanapy_results/ENTRY/environment-group*
- */NXmicrostructure_kanapy_results/ENTRY/environment/PROGRAM-group*
- */NXmicrostructure_kanapy_results/ENTRY/environment/PROGRAM/program-field*
- */NXmicrostructure_kanapy_results/ENTRY/environment/PROGRAM/program@version-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID-group*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals-group*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/area-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/bunge_euler-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/indices_crystal-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/indices_phase-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/number_of_crystals-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/number_of_phases-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/reference-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/crystals/volume-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid-group*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/cell_dimensions-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/extent-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/origin-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure-group*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/indices_crystal-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/title-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/x-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/x@long_name-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/y-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/y@long_name-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/z-field*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure/z@long_name-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure@axes-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure@AXISNAME_indices-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/structure@signal-attribute*

- */NXmicrostructure_kanapy_results/ENTRY/microstructureID/grid/symmetry-field*
- */NXmicrostructure_kanapy_results/ENTRY/profiling-group*
- */NXmicrostructure_kanapy_results/ENTRY/program1-group*
- */NXmicrostructure_kanapy_results/ENTRY/program1/program-field*
- */NXmicrostructure_kanapy_results/ENTRY/program1/program@url-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/program1/program@version-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/program2-group*
- */NXmicrostructure_kanapy_results/ENTRY/program2/program-field*
- */NXmicrostructure_kanapy_results/ENTRY/program2/program@url-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/program2/program@version-attribute*
- */NXmicrostructure_kanapy_results/ENTRY/start_time-field*
- */NXmicrostructure_kanapy_results/ENTRY/USER-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_kanapy_results.nxdl.xml

NXmicrostructure_mtex_config

Status:

base class (contribution), extends *NXparameters*

Description:

Base class to store the configuration when using the MTex/Matlab software.

MTex is a Matlab package for texture analysis used in the Materials and Earth Sciences. See R. Hielscher et al. and the MTex source code for details.

Symbols:

n_def_color_map: Number of entries in the default color map

n_color_map: Number of entries in color map

Groups cited:
NXcollection

Structure:

conventions: (optional) *NXcollection* <=

MTex reference frame and orientation conventions. Consult the MTex docs for details.

x_axis_direction: (optional) *NX_CHAR*

TODO with MTex developers

z_axis_direction: (optional) *NX_CHAR*

TODO with MTex developers

a_axis_direction: (optional) *NX_CHAR*

TODO with MTex developers

b_axis_direction: (optional) *NX_CHAR*

TODO with MTex developers

euler_angle: (optional) *NX_CHAR*

TODO with MTex developers

Any of these values: `unknown` | `undefined` | `bunge`

plotting: (optional) *NXcollection* <=

Settings relevant for generating plots.

font_size: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

inner_plot_spacing: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

outer_plot_spacing: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

marker_size: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

figure_size: (optional) *NX_NUMBER*

TODO with MTex developers

show_micron_bar: (optional) *NX_BOOLEAN*

True, if MTex renders a scale bar with figures.

show_coordinates: (optional) *NX_BOOLEAN*

True, if MTex renders a grid with figures.

pf_anno_fun_hdl: (optional) *NX_CHAR*

Code for the function handle used for annotating pole figure plots.

color_map: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_color_map, 3]) {units=*NX_UNITLESS*}

TODO with MTex developers

default_color_map: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_def_color_map, 3]) {units=*NX_UNITLESS*}

TODO with MTex developers

color_palette: (optional) *NX_CHAR*

degree_char: (optional) *NX_CHAR*

TODO with MTex developers

arrow_char: (optional) *NX_CHAR*

TODO with MTex developers

marker: (optional) *NX_CHAR*

TODO with MTex developers

marker_edge_color: (optional) *NX_CHAR*

TODO with MTex developers

marker_face_color: (optional) *NX_CHAR*

TODO with MTex developers

hit_test: (optional) *NX_BOOLEAN*

TODO with MTex developers

miscellaneous: (optional) *NXcollection* <=

Miscellaneous other settings of MTex.

mosek: (optional) *NX_BOOLEAN*

TODO with MTex developers

generating_help_mode: (optional) *NX_BOOLEAN*

TODO with MTex developers

methods_advise: (optional) *NX_BOOLEAN*

TODO with MTex developers

stop_on_symmetry_mismatch: (optional) *NX_BOOLEAN*

TODO with MTex developers

inside_poly: (optional) *NX_BOOLEAN*

TODO with MTex developers

text_interpreter: (optional) *NX_CHAR*

TODO with MTex developers

numerics: (optional) *NXcollection* <=

Miscellaneous settings relevant for numerics.

eps: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Return value of the Matlab eps command.

fft_accuracy: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

max_stwo_bandwidth: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

max_sothree_bandwidth: (optional) *NX_NUMBER* {units=*NX_ANY*}

TODO with MTex developers

system: (optional) *NXcollection* <=

Miscellaneous settings relevant of the system where MTex runs.

memory: (optional) *NX_NUMBER*

TODO with MTex developers

open_gl_bug: (optional) *NX_BOOLEAN*

TODO with MTex developers

save_to_file: (optional) *NX_BOOLEAN*

TODO with MTex developers

path: (optional) *NXcollection* <=

Collection of paths from where MTex reads information and code.

mtex: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

data: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

cif: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

ebsd: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

pf: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

odf: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

tensor: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

example: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

import_wizard: (optional) *NX_CHAR*

Absolute path to specific component of MTex source code.

pf_extensions: (optional) *NX_CHAR*

List of file type suffixes for which MTex assumes texture/pole figure information.

ebsd_extensions: (optional) *NX_CHAR*

List of file type suffixes for which MTex assumes EBSD content.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_mtex_config/conventions-group*
- */NXmicrostructure_mtex_config/conventions/a_axis_direction-field*
- */NXmicrostructure_mtex_config/conventions/b_axis_direction-field*
- */NXmicrostructure_mtex_config/conventions/euler_angle-field*
- */NXmicrostructure_mtex_config/conventions/x_axis_direction-field*
- */NXmicrostructure_mtex_config/conventions/z_axis_direction-field*
- */NXmicrostructure_mtex_config/miscellaneous-group*
- */NXmicrostructure_mtex_config/miscellaneous/generating_help_mode-field*

- /NXmicrostructure_mtex_config/miscellaneous/inside_poly-field
- /NXmicrostructure_mtex_config/miscellaneous/methods_advise-field
- /NXmicrostructure_mtex_config/miscellaneous/mosek-field
- /NXmicrostructure_mtex_config/miscellaneous/stop_on_symmetry_mismatch-field
- /NXmicrostructure_mtex_config/miscellaneous/text_interpreter-field
- /NXmicrostructure_mtex_config/numerics-group
- /NXmicrostructure_mtex_config/numerics/eps-field
- /NXmicrostructure_mtex_config/numerics/fft_accuracy-field
- /NXmicrostructure_mtex_config/numerics/max_sothree_bandwidth-field
- /NXmicrostructure_mtex_config/numerics/max_stwo_bandwidth-field
- /NXmicrostructure_mtex_config/path-group
- /NXmicrostructure_mtex_config/path/cif-field
- /NXmicrostructure_mtex_config/path/data-field
- /NXmicrostructure_mtex_config/path/ebsd-field
- /NXmicrostructure_mtex_config/path/ebsd_extensions-field
- /NXmicrostructure_mtex_config/path/example-field
- /NXmicrostructure_mtex_config/path/import_wizard-field
- /NXmicrostructure_mtex_config/path/mtex-field
- /NXmicrostructure_mtex_config/path/odf-field
- /NXmicrostructure_mtex_config/path/pf-field
- /NXmicrostructure_mtex_config/path/pf_extensions-field
- /NXmicrostructure_mtex_config/path/tensor-field
- /NXmicrostructure_mtex_config/plotting-group
- /NXmicrostructure_mtex_config/plotting/arrow_char-field
- /NXmicrostructure_mtex_config/plotting/color_map-field
- /NXmicrostructure_mtex_config/plotting/color_palette-field
- /NXmicrostructure_mtex_config/plotting/default_color_map-field
- /NXmicrostructure_mtex_config/plotting/degree_char-field
- /NXmicrostructure_mtex_config/plotting/figure_size-field
- /NXmicrostructure_mtex_config/plotting/font_size-field
- /NXmicrostructure_mtex_config/plotting/hit_test-field
- /NXmicrostructure_mtex_config/plotting/inner_plot_spacing-field
- /NXmicrostructure_mtex_config/plotting/marker-field
- /NXmicrostructure_mtex_config/plotting/marker_edge_color-field
- /NXmicrostructure_mtex_config/plotting/marker_face_color-field
- /NXmicrostructure_mtex_config/plotting/marker_size-field

- */NXmicrostructure_mtex_config/plotting/outer_plot_spacing-field*
- */NXmicrostructure_mtex_config/plotting/pf_anno_fun_hdl-field*
- */NXmicrostructure_mtex_config/plotting/show_coordinates-field*
- */NXmicrostructure_mtex_config/plotting/show_micron_bar-field*
- */NXmicrostructure_mtex_config/system-group*
- */NXmicrostructure_mtex_config/system/memory-field*
- */NXmicrostructure_mtex_config/system/open_gl_bug-field*
- */NXmicrostructure_mtex_config/system/save_to_file-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_mtex_config.nxdl.xml

NXmicrostructure_odf

Status:

base class (contribution), extends *NXprocess*

Description:

Base class to store an orientation distribution function (ODF).

An orientation distribution function is a probability distribution that details how much volume of material has a specific orientation. An ODF is computed from pole figure data in a computational process called pole figure inversion.

Symbols:

n_varphi_two: Number of pixel per varphi section plot along the φ_2 slow direction.

n_capital_phi: Number of pixel per varphi section plot along the Φ fast direction.

n_varphi_one: Number of pixel per varphi section plot along the φ_1 fastest direction.

k: Number of local maxima evaluated in the component analysis.

n_pos: Number of sampled positions in orientation space.

Groups cited:
NXdata, *NXparameters*, *NXprocess*

Structure:

configuration: (optional) *NXparameters* <=

Details about the algorithm used for computing the ODF.

crystal_symmetry_point_group: (optional) *NX_CHAR*

Point group of the crystal structure of the phase for which the here documented phase-dependent ODF was computed following the notation of the International Table of Crystallography.

specimen_symmetry_point_group: (optional) *NX_CHAR*

Point group assumed for additionally considered sample symmetries following the notation of the International Table of Crystallography.

kernel_halfwidth: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Halfwidth of the kernel.

kernel_name: (optional) *NX_CHAR*

Name of the kernel.

resolution: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Resolution of the kernel.

characteristics: (optional) *NXprocess*

Group to store descriptors for a rough classification of an ODF.

texture_index: (optional) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

The texture index $t = \int_{\mathcal{SO}(\ni)} f(R)^2 dR$ with $f(R)$, denoting the ODF is evaluated in orientation space $\mathcal{SO}(\ni)$.

The higher it is the texture index the sharper it is the ODF.

kth_extrema: (optional) *NXprocess*

Group to store descriptors and summary statistics for extrema of the ODF.

extrema: (optional) *NX_CHAR*

Minima or maxima, if extrema is set to minima values for location and volume_fraction are sorted in increasing order. If extrema is set to maxima values for location and volume_fraction are sorted in decreasing order. Therefore, the global extremum is always the first entry in location and volume_fraction.

Any of these values: `minima | maxima`

kth: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Number of local extrema evaluated

theta: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Disorientation threshold within which intensity of the ODF is integrated for the component analysis.

location: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [k, 3]) {units=*NX_ANGLE*}

Euler angle representation φ_1, Φ, φ_2 of the kth-most maxima in decreasing order of the intensity maximum.

volume_fraction: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [k]) {units=*NX_ANY*}

Integrated ODF intensity within a theta angular region of the orientation space $SO3$ about each location (obeying symmetries) as specified for each location.

sampling: (optional) *NXprocess*

The ODF intensity values (weights) as sampled with a software.

resolution: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Sampling resolution

euler: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_pos, 3]) {units=*NX_ANGLE*}

Bunge-Euler (i.e. ZXZ convention) locations of each position in orientation space for which a weight was sampled.

weight: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_pos]) {units=*NX_DIMENSIONLESS*}

Weight at each sampled position following the order in euler.

phi_two_plot: (optional) *NXdata* <=

Visualization of the ODF intensity as discretized orthogonal sections through orientation space parameterized using Bunge-Euler angles.

This is one example of typical default plots used in the texture community in materials engineering.

Mind that the orientation space is a distorted space when it using an Euler angle parameterization. Therefore, equivalent orientations show intensity contributions in eventually multiple locations.

intensity: (optional) *NX_NUMBER* (Rank: 3, Dimensions: [n_varphi_two, n_capital_phi, n_varphi_one]) {units=*NX_DIMENSIONLESS*}

ODF intensity at probed locations relative to the intensity of the null model of a random texture.

varphi_one: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_varphi_one]) {units=*NX_ANGLE*} <=

Pixel center angular position along the φ_1 direction.

capital_phi: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_capital_phi]) {units=*NX_ANGLE*}

Pixel center angular position along the Φ direction.

varphi_two: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_varphi_two]) {units=*NX_ANGLE*} <=

Pixel center angular position along the φ_2 direction.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_odf/characteristics-group*
- */NXmicrostructure_odf/characteristics/texture_index-field*
- */NXmicrostructure_odf/configuration-group*
- */NXmicrostructure_odf/configuration/crystal_symmetry_point_group-field*
- */NXmicrostructure_odf/configuration/kernel_halfwidth-field*
- */NXmicrostructure_odf/configuration/kernel_name-field*
- */NXmicrostructure_odf/configuration/resolution-field*
- */NXmicrostructure_odf/configuration/specimen_symmetry_point_group-field*
- */NXmicrostructure_odf/kth_extrema-group*
- */NXmicrostructure_odf/kth_extrema/extrema-field*
- */NXmicrostructure_odf/kth_extrema/kth-field*
- */NXmicrostructure_odf/kth_extrema/location-field*
- */NXmicrostructure_odf/kth_extrema/theta-field*
- */NXmicrostructure_odf/kth_extrema/volume_fraction-field*

- */NXmicrostructure_odf/phi_two_plot-group*
- */NXmicrostructure_odf/phi_two_plot/capital_phi-field*
- */NXmicrostructure_odf/phi_two_plot/intensity-field*
- */NXmicrostructure_odf/phi_two_plot/varphi_one-field*
- */NXmicrostructure_odf/phi_two_plot/varphi_two-field*
- */NXmicrostructure_odf/sampling-group*
- */NXmicrostructure_odf/sampling/euler-field*
- */NXmicrostructure_odf/sampling/resolution-field*
- */NXmicrostructure_odf/sampling/weight-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_odf.nxdl.xml

NXmicrostructure_pf

Status:

base class (contribution), extends *NXprocess*

Description:

Base class to store a pole figure (PF) computation.

A pole figure is the X-ray diffraction intensity for specific integrated peaks for a hemispherical illumination of a real or virtual specimen.

Symbols:

n_y: Number of pixel per pole figure in the slow direction.

n_x: Number of pixel per pole figure in the fast direction.

Groups cited:
NXdata, *NXparameters*

Structure:

configuration: (optional) *NXparameters* <=

Details about the algorithm that was used to compute the pole figure.

crystal_symmetry_point_group: (optional) *NX_CHAR*

Point group of the crystal structure of the phase for which the pole figure was computed following the notation of the International Table of Crystallography.

specimen_symmetry_point_group: (optional) *NX_CHAR*

Point group of assumed sample symmetries following the notation of the International Table of Crystallography.

halfwidth: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Halfwidth of the kernel.

miller_indices: (optional) *NX_CHAR*

Miller ($(hkl)[uvw]$) or Miller-Bravais indices used to specify the pole figure.

resolution: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Resolution of the kernel.

pf: (optional) *NXdata <=*

Pole figure.

intensity: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n_y, n_x]) {units=*NX_UNITLESS*}

Pole figure intensity.

axis_y: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_y]) {units=*NX_ANY*}

Pixel center along y direction in the equatorial plane of a stereographic projection of the unit sphere.

axis_x: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [n_x]) {units=*NX_ANY*}

Pixel center along x direction in the equatorial plane of a stereographic projection of the unit sphere.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_pf/configuration-group*
- */NXmicrostructure_pf/configuration/crystal_symmetry_point_group-field*
- */NXmicrostructure_pf/configuration/halfwidth-field*
- */NXmicrostructure_pf/configuration/miller_indices-field*
- */NXmicrostructure_pf/configuration/resolution-field*
- */NXmicrostructure_pf/configuration/specimen_symmetry_point_group-field*
- */NXmicrostructure_pf/pf-group*
- */NXmicrostructure_pf/pf/axis_x-field*
- */NXmicrostructure_pf/pf/axis_y-field*
- */NXmicrostructure_pf/pf/intensity-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_pf.nxdl.xml

NXmicrostructure_score_config

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition to configure a simulation with the SCORE model.

- M. Kühbach et al.
- M. Diehl et al.

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_ori: Number of texture components to analyze.

n_drag: Number of support points for the linearized drag profile.

n_temp: Number of support points for the desired time-temperature profile.

n_defrag: Number of entries when to defragment i.e. garbage collect the memory holding state information for recrystallized cells.

n_snapshot: Number of entries when to collect snapshots of the evolving microstructure.

n_su: Number of solitary unit domains to export.

d: Dimensionality of the simulation.

Groups cited:
NXcg_grid, *NXcollection*, *NXcs_profiling*, *NXdata*, *NXentry*, *NXmicrostructure*, *NXnote*, *NXparameters*, *NXprogram*, *NXsample*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXmicrostructure_score_config`

identifier_simulation: (required) *NX_UINT*

An alias to refer to this simulation.

description: (optional) *NX_CHAR*

Discouraged free-text field to add further details to the computation.

start_time: (required) *NX_DATE_TIME* <=

ISO 8601 time code with local time zone offset to UTC information included when the configuration file was created.

profiling: (optional) *NXcs_profiling*

userID: (optional) *NXuser* <=

sample: (recommended) *NXsample* <=

dimensionality: (required) *NX_POSINT* {units=*NX_UNITLESS*}

Dimensionality of the simulation.

Any of these values: `1 | 2 | 3`

is_simulation: (required) *NX_CHAR*

A qualifier whether the sample is a real one or a virtual one.

Any of these values: `experiment | simulation`

atom_types: (required) *NX_CHAR*

List of comma-separated elements from the periodic table that are contained in the specimen. If the specimen 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 other sources.

program1: (recommended) *NXprogram*

Name of the program whereby this config file was created.

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

environment: (recommended) *NXcollection* <=

Programs and libraries representing the computational environment

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

material: (required) *NXparameters* <=

(Mechanical) properties of the material which scale the amount of stored (elastic) energy in the system and thus mainly affect recrystallization kinetics.

shear_modulus: (required) *NX_FLOAT* {units=*NX_PRESSURE*}

Reference shear modulus at zero Kelvin.

burgers_vector: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Magnitude of the Burgers vector at zero Kelvin.

melting_temperature: (required) *NX_FLOAT* {units=*NX_TEMPERATURE*}

Melting temperature

deformation: (required) *NXparameters* <=

Details about the geometry and properties of the polycrystal that represents the starting configuration (typically a deformed microstructure) for the simulation.

model: (required) *NX_CHAR*

Which model should be used to generate a starting microstructure.

- cuboidal, a regular array of equally-shaped cuboidal grains
- poisson_voronoi, a discretized Poisson Voronoi tessellation
- ebsd, a microstructure synthesized based on a simulated or a measured EBSD orientation map
- damask, the result of a simulation from DAMASK.

Any of these values: `cuboidal` | `poisson_voronoi` | `ebsd` | `damask`

extent: (required) *NX_FLOAT* (Rank: 1, Dimensions: [3]) {units=*NX_LENGTH*}

Extent of each deformed grain in voxel along the x, y, and z direction when model is cuboidal.

diameter: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Average spherical diameter when model is poisson_voronoi.

ensemble: (optional) *NXparameters* <=

Settings for instantiating properties of deformed grains when model is cuboidal or poisson.

bunge_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_dg_ori, 3]) {units=*NX_ANGLE*}

Set of Bunge-Euler orientations (φ_1, Φ, φ_2) out of which the orientations of deformed grains are sampled.

stored_energy: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_dg_ori]) {units=*NX_ANY*}

Set of stored elastic energy quantified as a dislocation density which is assigned to deformed grains with orientations from bunge_euler with index queries matching for the bunge_euler and stored_energy fields.

ebsd: (optional) *NXnote* <=

Settings for instantiating properties of deformed grains from an EBSD orientation map when model is cuboidal or poisson.

file_name: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

stepsize: (required) *NX_FLOAT* (Rank: 1, Dimensions: [d]) {units=*NX_LENGTH*}

Extent of the pixel of the EBSD orientation mapping assuming square-shaped pixels or cube-shaped voxels respectively.

damask: (optional) *NXnote* <=

Settings for instantiating properties of deformed grains and nuclei when model is damask.

file_name: (required) *NX_CHAR* <=

Name of the DREAM.3D HDF5 file that was instantiated from the a previously performed DAMASK simulation.

algorithm: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

nucleation: (required) *NXparameters* <=

Phenomenological model according to which recrystallization nuclei are placed into the domain. Studying the growth of these nuclei is the main purpose of a SCORE simulation.

spatial_distribution: (required) *NX_CHAR*

According to which model will the nuclei become distributed spatially:

- csr, complete spatial randomness
- custom, implementation-specific
- gb, nuclei placed at grain boundaries

Any of these values: `csr | damask`

incubation_time: (required) *NX_CHAR*

According to which model will the nuclei start to grow:

- site_saturation, instantaneously

Obligatory value: `site_saturation`

orientation: (required) *NX_CHAR*

According to which model will the nuclei get their orientation assigned:

- ensemble, picking randomly one from ensemble/bunge_euler
- random, picking randomly on the SO3
- damask, picking based on information provided in deformation/damask

Any of these values: `ensemble` | `random` | `damask`

ensemble: (required) *NXparameters <=*

Settings for instantiating properties of nuclei for recrystallizing grains.

bunge_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_rx_ori, 3]) {units=*NX_ANGLE*}

Set of Bunge-Euler orientations (φ_1, Φ, φ_2) out of which the orientations of nuclei/recrystallized grains are sampled.

incubation_time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_rx_ori]) {units=*NX_TIME*}

Incubation time which is assigned to deformed grains with orientations from bunge_euler with index queries matching for the bunge_euler and stored_energy fields.

grain_boundary_mobility: (required) *NXparameters <=*

Model for the assumed mobility of grain boundaries with different disorientation implemented as a parameterized Turnbull's model for thermally-activated grain boundary migration.

model: (required) *NX_CHAR*

Which type of fundamental model for the grain boundary mobility.

Grain boundaries with disorientation angle smaller than 15 degree are considered as low-angle grain boundaries. Other grain boundaries are high-angle boundaries.

Any of these values: `sebald_gottstein` | `rollett_holm`

sebald_gottstein: (optional) *NXparameters <=*

Parameter of the Sebald-Gottstein migration model.

lagb_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

Pre-exponential factor for low-angle grain boundaries.

lagb_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

Migration activation enthalpy for low-angle grain boundaries.

hagb_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

Pre-exponential factor for high-angle grain boundaries.

hagb_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

Migration activation enthalpy for high-angle grain boundaries.

special_pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

Pre-exponential factor for high-angle grain boundaries which in bicrystal or other tailored experiments showed a particular high mobility.

special_enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

Migration activation enthalpy for high-angle grain boundaries which in bicrystal or other tailored experiments showed a particular high mobility.

rollett_holm: (required) *NXparameters* <=

Parameter of the Rollett-Holm migration model.

pre_factor: (required) *NX_FLOAT* {units=*NX_ANY*}

Pre-exponential factor for the fastest grain boundary in the system.

enthalpy: (required) *NX_FLOAT* {units=*NX_ANY*}

Migration activation enthalpy for the fastest grain boundary in the system.

c1: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Mobility scaling factor c_1. Typically 0.99 or higher but not 1.

c2: (required) *NX_FLOAT* {units=*NX_UNITLESS*}

Mobility scaling factor c_2. Typically 5.

c3: (required) *NX_FLOAT* {units=*NX_UNITLESS*}

Mobility scaling factor c_3. Typically 9.

stored_energy_recovery: (required) *NXparameters* <=

Time-dependent reduction of the stored energy to account for recovery effects.

model: (required) *NX_CHAR*

Which type of recovery model.

Obligatory value: `none`

dispersoid_drag: (required) *NXparameters* <=

Reduction of the grain boundary migration speed due to the presence of dispersoids through which the total grain boundary area of the recrystallization front can be reduced while the boundary is arrested at the dispersoids.

model: (required) *NX_CHAR*

Which type of drag model.

Any of these values: `none` | `zener_smith`

zener_smith: (required) *NXparameters* <=

Parameter of the Zener-Smith drag model when model is zener_smith.

pre_factor: (required) *NX_FLOAT* {units=*NX_UNITLESS*}

Configuration-dependent constant which factorizes the drag pressure.

surface_energy: (required) *NX_FLOAT* {units=*NX_ANY*}

Average surface energy of the grain-boundary-dispersoid-surface configuration which factorizes the drag pressure.

radius_evolution: (required) *NXdata* <=

Assumed dispersoid mean radius-time profile

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@time_indices: (required) *NX_UINT*

@radius_indices: (required) *NX_UINT*

title: (required) *NX_CHAR* <=

time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_drag]) {units=*NX_TIME*}

Support point of the linearized curve of simulated time matching a specific support point of the average dispersoid radius.

@long_name: (required) *NX_CHAR*

radius: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_drag]) {units=*NX_LENGTH*}

Support point of the linearized curve of the average dispersoid radius.

@long_name: (required) *NX_CHAR*

component_analysis: (required) *NXparameters* <=

name: (required) *NX_CHAR* (Rank: 1, Dimensions: [n_ori])

Given name of a texture component.

bunge_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_ori, 3]) {units=*NX_ANGLE*}

Bunge-Euler angle representation φ_1, Φ, φ_2 of the of texture components in sequence of the name field.

theta: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_ori]) {units=*NX_ANGLE*}

Integration radius that constraints the theta angular region of the orientation space (SO3) about each central location (obeying symmetries) as specified by bunge_euler indexed in the same sequence as the bunge_euler and name fields.

time_temperature: (required) *NXdata* <=

Desired simulated time-temperature profile

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@time_indices: (required) *NX_UINT*

@temperature_indices: (required) *NX_UINT*

title: (required) *NX_CHAR* <=

time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_temp]) {units=*NX_TIME*}

Support point of the linearized curve of simulated time matching a specific support point of the temperature.

@long_name: (required) *NX_CHAR*

temperature: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_temp]) {units=*NX_TEMPERATURE*}

Support point of the linearized curve of the temperature.

@long_name: (required) *NX_CHAR*

discretization: (required) *NXmicrostructure*

Relevant data to instantiate a starting configuration that is typically a microstructure in deformed conditions where (elastic) energy is stored in the form of crystal defects (mostly dislocations). The SCORE model does not resolve individual dislocations but works with one homogenized mean-field density per grain. For simulations that are instantiated from EBSD datasets or crystal plasticity simulations individual values are available for each voxel that may be used as is for each voxel or may need a pre-processing of the data to coarse-grain material point-specific values to values averaged per deformed grain. **grid**: (required) *NXcg_grid* <=

extent: (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.

cell_dimensions: (required) *NX_FLOAT* {units=*NX_LENGTH*}

Edge length of the material point that in SCORE is discretized via equi-sized cubic voxels.

numerics: (required) *NXparameters* <=

Criteria which enable to stop the simulation in a controlled manner and assure a stable numerical integration. Whichever criterion is fulfilled first stops the simulation.

max_x: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Maximum recrystallized volume fraction.

max_time: (required) *NX_FLOAT* {units=*NX_TIME*}

Maximum simulated physical time.

max_iteration: (required) *NX_UINT* {units=*NX_UNITLESS*}

Maximum number of iteration steps.

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.

x_set: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_snapshot]) {units=*NX_DIMENSIONLESS*}

List of target values at which recrystallized volume fractions the state of the CA is evaluated and stored. The code documents summary statistics like recrystallized volume fraction for each iteration and the volume of each grain.

Furthermore, snapshots of the microstructure are stored. These can take much disk space though because SCORE is able to evolve CA with up to 1600^3 cells. Snapshot data document the current microstructure including the assignment of grains and cells surplus the state of the recrystallization front.

Despite these, data about the cells that define the recrystallization front make up for approximately one order of magnitude less cells than present in the domain. For the cells in this front, though, more data have to be collected than just a grain identifier.

cell_cache: (required) *NXparameters* <=

Parameter which control the memory management of cells in the recrystallization front.

initial: (required) *NX_FLOAT* {units=*NX_UNITLESS*}

Fraction of the total number of cells in the CA which should initially be allocated for offering storage for cells making up the recrystallization front.

realloc: (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: (required) *NX_BOOLEAN*

Should the cache for cells in the recrystallization front be defragmented on-the-fly or not.

defragment_x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_defrag]) {units=*NX_DIMENSIONLESS*}

Target values at which recrystallized volume fraction the cache for cells in the recrystallization front will be defragmented on-the-fly. Defragmentation packs active cells closer into main memory to reduce cache misses in subsequent evaluations of the recrystallization front.

solitary_unit: (required) *NXparameters* <=

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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_score_config/ENTRY-group*
- */NXmicrostructure_score_config/ENTRY/component_analysis-group*
- */NXmicrostructure_score_config/ENTRY/component_analysis/bunge_euler-field*
- */NXmicrostructure_score_config/ENTRY/component_analysis/name-field*
- */NXmicrostructure_score_config/ENTRY/component_analysis/theta-field*
- */NXmicrostructure_score_config/ENTRY/definition-field*
- */NXmicrostructure_score_config/ENTRY/deformation-group*
- */NXmicrostructure_score_config/ENTRY/deformation/damask-group*
- */NXmicrostructure_score_config/ENTRY/deformation/damask/algorithm-field*
- */NXmicrostructure_score_config/ENTRY/deformation/damask/checksum-field*
- */NXmicrostructure_score_config/ENTRY/deformation/damask/file_name-field*
- */NXmicrostructure_score_config/ENTRY/deformation/diameter-field*
- */NXmicrostructure_score_config/ENTRY/deformation/ebsd-group*
- */NXmicrostructure_score_config/ENTRY/deformation/ebsd/algorithm-field*
- */NXmicrostructure_score_config/ENTRY/deformation/ebsd/checksum-field*
- */NXmicrostructure_score_config/ENTRY/deformation/ebsd/file_name-field*
- */NXmicrostructure_score_config/ENTRY/deformation/ebsd/stepsize-field*
- */NXmicrostructure_score_config/ENTRY/deformation/ensemble-group*
- */NXmicrostructure_score_config/ENTRY/deformation/ensemble/bunge_euler-field*
- */NXmicrostructure_score_config/ENTRY/deformation/ensemble/stored_energy-field*
- */NXmicrostructure_score_config/ENTRY/deformation/extent-field*
- */NXmicrostructure_score_config/ENTRY/deformation/model-field*
- */NXmicrostructure_score_config/ENTRY/description-field*
- */NXmicrostructure_score_config/ENTRY/discretization-group*
- */NXmicrostructure_score_config/ENTRY/discretization/grid-group*
- */NXmicrostructure_score_config/ENTRY/discretization/grid/cell_dimensions-field*
- */NXmicrostructure_score_config/ENTRY/discretization/grid/extent-field*
- */NXmicrostructure_score_config/ENTRY/dispersoid_drag-group*
- */NXmicrostructure_score_config/ENTRY/dispersoid_drag/model-field*
- */NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith-group*
- */NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/pre_factor-field*
- */NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution-group*
- */NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution/radius-field*

- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution/radius@long_name-attribute
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution/time-field
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution/time@long_name-attribute
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution/title-field
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution@axes-attribute
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution@radius_indices-attribute
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution@signal-attribute
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/radius_evolution@time_indices-attribute
- /NXmicrostructure_score_config/ENTRY/dispersoid_drag/zener_smith/surface_energy-field
- /NXmicrostructure_score_config/ENTRY/environment-group
- /NXmicrostructure_score_config/ENTRY/environment/programID-group
- /NXmicrostructure_score_config/ENTRY/environment/programID/program-field
- /NXmicrostructure_score_config/ENTRY/environment/programID/program@version-attribute
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility-group
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/model-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/rollett_holm-group
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/rollett_holm/c1-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/rollett_holm/c2-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/rollett_holm/c3-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/rollett_holm/enthalpy-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/rollett_holm/pre_factor-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein-group
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein/hagb_enthalpy-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein/hagb_pre_factor-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein/lagb_enthalpy-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein/lagb_pre_factor-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein/special_enthalpy-field
- /NXmicrostructure_score_config/ENTRY/grain_boundary_mobility/sebald_gottstein/special_pre_factor-field
- /NXmicrostructure_score_config/ENTRY/identifier_simulation-field
- /NXmicrostructure_score_config/ENTRY/material-group
- /NXmicrostructure_score_config/ENTRY/material/burgers_vector-field
- /NXmicrostructure_score_config/ENTRY/material/melting_temperature-field
- /NXmicrostructure_score_config/ENTRY/material/shear_modulus-field

- */NXmicrostructure_score_config/ENTRY/nucleation-group*
- */NXmicrostructure_score_config/ENTRY/nucleation/ensemble-group*
- */NXmicrostructure_score_config/ENTRY/nucleation/ensemble/bunge_euler-field*
- */NXmicrostructure_score_config/ENTRY/nucleation/ensemble/incubation_time-field*
- */NXmicrostructure_score_config/ENTRY/nucleation/incubation_time-field*
- */NXmicrostructure_score_config/ENTRY/nucleation/orientation-field*
- */NXmicrostructure_score_config/ENTRY/nucleation/spatial_distribution-field*
- */NXmicrostructure_score_config/ENTRY/numerics-group*
- */NXmicrostructure_score_config/ENTRY/numerics/cell_cache-group*
- */NXmicrostructure_score_config/ENTRY/numerics/cell_cache/defragment-field*
- */NXmicrostructure_score_config/ENTRY/numerics/cell_cache/defragment_x-field*
- */NXmicrostructure_score_config/ENTRY/numerics/cell_cache/initial-field*
- */NXmicrostructure_score_config/ENTRY/numerics/cell_cache/realloc-field*
- */NXmicrostructure_score_config/ENTRY/numerics/max_delta_x-field*
- */NXmicrostructure_score_config/ENTRY/numerics/max_iteration-field*
- */NXmicrostructure_score_config/ENTRY/numerics/max_time-field*
- */NXmicrostructure_score_config/ENTRY/numerics/max_x-field*
- */NXmicrostructure_score_config/ENTRY/numerics/x_set-field*
- */NXmicrostructure_score_config/ENTRY/profiling-group*
- */NXmicrostructure_score_config/ENTRY/program1-group*
- */NXmicrostructure_score_config/ENTRY/program1/program-field*
- */NXmicrostructure_score_config/ENTRY/program1/program@version-attribute*
- */NXmicrostructure_score_config/ENTRY/sample-group*
- */NXmicrostructure_score_config/ENTRY/sample/atom_types-field*
- */NXmicrostructure_score_config/ENTRY/sample/dimensionality-field*
- */NXmicrostructure_score_config/ENTRY/sample/is_simulation-field*
- */NXmicrostructure_score_config/ENTRY/solitary_unit-group*
- */NXmicrostructure_score_config/ENTRY/solitary_unit/apply-field*
- */NXmicrostructure_score_config/ENTRY/solitary_unit/number_of_domains-field*
- */NXmicrostructure_score_config/ENTRY/solitary_unit/rediscretization-field*
- */NXmicrostructure_score_config/ENTRY/start_time-field*
- */NXmicrostructure_score_config/ENTRY/stored_energy_recovery-group*
- */NXmicrostructure_score_config/ENTRY/stored_energy_recovery/model-field*
- */NXmicrostructure_score_config/ENTRY/time_temperature-group*
- */NXmicrostructure_score_config/ENTRY/time_temperature/temperature-field*
- */NXmicrostructure_score_config/ENTRY/time_temperature/temperature@long_name-attribute*

- */NXmicrostructure_score_config/ENTRY/time_temperature/time-field*
- */NXmicrostructure_score_config/ENTRY/time_temperature/time@long_name-attribute*
- */NXmicrostructure_score_config/ENTRY/time_temperature/title-field*
- */NXmicrostructure_score_config/ENTRY/time_temperature@axes-attribute*
- */NXmicrostructure_score_config/ENTRY/time_temperature@signal-attribute*
- */NXmicrostructure_score_config/ENTRY/time_temperature@temperature_indices-attribute*
- */NXmicrostructure_score_config/ENTRY/time_temperature@time_indices-attribute*
- */NXmicrostructure_score_config/ENTRY/userID-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_score_config.nxdl.xml

NXmicrostructure_score_results

Status:

application definition (contribution), extends *NXobject*

Description:

Application definition for storing results of the SCORE cellular automata model.

The SCORE cellular automata model for primary recrystallization is an example of a typical materials engineering application 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 nuclei during static recrystallization. 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_summary_stats: The total number of summary statistic log entries

n_b: Number of boundaries of the bounding box or primitive about the computational 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_hexahedron*, *NXcollection*, *NXcoordinate_system*, *NXcs_profiling*, *NXdata*, *NXentry*, *NXmicrostructure_feature*, *NXmicrostructure*, *NXnote*, *NXprocess*, *NXprogram*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

Obligatory value: `NXmicrostructure_score_results`

identifier_simulation: (required) *NX_UINT*

Simulation ID as an alias to refer to this simulation.

description: (optional) *NX_CHAR*

Discouraged free-text field to add further details to the computation.

start_time: (required) *NX_DATE_TIME* <=

ISO 8601 time code with local time zone offset to UTC information included when the simulation was started.

end_time: (recommended) *NX_DATE_TIME* <=

ISO 8601 time code with local time zone offset to UTC information included when the simulation ended.

config: (required) *NXnote* <=

Configuration file with the parameterization of the SCORE model that was used for this simulation.

file_name: (required) *NX_CHAR* <=

algorithm: (required) *NX_CHAR* <=

checksum: (required) *NX_CHAR* <=

profiling: (optional) *NXcs_profiling*

program1: (required) *NXprogram*

Name of the program with which the simulation was performed.

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

environment: (optional) *NXcollection* <=

Programs and libraries representing the computational environment

programID: (required) *NXprogram*

program: (required) *NX_CHAR* <=

@version: (required) *NX_CHAR* <=

sample_reference_frame: (required) *NXcoordinate_system*

type: (required) *NX_CHAR* <=

Obligatory value: `cartesian`

handedness: (required) *NX_CHAR*

Obligatory value: `right_handed`

origin: (required) *NX_CHAR* <=

Obligatory value: `front_bottom_left`

x_alias: (required) *NX_CHAR* <=

Obligatory value: `rolling_direction`

x_direction: (required) *NX_CHAR* <=

Obligatory value: `east`

y_alias: (required) *NX_CHAR* <=

Obligatory value: `transverse_direction`

y_direction: (required) *NX_CHAR* <=

Obligatory value: `in`

z_alias: (required) *NX_CHAR* <=

Obligatory value: `normal direction`

z_direction: (required) *NX_CHAR* <=

Obligatory value: `north`

discretization: (required) *NXmicrostructure*

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_UINT*

index_offset: (required) *NX_INT* <=

boundary: (required) *NXcg_hexahedron*

A tight bounding box or sphere or bounding primitive about the grid.

number_of_boundaries: (required) *NX_UINT* {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.

boundary_conditions: (required) *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

boundaries: (required) *NX_CHAR* (Rank: 1, Dimensions: [6])

Name of the boundaries. Left, right, front, back, bottom, top, The field must have as many entries as there are number_of_boundaries.

spatiotemporalID: (required) *NXprocess* <=

Documentation of the spatiotemporal evolution for each CA domain.

SCORE is a hybrid parallelized code that can evolve multiple replicas in parallel. The set of replicas is distributed across MPI processes. Each such replica is then evolved via OpenMP multi-threading. **summary_statistics**: (required) *NXprocess*

Summary quantities which are the result of some post-processing of the snapshot data (averaging, integrating, interpolating) happening for practical and performance reasons during the simulation. Place used for storing descriptors from continuum mechanics and thermodynamics at the scale of the entire ROI. **kinetics**: (recommended) *NXdata* <=

Evolution of the recrystallized volume fraction over time.

@signal: (required) *NX_CHAR* <=

@axes: (required) *NX_CHAR* <=

@time_indices: (required) *NX_UINT*

@iteration_indices: (optional) *NX_UINT*

@temperature_indices: (optional) *NX_UINT*

@x_indices: (required) *NX_UINT*

title: (recommended) *NX_CHAR* <=

time: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_summary_stats]) {units=*NX_TIME*}

Evolution of the physical time not to be confused with wall-clock time or profiling data.

iteration: (required) *NX_INT* (Rank: 1, Dimensions: [n_summary_stats]) {units=*NX_UNITLESS*}

Iteration or increment counter.

temperature: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_summary_stats]) {units=*NX_TEMPERATURE*}

Evolution of the simulated temperature over time.

x: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_summary_stats]) {units=*NX_DIMENSIONLESS*} <=

Recrystallized volume fraction.

stress: (optional) *NXdata* <=

type: (required) *NX_CHAR*

Which type of stress.

Obligatory value: `cauchy`

tensor: (required) *NX_FLOAT* (Rank: 3, Dimensions: [n_summary_stats, 3, 3]) {units=*NX_ANY*}

Applied external stress tensor on the ROI.

strain: (optional) *NXdata* <=

type: (required) *NX_CHAR*

Which type of strain.

tensor: (required) *NX_FLOAT* (Rank: 3, Dimensions: [n_summary_stats, 3, 3]) {units=*NX_ANY*}

Applied external strain tensor on the ROI.

deformation_gradient: (optional) *NXprocess*

type: (required) *NX_CHAR*

Which type of deformation gradient.

Obligatory value: `piola`

tensor: (required) *NX_FLOAT* (Rank: 3, Dimensions: [n_summary_stats, 3, 3]) {units=*NX_ANY*}

Applied deformation gradient tensor on the ROI.

microstructureID: (required) *NXmicrostructure*

time: (required) *NX_FLOAT* {units=*NX_TIME*}

Simulated physical time for this snapshot.

iteration: (required) *NX_UINT* {units=*NX_UNITLESS*}

Iteration or increment counter of this snapshot.

temperature: (required) *NX_FLOAT* {units=*NX_TEMPERATURE*}

Simulated temperature for this snapshot.

x: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Current recrystallized volume fraction (taking fractional infections into account).

x_set: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Target value for which a snapshot was requested for the recrystallized volume fraction.

grid: (recommended) *NXcg_grid* <=

indices_crystal: (recommended) *NX_INT* (Rank: 3, Dimensions: [n_x, n_y, n_z]) {units=*NX_UNITLESS*}

Index for each crystal whereby its metadata can be retrieved.

thread_id: (optional) *NX_UINT* (Rank: 3, Dimensions: [n_x, n_y, n_z]) {units=*NX_UNITLESS*}

Identifier of the OpenMP thread that processed this part of the grid.

crystals: (required) *NXmicrostructure_feature* <=

representation: (recommended) *NX_CHAR* <=

number_of_crystals: (recommended) *NX_UINT* <=

number_of_phases: (recommended) *NX_UINT* <=

index_offset_crystal: (recommended) *NX_INT*

indices_crystal: (recommended) *NX_INT* (Rank: 1, Dimensions: [n_grains]) <=

index_offset_phase: (recommended) *NX_INT*

indices_phase: (recommended) *NX_INT* (Rank: 1, Dimensions: [n_grains]) <=

volume: (required) *NX_FLOAT* (Rank: 1, Dimensions: [n_grains]) {units=*NX_VOLUME*}

Volume of each grain (partially transformed cells are accounted for).

bunge_euler: (required) *NX_FLOAT* (Rank: 2, Dimensions: [n_grains, 3]) {units=*NX_ANGLE*}

Bunge-Euler angle triplets for each grain.

dislocation_density: (recommended) *NX_FLOAT* (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.

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.

recrystallization_front: (recommended) *NXmicrostructure_feature*

Details about those cells which in this time step represent the discrete recrystallization front.

Each CA is processed by a team of OpenMP threads.

halo_region: (optional) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Which cells are currently in a halo region of threads.

The halo region is a layer of cells about the sub-domain of the simulation grid evolved by a thread.

mobility_weight: (recommended) *NX_FLOAT* (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 that is modelled sweeping cells that make the discrete recrystallization front.

coordinate: (recommended) *NX_NUMBER* (Rank: 2, Dimensions: [n_front, 3]) {units=*NX_UNITLESS*}

The x, y, z grid coordinates of each cell in the recrystallization front.

deformed_grain_id: (recommended) *NX_UINT* (Rank: 1, Dimensions: [n_front]) {units=*NX_UNITLESS*}

Grain identifier assigned to each cell in the recrystallization front.

recrystallized_grain_id: (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.

thread_id: (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.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_score_results/ENTRY-group*
- */NXmicrostructure_score_results/ENTRY/config-group*
- */NXmicrostructure_score_results/ENTRY/config/algorithm-field*
- */NXmicrostructure_score_results/ENTRY/config/checksum-field*
- */NXmicrostructure_score_results/ENTRY/config/file_name-field*
- */NXmicrostructure_score_results/ENTRY/definition-field*
- */NXmicrostructure_score_results/ENTRY/description-field*
- */NXmicrostructure_score_results/ENTRY/discretization-group*
- */NXmicrostructure_score_results/ENTRY/discretization/boundary-group*
- */NXmicrostructure_score_results/ENTRY/discretization/boundary/boundaries-field*
- */NXmicrostructure_score_results/ENTRY/discretization/boundary/boundary_conditions-field*
- */NXmicrostructure_score_results/ENTRY/discretization/boundary/number_of_boundaries-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid-group*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/cardinality-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/cell_dimensions-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/dimensionality-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/extent-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/index_offset-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/origin-field*
- */NXmicrostructure_score_results/ENTRY/discretization/grid/symmetry-field*
- */NXmicrostructure_score_results/ENTRY/end_time-field*
- */NXmicrostructure_score_results/ENTRY/environment-group*
- */NXmicrostructure_score_results/ENTRY/environment/programID-group*

- /NXmicrostructure_score_results/ENTRY/environment/programID/program-field
- /NXmicrostructure_score_results/ENTRY/environment/programID/program@version-attribute
- /NXmicrostructure_score_results/ENTRY/identifier_simulation-field
- /NXmicrostructure_score_results/ENTRY/profiling-group
- /NXmicrostructure_score_results/ENTRY/program1-group
- /NXmicrostructure_score_results/ENTRY/program1/program-field
- /NXmicrostructure_score_results/ENTRY/program1/program@version-attribute
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame-group
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/handedness-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/origin-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/type-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/x_alias-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/x_direction-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/y_alias-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/y_direction-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/z_alias-field
- /NXmicrostructure_score_results/ENTRY/sample_reference_frame/z_direction-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID-group
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID-group
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals-group
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/bunge_euler-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/dislocation_density-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/index_offset_crystal-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/index_offset_phase-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/indices_crystal-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/indices_phase-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/is_deformed-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/is_recrystallized-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/number_of_crystals-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/number_of_phases-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/representation-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/crystals/volume-field
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/grid-group
- /NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/grid/indices_crystal-field

- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/grid/thread_id-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/iteration-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front-group*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/coordinate-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/deformed_grain_id-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/halo_region-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/infection_direction-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/mobility_weight-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/recrystallized_grain_id-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/recrystallization_front/thread_id-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/temperature-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/time-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/x-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/microstructureID/x_set-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics-group*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/deformation_gradient-group*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/deformation_gradient/tensor-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/deformation_gradient/type-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics-group*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics/iteration-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics/temperature-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics/time-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics/title-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics/x-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics@axes-attribute*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics@iteration_indices-attribute*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics@signal-attribute*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics@temperature_indices-attribute*

- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics@time_indices-attribute*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/kinetics@x_indices-attribute*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/strain-group*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/strain/tensor-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/strain/type-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/stress-group*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/stress/tensor-field*
- */NXmicrostructure_score_results/ENTRY/spatiotemporalID/summary_statistics/stress/type-field*
- */NXmicrostructure_score_results/ENTRY/start_time-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_score_results.nxdl.xml

NXmicrostructure_slip_system

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.

m: Number of indices used for reporting Miller (3) or Miller-Bravais indices (4).

Groups cited:
none

Structure:

lattice_type: (optional) *NX_CHAR*

Bravais lattice type

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 planes.

miller_direction: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [n, m]) {units=*NX_UNITLESS*}

Array of Miller or Miller-Bravais indices that describe the crystallographic direction.

is_specific: (optional) *NX_BOOLEAN* (Rank: 1, Dimensions: [n]) {units=*NX_UNITLESS*}

For each slip system a marker whether the Miller indices refer to a specific slip system or to a set of equivalent crystallographic slip systems.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXmicrostructure_slip_system/is_specific-field*
- */NXmicrostructure_slip_system/lattice_type-field*
- */NXmicrostructure_slip_system/miller_direction-field*
- */NXmicrostructure_slip_system/miller_plane-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXmicrostructure_slip_system.nxdl.xml

NXoptical_fiber

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 or a custom value (if you use a custom value, also set @custom=True): `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=GB/s}

Transfer rate of the fiber (in GB per second).

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=dB/km}

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.

material: (optional) *NX_CHAR*

Specify the material of the core of the fiber.

diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Core diameter of the fiber (e.g. given in micrometer).

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.

material: (optional) *NX_CHAR*

Specify the material of the core of the fiber.

diameter: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Clad diameter of the fiber (e.g. given in micrometer).

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.

material: (optional) *NX_CHAR*

Specify the material of the coating of the fiber.

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.

- */NXoptical_fiber/acceptance_angle-field*
- */NXoptical_fiber/attenuation-field*
- */NXoptical_fiber/attenuation@units-attribute*
- */NXoptical_fiber/cladding-group*
- */NXoptical_fiber/cladding/diameter-field*
- */NXoptical_fiber/cladding/index_of_refraction-field*
- */NXoptical_fiber/cladding/material-field*
- */NXoptical_fiber/coating-group*
- */NXoptical_fiber/coating/diameter-field*
- */NXoptical_fiber/coating/material-field*
- */NXoptical_fiber/core-group*
- */NXoptical_fiber/core/diameter-field*

- */NXoptical_fiber/core/index_of_refraction-field*
- */NXoptical_fiber/core/material-field*
- */NXoptical_fiber/description-field*
- */NXoptical_fiber/dispersion-field*
- */NXoptical_fiber/dispersion_type-field*
- */NXoptical_fiber/length-field*
- */NXoptical_fiber/numerical_aperture-field*
- */NXoptical_fiber/power_loss-field*
- */NXoptical_fiber/spectral_range-field*
- */NXoptical_fiber/spectral_range@units-attribute*
- */NXoptical_fiber/transfer_rate-field*
- */NXoptical_fiber/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXoptical_fiber.nxdl.xml

NXoptical_polarizer

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

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `dichroic`
- `linear`
- `circular`
- `Glan-Thompson prism`
- `Nicol prism`

- `Glan-Taylor prism`
- `Glan-Focault prism`
- `Wollaston prism`
- `Normarski prism`
- `Senarmont prism`
- `thin-film polarizer`
- `wire grid polarizer`

polarizer_angle: (recommended) *NX_NUMBER* {units=*NX_ANGLE*}

Angle of the polarizer.

acceptance_angle: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_ANGLE*}

Acceptance angle of the polarizer (range).

wavelength_range: (recommended) *NX_FLOAT* (Rank: 1, Dimensions: [2]) {units=*NX_WAVELENGTH*}

Wavelength range for which the polarizer is designed. Enter the minimum and maximum wavelength (lower and upper limit) of the range.

extinction_ratio: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum]) {units=*NX_UNITLESS*}

Extinction ratio (maximum to minimum transmission).

reflection: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Reflection of the polarizer at given wavelength values.

transmission: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [N_spectrum_RT]) {units=*NX_UNITLESS*}

Transmission of the polarizer at given wavelength values.

SHAPE: (recommended) *NXshape*

Describe the geometry (shape, dimension etc.) of the device. Specify the dimensions in 'SHAPE/size'. A sketch of the device should be provided in the 'sketch(NXdata)' field to clarify (i) the shape and dimensions of the device, and (ii) the input and outputs (i.e. the direction of the incoming and outcoming (split) beams).

shape: (optional) *NX_CHAR* <=

Describe the shape (plate, cube, wedged, prism etc.).

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `cube`
- `cylinder`
- `plate`
- `prism`
- `wedged`
- `other`

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.).

coatingTYPE: (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. coating_front and coating_back).

type: (optional) *NX_CHAR* <=

Specify the coating type (e.g. dielectric, anti-reflection (AR), multilayer coating etc.).

material: (optional) *NX_CHAR*

Describe the coating material (e.g. MgF2).

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.

- */NXoptical_polarizer/acceptance_angle-field*
- */NXoptical_polarizer/coatingTYPE-group*
- */NXoptical_polarizer/coatingTYPE/index_of_refraction_coating-field*
- */NXoptical_polarizer/coatingTYPE/material-field*
- */NXoptical_polarizer/coatingTYPE/thickness-field*
- */NXoptical_polarizer/coatingTYPE/type-field*
- */NXoptical_polarizer/extinction_ratio-field*
- */NXoptical_polarizer/polarizer_angle-field*
- */NXoptical_polarizer/reflection-field*
- */NXoptical_polarizer/SHAPE-group*
- */NXoptical_polarizer/SHAPE/shape-field*
- */NXoptical_polarizer/SHAPE/size-field*
- */NXoptical_polarizer/SHAPE/sketch-group*
- */NXoptical_polarizer/SHAPE/wedge_angle-field*
- */NXoptical_polarizer/substrate-group*
- */NXoptical_polarizer/substrate/index_of_refraction-field*
- */NXoptical_polarizer/substrate/substrate_material-field*
- */NXoptical_polarizer/substrate/substrate_thickness-field*
- */NXoptical_polarizer/transmission-field*
- */NXoptical_polarizer/type-field*
- */NXoptical_polarizer/wavelength_range-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXoptical_polarizer.nxdl.xml

NXphase_lock_loop

Status:

base class (contribution), extends *NXcomponent*

Description:

A base class definition for a phase-locked loop (PLL) used in AFM experiments.

The PLL has three main components:

1. Phase-detector
2. Loop-filter
3. Voltage-controlled oscillator (VCO)

Depending on the phase error from the phase detector due to the phase difference between the Reference and output from the VCO signals, the PLL adjusts the output frequency to minimize the phase error.

Measuring the phase error, the PLL can be in one of three states:

- Zero phase error: The output signal is in coherence with the reference signal (lock condition).
- **Constant phase error: The output signal has a constant phase difference from the reference signal, and therefore,**
  both reference signals have the same frequency (lock condition).
- **Continuous increasing phase error: The output signal's phase continuously changes therefore they have different frequencies,**
  indicating the PLL is in an unlock condition.

Symbols:

No symbol table

Groups cited:
NXcalibration, *NXcomponent*, *NXfabrication*

Structure:

center_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The center frequency of the PLL is the middle frequency of the PLL's operating range.

frequency_bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The frequency bandwidth of the PLL is where it responds or is stable. If phase difference (also called phase error) is calculated from the input and output signal frequencies.

phase_bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The phase bandwidth of the PLL is where it responds or is stable. If phase difference (also called phase error) is calculated from the input and output signal phases.

frequency_step: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The frequency step size of the PLL determines the resolution of frequency changes.

hardware: (optional) *NXfabrication* <=

Manufacturer and model of the PLL hardware or controller.

type: (optional) *NX_CHAR*

Types (design parameter for circuit) of the PLL from type-1, type-2, and type-3 available in device specifications (for mathematical details: https://www.nxp.com/docs/en/application-note/AN535.pdf). Type is defined by the number of poles that exist in the transfer function of the PLL.

Any of these values: `type-1` | `type-2` | `type-3`

order: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

Order of the PLL. This is characteristic of circuit design and is defined as the number of roots of characteristics polynomial of the PLL circuit.

calibration: (optional) *NXcalibration*

Calibration information for the PLL, including any relevant parameters or procedures.

phase_detector: (optional) *NXcomponent*

The phase detector component of the PLL, which compares the phase between the reference signal generated (e.g., in AFM, generated by the cantilever) and the signal coming from the VCO.

reference_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The input reference frequency to the PLL through the component Phase-detector.

frequency_shift: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

Measured or target shift in oscillation from the VCO. relative to the reference signal.

reference_phase: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Reference phase of the input signal to the phase detector.

phase_error: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

Measured phase error (also called phase difference) between the reference signal and the VCO output.

Kp_coefficient: (optional) *NX_NUMBER* {units=V/rad}

The gain coefficient of the phase detector which determines how much the output voltage changes in response to a phase difference (V/rad).

loop_filter: (optional) *NXcomponent*

The loop filter component of the PLL filters the voltage from the phase detector to the VCO. It extracts the average voltage of the phase error signals emitted from the phase detector.

Kf_coefficient: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

The Kf coefficient, called loop filter gain, determines the gain of the loop filter circuit in the PLL.

voltage_controlled_oscillator: (optional) *NXcomponent*

The voltage-controlled oscillator (VCO) component of the PLL, which generates the output signal based on the filtered voltage from the loop filter.

output_frequency_range: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The output frequency range (written in an array of frequencies as [min, max]) of the VCO which is the range of frequencies it can generate. The range is either the same as the frequency bandwidth of the PLL or shorter and inside that.

Kn_coefficient: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

The Kn coefficient, called the input-output frequency ratio, relates the input frequency to the output frequency of the VCO. This coefficient is used to adjust the output frequency based on the input frequency.

Ko_coefficient: (optional) *NX_NUMBER* {units=*NX_ANY*}

The output voltage coefficient of the VCO, which determines how much the output frequency changes in response to a change in input voltage (Hz/V).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXphase_lock_loop/calibration-group*
- */NXphase_lock_loop/center_frequency-field*
- */NXphase_lock_loop/frequency_bandwidth-field*
- */NXphase_lock_loop/frequency_step-field*
- */NXphase_lock_loop/hardware-group*
- */NXphase_lock_loop/hardware/order-field*
- */NXphase_lock_loop/hardware/type-field*
- */NXphase_lock_loop/loop_filter-group*
- */NXphase_lock_loop/loop_filter/Kf_coefficient-field*
- */NXphase_lock_loop/phase_bandwidth-field*
- */NXphase_lock_loop/phase_detector-group*
- */NXphase_lock_loop/phase_detector/frequency_shift-field*
- */NXphase_lock_loop/phase_detector/Kp_coefficient-field*
- */NXphase_lock_loop/phase_detector/phase_error-field*
- */NXphase_lock_loop/phase_detector/reference_frequency-field*
- */NXphase_lock_loop/phase_detector/reference_phase-field*
- */NXphase_lock_loop/voltage_controlled_oscillator-group*
- */NXphase_lock_loop/voltage_controlled_oscillator/Kn_coefficient-field*
- */NXphase_lock_loop/voltage_controlled_oscillator/Ko_coefficient-field*
- */NXphase_lock_loop/voltage_controlled_oscillator/output_frequency_range-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXphase_lock_loop.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:

Base class 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

NXrcs

Status:

base class (contribution), extends *NXobject*

Description:

A base class for the Real Time Control System (RCS).

A real time control system (RCS) is a reference architecture model used to design highly task intensive software systems and computing control systems.

Symbols:

No symbol table

Groups cited:
NXfabrication

Structure:

name: (optional) *NX_CHAR*

The name of the real-time control system.

description: (optional) *NX_CHAR*

The description of the real-time control system.

type: (optional) *NX_CHAR*

The type of the real-time control system.

frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The frequency of the real-time controller system which indicated the number of close-loop process (gathering data, process data and update system) control cycles per unit time.

measurement_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time of data measuring in a data lifecycle or process
cycle.

acquisition_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Data acquisition time in single data lifecycle or process
cycle of the real-time control system.

animation_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time of data animation of single in a data lifecycle or process
cycle.

indication_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time of data indication of the end of process.

fabrication: (optional) *NXfabrication*

The fabrication information.

model: (optional) *NX_CHAR_OR_NUMBER*

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXrcs/acquisition_time-field*
- */NXrcs/animation_time-field*
- */NXrcs/description-field*
- */NXrcs/fabrication-group*
- */NXrcs/fabrication/model-field*
- */NXrcs/frequency-field*
- */NXrcs/indication_time-field*
- */NXrcs/measurement_time-field*
- */NXrcs/name-field*
- */NXrcs/type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXrcs.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** $\equiv$ **step** in Python slicing.

For example, a ROI sum of an area starting at index of [20,50] and shape [220,120] in image data:

```
detector:NXdetector/
        data[60,256,512]
        region:NXregion/
                @region_type  = "rectangular"
                parent = "data"
                start  = [20,50]
                count  = [220,120]
                statistics:NXdata/
                        @signal = "sum"
                        sum[60]
```

the `sum` dataset contains the summed areas in each frame. Another example, a hyperspectral image downsampled 16-fold in energy:

```
detector:NXdetector/
        data[128,128,4096]
        region:NXregion/
                @region_type  = "rectangular"
                parent = "data"
                start  = [2]
                count  = [20]
                stride = [32]
                block  = [16]
                downsampled:NXdata/
                        @signal = "maximum"
                        @auxiliary_signals = "copy"
                        maximum[128,128,20]
                        copy[128,128,320]
```

the `copy` dataset selects 20 16-channel blocks that start 32 channels apart, the `maximum` dataset will show maximum values in each 16-channel block in every spectra.

Symbols:

These symbols will denote how the shape of the parent group's data field,

$$(D_0, ..., D_{\mathbf{O}-1}, d_0, ..., d_{\mathbf{R}-1})$$

could be split into a left set of **O** outer dimensions, $\boldsymbol{D}$, and a right set of **R** region dimensions, $\boldsymbol{d}$, where the data field has rank **O** + **R**. Note **O** $>= 0$.

O: Outer rank

R: Region rank

Groups cited:
NXdata

Structure:

@region_type: (required) *NX_CHAR*

This is `rectangular` to describe the region as a hyper-rectangle in the index space of its parent group's data field.

Obligatory value: `rectangular`

parent: (optional) *NX_CHAR*

The name of data field in the parent group or the path of a data field relative to the parent group (so it could be a field in a subgroup of the parent group)

parent_mask: (optional) *NX_CHAR*

The name of an optional mask field in the parent group with rank $\boldsymbol{R}$ and dimensions $\boldsymbol{d}$. For example, this could be `pixel_mask` of an *NXdetector*.

start: (recommended) *NX_NUMBER* (Rank: 1, Dimensions: [R])

The starting position for region in detector data field array. This is recommended as it also defines the region rank. If omitted then defined as an array of zeros.

count: (recommended) *NX_INT* (Rank: 1, Dimensions: [R])

The number of blocks or items in the hyperslab selection. If omitted then defined as an array of dimensions that take into account the other hyperslab selection fields to span the parent data field's shape.

stride: (optional) *NX_INT* (Rank: 1, Dimensions: [R])

An optional field to define striding used to downsample data. If omitted then defined as an array of ones.

block: (optional) *NX_INT* (Rank: 1, Dimensions: [R])

An optional field to define the block size used to copy or downsample data. In the i-th dimension, if $\mathbf{block}[i] < \mathbf{stride}[i]$ then the downsampling blocks have gaps between them; when `block` matches `stride` then the blocks are contiguous; otherwise the blocks overlap. If omitted then defined as an array of ones.

scale: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [R])

An optional field to define a divisor for scaling of reduced data. For example, in a downsampled sum, it can reduce the maximum values to fit in the domain of the result data type. In an image that is downsampled by summing 2x2 blocks, using $\mathrm{scale} = 4$ allows the result to fit in the same integer type dataset as the parent dataset. If omitted then no scaling occurs.

downsampled: (optional) *NXdata* <=

An optional group containing data copied/downsampled from parent group's data. Its dataset name must reflect how the downsampling is done over each block. So it could be a reduction operation such as sum, minimum, maximum, mean, mode, median, etc. If downsampling is merely copying each block then use "copy" as the name. Where more than one downsample dataset is written (specified with `@signal`) then add `@auxiliary_signals` listing the others. In the copy case, the field should have a shape of $(D_0, ..., D_{\mathbf{O}-1}, \mathbf{block}[0] * \mathbf{count}[0], ..., \mathbf{block}[\mathbf{R}-1] * \mathbf{count}[\mathbf{R}-1])$, otherwise the expected shape is $(D_0, ..., D_{\mathbf{O}-1}, \mathbf{count}[0], ..., \mathbf{count}[\mathbf{R}-1])$.

The following figure shows how blocks are used in downsampling:

statistics: (optional) *NXdata* <=

An optional group containing any statistics derived from the region in parent group's data such as sum, minimum, maximum, mean, mode, median, rms, variance, etc. Where more than one statistical dataset is written (specified with `@signal`) then add `@auxiliary_signals` listing the others. All data fields should have shapes of $\boldsymbol{D}$.

Fig. 18: A selection with $\mathbf{start} = 2, \mathbf{count} = 4, \mathbf{stride} = 3, \mathbf{block} = 2$ from a dataset with shape [13] will result in the `reduce` dataset of shape [4] and a `copy` dataset of shape [8].

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*, *NXhistory*, *NXinstrument*, *NXnote*, *NXpid_controller*, *NXprocess*, *NXsample*, *NXsensor*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

@default: (recommended) *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 visualized upon entry. See https://www.nexusformat.org/2014_How_to_find_default_data.html for a summary of the discussion.

definition: (required) *NX_CHAR* <=

Obligatory value: `NXsensor_scan`

@version: (required) *NX_CHAR* <=

identifier_experiment: (recommended) *NX_CHAR* <=

The unique identifier for the entry. The identifier is mainly lab-defined and can be a combination of the sample name, date and time, experiment condition (such as temperature) or instrument-generated unique identifier.

identifier_collection: (optional) *NX_CHAR* <=

The unique identifier for the collection. The identifier is used to group a number of the experiments run upon the same setup and/or same sample.

experiment_description: (recommended) *NX_CHAR* <=

start_time: (recommended) *NX_DATE_TIME* <=

The start time of the experiment.

end_time: (recommended) *NX_DATE_TIME* <=

The end time of the experiment.

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.

NOTE: (optional) *NXnote* <=

Any additional information or notes (e.g. purpose of the experiment) that might be useful to understand the experiment.

INSTRUMENT: (recommended) *NXinstrument* <=

ENVIRONMENT: (recommended) *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: (recommended) *NX_CHAR*

A list of names of NXsensor groups used as independently scanned controllers.

measurement_sensors: (recommended) *NX_CHAR*

A list of names of NXsensor groups used as measurement sensors.

SENSOR: (recommended) *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: (recommended) *NXpid_controller*

SAMPLE: (recommended) *NXsample* <=

name: (required) *NX_CHAR* <=

HISTORY: (optional) *NXhistory*

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/identifier_collection-field*
- */NXsensor_scan/ENTRY/identifier_experiment-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/NOTE-group*
- */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/HISTORY-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*
- */NXsensor_scan/ENTRY@default-attribute*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsensor_scan.nxdl.xml

NXseparator

Status:

base class (contribution), extends *NXcomponent*

Description:

Base class 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:

Base class to store results obtained from applying a similarity grouping (clustering) algorithm.

Similarity grouping algorithms are segmentation or machine learning algorithms for partitioning the members of a set of objects (e.g. geometric primitives) into (sub-)groups aka features of different kind/type. A plethora of algorithms exists.

This base class considers metadata and results of having a similarity grouping algorithm applied to a set in which objects are either categorized as noise or belonging to a cluster, i.e. members of a cluster. The algorithm assigns each similarity group (feature/cluster) at least one identifier (numerical or categorical labels) to distinguish different cluster.

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/clusters.

Groups cited:
NXprocess

Structure:

cardinality: (optional) *NX_POSINT* {units=*NX_UNITLESS*}

Number of members in the set which gets 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.

index_offset: (optional) *NX_INT* {units=*NX_UNITLESS*}

Which numerical index is the first to be used to label a feature.

The value should be chosen in such a way that special values can be resolved: * index_offset - 1 indicates that an object belongs to no cluster. * index_offset - 2 indicates that an object belongs to the noise category. Setting for instance index_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_INT* (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 indices_cluster.

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 objects were grouped to which feature/group summary statistics are stored under this group.

unassigned: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of features categorized as unassigned.

noise: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of features categorized as noise.

total: (optional) *NX_UINT* {units=*NX_UNITLESS*}

Total number of features.

indices_cluster: (optional) *NX_INT* (Rank: 1, Dimensions: [n_features]) {units=*NX_UNITLESS*}

Array of numerical identifier of each feature.

member_count: (optional) *NX_UINT* (Rank: 2, Dimensions: [n_features, n_lbl_num]) {units=*NX_UNITLESS*}

Array of number of objects 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/index_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/indices_cluster-field*
- */NXsimilarity_grouping/statistics/member_count-field*
- */NXsimilarity_grouping/statistics/noise-field*
- */NXsimilarity_grouping/statistics/total-field*
- */NXsimilarity_grouping/statistics/unassigned-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsimilarity_grouping.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/shape/size-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/translation-group*
- */NXsnsevent/ENTRY/instrument/DETECTOR/origin/translation/distance-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/pixel_id-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/polar_angle-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/pulse_time-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/total_counts-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/x_pixel_offset-field*
- */NXsnsevent/ENTRY/instrument/DETECTOR/y_pixel_offset-field*

- */NXsnsevent/ENTRY/instrument/DISK_CHOPPER-group*
- */NXsnsevent/ENTRY/instrument/DISK_CHOPPER/distance-field*
- */NXsnsevent/ENTRY/instrument/moderator-group*
- */NXsnsevent/ENTRY/instrument/moderator/coupling_material-field*
- */NXsnsevent/ENTRY/instrument/moderator/distance-field*
- */NXsnsevent/ENTRY/instrument/moderator/temperature-field*
- */NXsnsevent/ENTRY/instrument/moderator/type-field*
- */NXsnsevent/ENTRY/instrument/name-field*
- */NXsnsevent/ENTRY/instrument/POLARIZER-group*
- */NXsnsevent/ENTRY/instrument/SNS-group*
- */NXsnsevent/ENTRY/instrument/SNS/frequency-field*
- */NXsnsevent/ENTRY/instrument/SNS/name-field*
- */NXsnsevent/ENTRY/instrument/SNS/probe-field*
- */NXsnsevent/ENTRY/instrument/SNS/type-field*
- */NXsnsevent/ENTRY/instrument/SNSdetector_calibration_id-field*
- */NXsnsevent/ENTRY/instrument/SNSgeometry_file_name-field*
- */NXsnsevent/ENTRY/instrument/SNStranslation_service-field*
- */NXsnsevent/ENTRY/MONITOR-group*
- */NXsnsevent/ENTRY/MONITOR/data-field*
- */NXsnsevent/ENTRY/MONITOR/distance-field*
- */NXsnsevent/ENTRY/MONITOR/mode-field*
- */NXsnsevent/ENTRY/MONITOR/time_of_flight-field*
- */NXsnsevent/ENTRY/notes-field*
- */NXsnsevent/ENTRY/proton_charge-field*
- */NXsnsevent/ENTRY/raw_frames-field*
- */NXsnsevent/ENTRY/run_number-field*
- */NXsnsevent/ENTRY/sample-group*
- */NXsnsevent/ENTRY/sample/changer_position-field*
- */NXsnsevent/ENTRY/sample/holder-field*
- */NXsnsevent/ENTRY/sample/identifier-field*
- */NXsnsevent/ENTRY/sample/name-field*
- */NXsnsevent/ENTRY/sample/nature-field*
- */NXsnsevent/ENTRY/SNSHistoTool-group*
- */NXsnsevent/ENTRY/SNSHistoTool/author-field*
- */NXsnsevent/ENTRY/SNSHistoTool/command1-field*
- */NXsnsevent/ENTRY/SNSHistoTool/date-field*

- */NXsnsevent/ENTRY/SNSHistoTool/description-field*
- */NXsnsevent/ENTRY/SNSHistoTool/SNSbanking_file_name-field*
- */NXsnsevent/ENTRY/SNSHistoTool/SNSmapping_file_name-field*
- */NXsnsevent/ENTRY/SNSHistoTool/version-field*
- */NXsnsevent/ENTRY/start_time-field*
- */NXsnsevent/ENTRY/title-field*
- */NXsnsevent/ENTRY/total_counts-field*
- */NXsnsevent/ENTRY/total_uncounted_counts-field*
- */NXsnsevent/ENTRY/USER-group*
- */NXsnsevent/ENTRY/USER/facility_user_id-field*
- */NXsnsevent/ENTRY/USER/name-field*
- */NXsnsevent/ENTRY/USER/role-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsnsevent.nxdl.xml

NXsnshisto

Status:

application definition (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/shape/size-field
- /NXsnshisto/ENTRY/instrument/APERTURE/origin/translation-group
- /NXsnshisto/ENTRY/instrument/APERTURE/origin/translation/distance-field
- /NXsnshisto/ENTRY/instrument/APERTURE/x_pixel_offset-field
- /NXsnshisto/ENTRY/instrument/ATTENUATOR-group

- */NXsnshisto/ENTRY/instrument/ATTENUATOR/distance-field*
- */NXsnshisto/ENTRY/instrument/beamline-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL-group*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin-group*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/description-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/orientation-group*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/orientation/value-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape-group*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape/description-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape/shape-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/shape/size-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/translation-group*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/origin/translation/distance-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/type-field*
- */NXsnshisto/ENTRY/instrument/CRYSTAL/wavelength-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR-group*
- */NXsnshisto/ENTRY/instrument/DETECTOR/azimuthal_angle-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/data-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/data_x_time_of_flight-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/data_x_y-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/data_y_time_of_flight-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/distance-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin-group*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/orientation-group*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/orientation/value-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape-group*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape/description-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape/shape-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/shape/size-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/translation-group*
- */NXsnshisto/ENTRY/instrument/DETECTOR/origin/translation/distance-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/pixel_id-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/polar_angle-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/time_of_flight-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/total_counts-field*
- */NXsnshisto/ENTRY/instrument/DETECTOR/x_pixel_offset-field*

- /NXsnshisto/ENTRY/instrument/DETECTOR/y_pixel_offset-field
- /NXsnshisto/ENTRY/instrument/DISK_CHOPPER-group
- /NXsnshisto/ENTRY/instrument/DISK_CHOPPER/distance-field
- /NXsnshisto/ENTRY/instrument/FERMI_CHOPPER-group
- /NXsnshisto/ENTRY/instrument/FERMI_CHOPPER/distance-field
- /NXsnshisto/ENTRY/instrument/moderator-group
- /NXsnshisto/ENTRY/instrument/moderator/coupling_material-field
- /NXsnshisto/ENTRY/instrument/moderator/distance-field
- /NXsnshisto/ENTRY/instrument/moderator/temperature-field
- /NXsnshisto/ENTRY/instrument/moderator/type-field
- /NXsnshisto/ENTRY/instrument/name-field
- /NXsnshisto/ENTRY/instrument/POLARIZER-group
- /NXsnshisto/ENTRY/instrument/SNS-group
- /NXsnshisto/ENTRY/instrument/SNS/frequency-field
- /NXsnshisto/ENTRY/instrument/SNS/name-field
- /NXsnshisto/ENTRY/instrument/SNS/probe-field
- /NXsnshisto/ENTRY/instrument/SNS/type-field
- /NXsnshisto/ENTRY/instrument/SNSdetector_calibration_id-field
- /NXsnshisto/ENTRY/instrument/SNSgeometry_file_name-field
- /NXsnshisto/ENTRY/instrument/SNStranslation_service-field
- /NXsnshisto/ENTRY/MONITOR-group
- /NXsnshisto/ENTRY/MONITOR/data-field
- /NXsnshisto/ENTRY/MONITOR/distance-field
- /NXsnshisto/ENTRY/MONITOR/mode-field
- /NXsnshisto/ENTRY/MONITOR/time_of_flight-field
- /NXsnshisto/ENTRY/notes-field
- /NXsnshisto/ENTRY/proton_charge-field
- /NXsnshisto/ENTRY/raw_frames-field
- /NXsnshisto/ENTRY/run_number-field
- /NXsnshisto/ENTRY/sample-group
- /NXsnshisto/ENTRY/sample/changer_position-field
- /NXsnshisto/ENTRY/sample/holder-field
- /NXsnshisto/ENTRY/sample/identifier-field
- /NXsnshisto/ENTRY/sample/name-field
- /NXsnshisto/ENTRY/sample/nature-field
- /NXsnshisto/ENTRY/SNSHistoTool-group

- */NXsnshisto/ENTRY/SNSHistoTool/author-field*
- */NXsnshisto/ENTRY/SNSHistoTool/command1-field*
- */NXsnshisto/ENTRY/SNSHistoTool/date-field*
- */NXsnshisto/ENTRY/SNSHistoTool/description-field*
- */NXsnshisto/ENTRY/SNSHistoTool/SNSbanking_file_name-field*
- */NXsnshisto/ENTRY/SNSHistoTool/SNSmapping_file_name-field*
- */NXsnshisto/ENTRY/SNSHistoTool/version-field*
- */NXsnshisto/ENTRY/start_time-field*
- */NXsnshisto/ENTRY/title-field*
- */NXsnshisto/ENTRY/total_counts-field*
- */NXsnshisto/ENTRY/total_uncounted_counts-field*
- */NXsnshisto/ENTRY/USER-group*
- */NXsnshisto/ENTRY/USER/facility_user_id-field*
- */NXsnshisto/ENTRY/USER/name-field*
- */NXsnshisto/ENTRY/USER/role-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsnshisto.nxdl.xml

NXsolenoid_magnet

Status:

base class (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.

- S. Ghebi
- L. H. Laidlaw

for an introduction into the topic of modeling shapes with constructive solid geometry (CSG).

Symbols:

No symbol table

Groups cited:
NXcsg, *NXoff_geometry*, *NXquadric*

Structure:

QUADRIC: (optional) *NXquadric*

Instances of *NXquadric* making up elements of the geometry.

OFF_GEOMETRY: (optional) *NXoff_geometry*

Instances of *NXoff_geometry* making up elements of the geometry.

CSG: (optional) *NXcsg*

The geometries defined, made up of e.g. instances of *NXquadric*, *NXoff_geometry*, or instances of other base classes that define geometries.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsolid_geometry/CSG-group*
- */NXsolid_geometry/OFF_GEOMETRY-group*
- */NXsolid_geometry/QUADRIC-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsolid_geometry.nxdl.xml

NXspatial_filter

Status:

base class (contribution), extends *NXparameters*

Description:

Base class for a spatial filter for objects within a region-of-interest (ROI).

Objects can be points, objects composed from other geometric primitives, or objects.

Symbols:

The symbols used in the schema to specify e.g. dimensions of arrays.

n_hexahedra: Number of hexahedra.

n_cylinders: Number of cylinders.

n_ellipsoids: Number of ellipsoids.

n_polyhedra: Number of polyhedra.

Groups cited:
NXcg_cylinder, *NXcg_ellipsoid*, *NXcg_hexahedron*, *NXcg_polyhedron*, *NXcs_filter_boolean_mask*

Structure:

windowing_method: (optional) *NX_CHAR*

Qualitative statement which describes the logical operations that define which objects will be included and which excluded:

- entire_dataset, no filter is applied, all objects are included.
- union_of_primitives, a filter with (possibly non-axis-aligned) geometric primitives. Objects in or on the surface of the primitives are included. All other objects are excluded.
- bitmask, a boolean array whose bits encode with 1 which objects are included. Bits set to zero encode which objects are excluded.

Users of python can use the bitfield operations of the numpy package to work with bitfields. Multiple instances of NXcg base classes are used to compose a union_of_primitives.

Any of these values: `entire_dataset | union_of_primitives | bitmask`

CG_HEXAHEDRON: (optional) *NXcg_hexahedron*

CG_CYLINDER: (optional) *NXcg_cylinder*

CG_ELLIPSOID: (optional) *NXcg_ellipsoid*

CG_POLYHEDRON: (optional) *NXcg_polyhedron*

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-group*
- */NXspatial_filter/CG_ELLIPSOID-group*
- */NXspatial_filter/CG_HEXAHEDRON-group*
- */NXspatial_filter/CG_POLYHEDRON-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:

Base class 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

NXspm

Status:

application definition (contribution), extends *NXsensor_scan*

Description:

Scanning Probe Microscopy (SPM) is a branch of microscopy that utilizes a physical probe to scan the surface of sample and image it at the atomic level.

The application class NXspm is designed as a skeleton and contains common technical concepts for specific SPM sub-techniques such as STM, STS, AFM etc. In addition, it can be utilized to describe the SPM experiments without further specialization for each sub-technique.

Symbols:

No symbol table

Groups cited:

NXamplifier, *NXcalibration*, *NXcollection*, *NXdata*, *NXentry*, *NXenvironment*, *NXfabrication*, *NXhistory*, *NXinstrument*, *NXlockin*, *NXparameters*, *NXpid_controller*, *NXprocess*, *NXrcs*, *NXsample*, *NXsensor*, *NXspm_bias_spectroscopy*, *NXspm_piezo_sensor*, *NXspm_positioner*, *NXspm_scan_control*, *NXspm_scan_pattern*, *NXspm_scan_region*, *NXspm_temperature_sensor*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Name of the definition that is used for the application.

Obligatory value: `NXspm`

experiment_technique: (required) *NX_CHAR*

The technique of the experiment like STM, STS, AFM.

Any of these values: `STM` | `STS` | `AFM`

scan_mode: (optional) *NX_CHAR*

The mode of the scan. The possible options depend on the type of experiment. For example, in STM, the scan mode could be constant height or constant current, in AFM, the scan mode could be contact mode, tapping mode or non-contact mode.

For general purpose usage, all scan modes from its sub-techniques are listed.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `constant height`
- `constant current`
- `contact mode`
- `tapping mode`
- `peak force tapping mode`
- `non-contact mode`

scan_type: (recommended) *NX_CHAR*

The type of the scan. It mainly describes how scan probe moves in the scan region, e.g. forward, backward, or both (if scan is repeated). Any lab defined scan type

identifier_experiment: (recommended) *NX_CHAR* <=

The identifier for the experiment which should be unique at least in lab.

PROCESS: (optional) *NXprocess* <=

Define data processing (e.g., data analysis, image processing) program and associated workflow, software and store results.

INSTRUMENT: (required) *NXinstrument* <=

hardware: (required) *NXfabrication* <=

The hardware description of core instrument setup of experiment. Usually, the entire instrument is supplied by a single manufacturer. To describe the hardware from any sub-components, use the `hardware` group of that sub-component (child group of the NXinstrument group) group.

name: (recommended) *NX_CHAR*

Name of the hardware.

vendor: (required) *NX_CHAR* <=

Company name of the manufacturer.

model: (recommended) *NX_CHAR* <=

Version or model of the hardware setup provided by the manufacturer.

software: (required) *NXfabrication* <=

The software description of core instrument setup of experiment. Usually, the entire instrument is supplied by a single name/manufacturer/model/etc. To describe the software from any sub-components, use the `software` group of that component.

name: (optional) *NX_CHAR*

Name of the software.

vendor: (required) *NX_CHAR* <=

Company name of the manufacturer.

model: (recommended) *NX_CHAR* <=

Version or model of the component named by the manufacturer.

real_time_controller: (recommended) *NXrcs*

lockin_amplifier: (optional) *NXlockin*

The lock-in amplifier information.

flip_sign: (optional) *NX_NUMBER*

The sign (1 or -1) that renders the values of the lock-in current positive. The calibration procedure with retracted tip is normally performed to compensate for the signal phase delay in SPM. The procedure yields two possible solutions corresponding to the chosen phase, this number should be equal to 1 or -1 depending on which solution is chosen (this concept mainly used in STS experiments).

active_channel: (recommended) *NX_CHAR*

The name of the active channel of the lock-in amplifier which is used for the measurement.

SCAN_ENVIRONMENT: (required) *NXenvironment* <=

Information of the scan environment holding concept for temperature, set-point (current or height), scan area and scan data.

Note: At least one field from head_temperature, cryo_bottom_temperature and cryo_shield_temperature must be provided.

head_temperature: (recommended) *NX_NUMBER* {units=*NX_TEMPERATURE*}

Temperature (stabilized or target value) of STM head. For array data of head_temperature, use head_temperature_sensor group.

identifier_environment: (recommended) *NX_CHAR* <=

Unique identifier for the scan environment defined by the user or lab. When multiple scans are performed in a single environment conditions or settings, the entire scan environment can be differentiated by this identifier. For example, scan on a sample of TiSe2 with layered of evaporated pyrene and annealed at 300K temperature for 5 min process.

cryo_bottom_temperature: (recommended) *NX_NUMBER* {units=*NX_TEMPERATURE*}

Temperature (stabilized or targeted single value) of the cold tail of the cryostat. For array data of cryo_bottom_temperature, use cryo_bottom_temperature_sensor group.

cryo_shield_temperature: (recommended) *NX_NUMBER* {units=*NX_TEMPERATURE*}

Temperature (stabilized or targeted single value) of liquid nitrogen shield. For array data of cryo_shield_temperature, use cryo_shield_temperature_sensor group.

current_sensorTAG: (optional) *NXsensor* <=

This should be a link to /entry/instrument/current_sensorTAG

voltage_sensorTAG: (optional) *NXsensor* <=

This should be a link to /entry/instrument/voltage_sensorTAG

piezo_sensor: (optional) *NXspm_piezo_sensor*

This should be a link to /entry/instrument/piezo_sensor.

XYZ**piezo_sensor**: (optional) *NXspm_piezo_sensor*

This should be a link to /entry/instrument/XYZpiezo_sensor.

height_piezo_sensor: (optional) *NXspm_piezo_sensor*

This should be a link to the concept entry/instrument/height_piezo_sensor.

SPM_SCAN_CONTROL: (required) *NXspm_scan_control*

The scan control information like scan region or phase space, type of scan (e.g. mesh, spiral, etc.), and scan speed, etc. This group mainly stores the scan settings data. For processed data or final experimental data would go to NXdata group.

scanTAG: (recommended) *NX_CHAR*

If there are multiple scans performed under the same environment, use this field to differentiate among them.

scan_region: (required) *NXspm_scan_region* <=

The scan region (phase space or sub-phase space) is the region where the scan is performed.

scan_range_x: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The range of the scan in x direction.

scan_range_y: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The range of the scan in y direction.

scan_offset_value_x: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The offset of the scan in x direction.

scan_offset_value_y: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The offset of the scan in y direction.

scan_angle_x: (recommended) *NX_NUMBER* {units=*NX_ANGLE*} <=

The angle of the scan region in x direction.

scan_angle_y: (recommended) *NX_NUMBER* {units=*NX_ANGLE*} <=

The angle of the scan region in y direction.

scan_start_x: (required) *NX_NUMBER* {units=*NX_LENGTH*} <=

The start of the scan in x direction.

scan_start_y: (required) *NX_NUMBER* {units=*NX_LENGTH*} <=

The start of the scan in y direction.

scan_end_x: (required) *NX_NUMBER* {units=*NX_LENGTH*} <=

The end of the scan in x direction.

scan_end_y: (required) *NX_NUMBER* {units=*NX_LENGTH*} <=

The end of the scan in y direction.

meshSCAN: (required) *NXspm_scan_pattern* <=

The mesh scan is a common technique used in SPM to scan the surface of the sample in a grid pattern.

scan_points_x: (required) *NX_NUMBER* <=

The number of points scanned in x direction.

scan_points_y: (required) *NX_NUMBER* <=

The number of points scanned in y direction.

step_size_x: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The step size in x direction.

step_size_y: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The step size in y direction.

current_sensorTAG: (optional) *NXsensor* <=

Information for current sensor. Any current sensor such as a current-voltage transimpedance amplifier involved in the experiment or in any special measurement or in any specialized experiment component can be registered under this group.

NAME**current**: (optional) *NX_CHAR*

The name of the current sensor used for specific measurement or in component.

current: (required) *NX_NUMBER* {units=*NX_CURRENT*}

Name of the current according to the purpose of the measurement. E.g., the field can be named as tip_current defining the current measured at the tip.

offset_value: (optional) *NX_NUMBER* {units=*NX_CURRENT*}

The offset in the tunneling current between tip and sample.

calibration: (optional) *NXcalibration*

Calibration data of the current sensor.

calibration_parameters: (optional) *NXparameters* <=

coefficient: (optional) *NX_NUMBER* {units=*NX_ANY*}

The coefficient of the calibration.

AMPLIFIER: (optional) *NXamplifier*

An amplifier information that amplifies the input signal.

current_gain: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

The gain of the current sensor.

voltage_sensorTAG: (optional) *NXsensor* <=

The sensor information for the voltage device. Any voltage sensor involved in the experiment or in any special measurement or in any specialized experiment component can be registered under this group.

For this purpose, replace the TAG with the specific name or ID of the voltage sensor.

Do not register this group for sample bias voltage (DC), for that we have *sample_bias_voltage* group.

NAME**voltage**: (optional) *NX_CHAR*

The name of the voltage according to the purpose of the measurement. E.g., the field can be named as x_source_drain_bias_voltage defining the applied bias along the x-axis as source-drain channel, while current (tip current) will be measured along the z-axis.

voltage: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

Voltage measured by sensor.

offset_value: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

The offset voltage. The real voltage is the sum of the voltage and the offset voltage.

calibration: (optional) *NXcalibration*

Calibration data of the voltage sensor.

calibration_parameters: (optional) *NXparameters* <=

coefficient: (optional) *NX_NUMBER* {units=*NX_ANY*}

The coefficient of the calibration.

AMPLIFIER: (optional) *NXamplifier*

An amplifier information that amplifies the input signal.

voltage_gain: (optional) *NX_NUMBER* {units=*NX_UNITLESS*}

The gain of the voltage sensor.

piezo_sensor: (optional) *NXspm_piezo_sensor*

This piezo sensor group refers to the XYZ (in all directions) piezo sensor.

x: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The x position of the piezo.

y: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The y position of the piezo.

z: (recommended) *NX_NUMBER* {units=*NX_LENGTH*} <=

The z position of the piezo.

height_piezo_sensor: (recommended) *NXspm_piezo_sensor*

The sensor information for the height (Z-axis) piezo device.

XYZ**piezo_sensor**: (optional) *NXspm_piezo_sensor*

For separate piezo sensors along X-, Y-, and Z-axis register this group, e.g., x_piezo_sensor or height_piezo_sensor.

head_temperature_sensor: (recommended) *NXspm_temperature_sensor*

The temperature of the SPM measured near the sample or tip. This group is intended to be used for an array data (e.g., time series).

cryo_bottom_temperature_sensor: (optional) *NXspm_temperature_sensor*

The temperature of the cold tail of the cryostat. This group is intended to be used for an array data (e.g., time series).

cryo_shield_temperature_sensor: (optional) *NXspm_temperature_sensor*

The temperature of the liquid nitrogen shield. This group is intended to be used for an array data (e.g., time series).

sample_temperature_sensor: (optional) *NXspm_temperature_sensor*

The temperature of the sample. This group is intended to be used for an array data (e.g., time series).

bias_spectroscopy_environment: (optional) *NXenvironment* <=

To explain bias and current behavior (sweep measurement especially in STS experiment) due to voltage applied to the sample.

In some experiments, e.g., STM, bias spectroscopy could also be part of the measurement setup. SPM_BIAS_SPECTROSCOPY: (required) *NXspm_bias_spectroscopy*

Setup and scan data for continuous measurement of bias voltage on the subject of experiment vs tunneling current from probe. SPM_POSITIONER: (recommended) *NXspm_positioner* <=

Information about positioner used in STS scan settings, like PID controller and offset.

Reuse, if needed, this group for positioners for multiple scans at different points on sample. **z_controller**: (recommended) *NXpid_controller* <=

BIAS_SWEEP: (required) *NXspm_scan_control* <=

The bias voltage sweep is a common technique used to study properties (in this case current) in the sample or environment due to change in applied bias voltage. **scan_region**: (required) *NXspm_scan_region* <=

The scan region (phase space or sub-phase space) is the region where the scan is performed.

scan_start_bias: (required) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

scan_end_bias: (required) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

scan_offset_bias: (required) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

The offset of the bias scan voltage.

scan_range_bias: (recommended) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

linear_sweep: (required) *NXspm_scan_pattern* <=

The linear sweep is a common technique used on the substance or sample or environment to study the change in the behavior of the sample or substance or environment due to change in applied bias voltage.

scan_points_bias: (required) *NX_NUMBER* <=

step_size_bias: (required) *NX_NUMBER* <=

sample_bias_voltage: (optional) *NXsensor* <=

The DC bias voltage that is applied to the sample (for example in constant-current mode in STM).

bias_voltage: (required) *NX_NUMBER* {units=*NX_VOLTAGE*}

The bias voltage (DC) applied to the sample.

bias_offset_value: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

Offset value of the bias voltage.

calibration: (optional) *NXcalibration*

Calibration of the bias voltage measurement (V/V).

calibration_parameters: (optional) *NXparameters* <=

coefficient: (optional) *NX_NUMBER* {units=*NX_ANY*}

The coefficient of the calibration.

SAMPLE: (optional) *NXsample* <=

The sample information.

history: (recommended) *NXhistory* <=

A set of physical processes that occurred to the sample prior/during experiment.

sample_environment: (optional) *NXenvironment* <=

Information of environment around the sample.

temperature: (recommended) *NX_NUMBER* {units=*NX_TEMPERATURE*}

The single-valued temperature of the sample, also referred to as the tip temperature (not head temperature), since the tip and sample are in contact or in close proximity. For array like temperature data use sample_temperature_sensor group.

sample_bias_voltage: (optional) *NXsensor* <=

This should be a link to

/entry/instrument/sample_bias_voltage sensor under instrument.

sample_temperature_sensor: (optional) *NXspm_temperature_sensor*

This should be a link to

/entry/instrument/sample_temperature_sensor under instrument.

DATA: (required) *NXdata* <=

The data group.

DATA: (required) *NX_NUMBER* {units=*NX_ANY*} <=

The data (e.g. current, voltage, temperature) field that can be plotted against the axes.

AXISNAME: (required) *NX_CHAR_OR_NUMBER* {units=*NX_ANY*} <=

The name of the axis that corresponds to the data field.

reproducibility_indicators: (optional) *NXcollection* <=

The group of indicators (links to the existing fields in different groups) that measure the reproducibility of the experiment.

LINK_TO_FIELD: (optional) *NX_CHAR*

A place holder to create link to any field relevant considered as reproducibility indicators (defined by laboratory).

LINK_TO_GROUP: (optional) *NXcollection* <=

A place holder to create link to any group relevant considered as reproducibility indicators (defined by laboratory).

resolution_indicators: (optional) *NXcollection* <=

The group of indicators (links to the existing fields in different groups) that are used to measure the resolution of the experiment results.

LINK_TO_FIELD: (optional) *NX_CHAR*

A place holder to create link to any field relevant considered as reproducibility indicators (defined by laboratory).

LINK_TO_GROUP: (optional) *NXcollection* <=

A place holder to create link to any group relevant considered as reproducibility indicators (defined by laboratory).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm/ENTRY-group*
- */NXspm/ENTRY/DATA-group*
- */NXspm/ENTRY/DATA/AXISNAME-field*
- */NXspm/ENTRY/DATA/DATA-field*
- */NXspm/ENTRY/definition-field*
- */NXspm/ENTRY/experiment_technique-field*
- */NXspm/ENTRY/identifier_experiment-field*
- */NXspm/ENTRY/INSTRUMENT-group*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment-group*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY-group*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP-group*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/linear_sweep-group*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/linear_sweep/sca field*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/linear_sweep/step field*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/scan_region-group*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/scan_region/scan field*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/scan_region/scan field*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/scan_region/scan field*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/BIAS_SWEEP/scan_region/scan field*
- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/SPM_POSITIONER-group*

- */NXspm/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/SPM_POSITIONER/z_controller-group*
- */NXspm/ENTRY/INSTRUMENT/cryo_bottom_temperature_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/cryo_shield_temperature_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG-group*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/AMPLIFIER-group*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/AMPLIFIER/current_gain-field*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/calibration-group*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/calibration/calibration_parameters-group*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/calibration/calibration_parameters/coefficient-field*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/current-field*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/NAMEcurrent-field*
- */NXspm/ENTRY/INSTRUMENT/current_sensorTAG/offset_value-field*
- */NXspm/ENTRY/INSTRUMENT/hardware-group*
- */NXspm/ENTRY/INSTRUMENT/hardware/model-field*
- */NXspm/ENTRY/INSTRUMENT/hardware/name-field*
- */NXspm/ENTRY/INSTRUMENT/hardware/vendor-field*
- */NXspm/ENTRY/INSTRUMENT/head_temperature_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/height_piezo_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/lockin_amplifier-group*
- */NXspm/ENTRY/INSTRUMENT/lockin_amplifier/active_channel-field*
- */NXspm/ENTRY/INSTRUMENT/lockin_amplifier/flip_sign-field*
- */NXspm/ENTRY/INSTRUMENT/piezo_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/piezo_sensor/x-field*
- */NXspm/ENTRY/INSTRUMENT/piezo_sensor/y-field*
- */NXspm/ENTRY/INSTRUMENT/piezo_sensor/z-field*
- */NXspm/ENTRY/INSTRUMENT/real_time_controller-group*
- */NXspm/ENTRY/INSTRUMENT/sample_bias_voltage-group*
- */NXspm/ENTRY/INSTRUMENT/sample_bias_voltage/bias_offset_value-field*
- */NXspm/ENTRY/INSTRUMENT/sample_bias_voltage/bias_voltage-field*
- */NXspm/ENTRY/INSTRUMENT/sample_bias_voltage/calibration-group*
- */NXspm/ENTRY/INSTRUMENT/sample_bias_voltage/calibration/calibration_parameters-group*
- */NXspm/ENTRY/INSTRUMENT/sample_bias_voltage/calibration/calibration_parameters/coefficient-field*
- */NXspm/ENTRY/INSTRUMENT/sample_temperature_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/cryo_bottom_temperature-field*

- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/cryo_shield_temperature-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/current_sensorTAG-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/head_temperature-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/height_piezo_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/identifier_environment-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/piezo_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/meshSCAN-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/meshSCAN/scan_points_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/meshSCAN/scan_points_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/meshSCAN/step_size_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/meshSCAN/step_size_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_angle_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_angle_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_end_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_end_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_offset_value_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_offset_value_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_range_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_range_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_start_x-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scan_region/scan_start_y-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL/scanTAG-field*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/voltage_sensorTAG-group*
- */NXspm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/XYZpiezo_sensor-group*
- */NXspm/ENTRY/INSTRUMENT/software-group*

- */NXspm/ENTRY/INSTRUMENT/software/model-field*
- */NXspm/ENTRY/INSTRUMENT/software/name-field*
- */NXspm/ENTRY/INSTRUMENT/software/vendor-field*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG-group*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/AMPLIFIER-group*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/AMPLIFIER/voltage_gain-field*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/calibration-group*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/calibration/calibration_parameters-group*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/calibration/calibration_parameters/coefficient-field*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/NAMEvoltage-field*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/offset_value-field*
- */NXspm/ENTRY/INSTRUMENT/voltage_sensorTAG/voltage-field*
- */NXspm/ENTRY/INSTRUMENT/XYZpiezo_sensor-group*
- */NXspm/ENTRY/PROCESS-group*
- */NXspm/ENTRY/reproducibility_indicators-group*
- */NXspm/ENTRY/reproducibility_indicators/LINK_TO_FIELD-field*
- */NXspm/ENTRY/reproducibility_indicators/LINK_TO_GROUP-group*
- */NXspm/ENTRY/resolution_indicators-group*
- */NXspm/ENTRY/resolution_indicators/LINK_TO_FIELD-field*
- */NXspm/ENTRY/resolution_indicators/LINK_TO_GROUP-group*
- */NXspm/ENTRY/SAMPLE-group*
- */NXspm/ENTRY/SAMPLE/history-group*
- */NXspm/ENTRY/SAMPLE/sample_environment-group*
- */NXspm/ENTRY/SAMPLE/sample_environment/sample_bias_voltage-group*
- */NXspm/ENTRY/SAMPLE/sample_environment/sample_temperature_sensor-group*
- */NXspm/ENTRY/SAMPLE/sample_environment/temperature-field*
- */NXspm/ENTRY/scan_mode-field*
- */NXspm/ENTRY/scan_type-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm.nxdl.xml

NXspm_bias_spectroscopy

Status:

base class (contribution), extends *NXobject*

Description:

A base class for bias spectroscopy to describe the change in the physical properties of the sample with respect to the sweep voltage applied on a sample of STM/AFM/… experiments.

In these experiments an electric potential is applied between the (conductive) sample and the probe (tip), and the physical properties (e.g. tunnelling current) are measured as the function of this potential. The potential is varied in so-called voltage sweeps and the corresponding properties are recorded accordingly.

Symbols:

No symbol table

Groups cited:

NXcircuit, *NXcoordinate_system*, *NXdata*, *NXpid_controller*, *NXspm_positioner*, *NXspm_scan_control*, *NXspm_scan_pattern*, *NXspm_scan_region*

Structure:

measurement_type: (optional) *NX_CHAR*

The measurement type defines how current is measured under the different input variables like bias voltage (constant_spacing) or height (variadic_spacing) is applied during the measurement.

Any of these values: `constant_spacing` | `variadic_spacing`

identifier_environment: (recommended) *NX_CHAR* <=

Unique identifier for the environment defined by the user or lab. When multiple scans are performed in a single environment conditions or settings, the entire scan environment can be differentiated by this identifier. For example, scan on a sample of TiSe2 with layered of evaporated pyrene and annealed at 300K temperature for 5 min process.

SPM_POSITIONER: (optional) *NXspm_positioner*

Information about the positioner PID (proportional, integral, differential feedback system), offset values, setpoint values and so on, while running bias voltage-tunneling current measurement.

These components position the probe relative to the sample, thus help obtaining maps of the data across the sample surface. **z_controller**: (optional) *NXpid_controller* <=

CIRCUIT: (optional) *NXcircuit*

acquisition_time: (optional) *NX_NUMBER* {units=*NX_TIME*} <=

The time or period (average) to acquire the data for a single bias sweep point.

animation_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time or period a bias sweep to be displayed.

measurement_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time or period taken by the circuit to measure a full bias sweep (duration of the voltage-current measurement measurement).

BIAS_SWEEP: (optional) *NXspm_scan_control*

The bias sweep scan which is is performed in the scanning probe microscopy experiments.

scan_type: (optional) *NX_CHAR* <=

This combines not only how the voltages are changed, but how the voltage values are correlated to a position across the sample surface, measuring sweeps are each spatial coordinate or mapping the response at constant voltage, etc. For STS experiment, the scan type is usually a single-point scan (trajectory scan).

Obligatory value: `linear`

number_of_sweeps: (optional) *NX_NUMBER*

The number of sweeps (a full scan from starting bias to end bias) taken during the bias spectroscopy.

first_settling_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The initial time taken to settle the bias voltage at the desired value. On each sweep usually, the system takes time to settle to the bias voltage at the next value.

end_settling_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time (at the last sweep) to settle for the last value of the sweep.

settling_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time taken to settle the bias voltage at the desired value. On each sweep usually, the system takes time to settle the bias voltage at the next value.

max_slew_rate: (optional) *NX_NUMBER* {units=*NX_ANY*}

The rate at which the amplifier responds to the voltage change (to reach at the desired value). It defines if the tip movement and voltage sweep are synchronized.

final_z: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The z position after the sweeps are done.

total_spectroscopy_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The total time needed for the entire voltage sweep.

spatial_location: (optional) *NXcoordinate_system*

The spatial location of the scan points. This would be recommended to use if there are multiple bias sweep scans at different locations.

x: (optional) *NX_NUMBER* {units=*NX_LENGTH*} <=

y: (optional) *NX_NUMBER* {units=*NX_LENGTH*} <=

z: (optional) *NX_NUMBER* {units=*NX_LENGTH*} <=

scan_region: (optional) *NXspm_scan_region* <=

The scan region is the area of phase space or sub-phase space where the scan is performed.

scan_offset_bias: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

The starting voltage of the bias sweep. The range of voltages for the sweep can be defined with scan voltage offset and scan voltage range (difference between minimum and maximum voltage values in a sweep)

scan_range_bias: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

The range of voltages for the sweep can be defined with scan voltage offset and scan voltage range (difference between minimum and maximum voltage values in a sweep)

scan_start_bias: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

The start of the bias scan voltage.

scan_end_bias: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

The end value of the bias scan voltage.

linear_sweep: (optional) *NXspm_scan_pattern* <=

In the linear sweep, the bias voltage is changed linearly from the start value to the end value.

backward_sweep: (optional) *NX_BOOLEAN*

If the bias voltage sweep is also performed in the opposite direction.

scan_points_bias: (optional) *NX_NUMBER* <=

The number of voltage points per sweep.

step_size_bias: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*} <=

The step size between the two consecutive bias voltage values during the sweep.

scan_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The time taken by the scanner to scan the entire area.

reset_bias: (optional) *NX_BOOLEAN*

The reset_bias defines whether the bias voltage should be reset to the starting value after the sweep is completed.

DATA: (optional) *NXdata* <=

The scan data is the data collected during the scan. If the scan has several channels or derivatives from the channel data, please duplicate this NXdata group for each.

To define specific current or voltage use *DATA* and *AXISNAME*.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_bias_spectroscopy/BIAS_SWEEP-group*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/end_settling_time-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/final_z-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/first_settling_time-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep-group*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep/backward_sweep-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep/DATA-group*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep/reset_bias-field*

- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep/scan_points_bias-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep/scan_time-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/linear_sweep/step_size_bias-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/max_slew_rate-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/number_of_sweeps-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/scan_region-group*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/scan_region/scan_end_bias-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/scan_region/scan_offset_bias-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/scan_region/scan_range_bias-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/scan_region/scan_start_bias-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/scan_type-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/settling_time-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/spatial_location-group*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/spatial_location/x-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/spatial_location/y-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/spatial_location/z-field*
- */NXspm_bias_spectroscopy/BIAS_SWEEP/total_spectroscopy_time-field*
- */NXspm_bias_spectroscopy/CIRCUIT-group*
- */NXspm_bias_spectroscopy/CIRCUIT/acquisition_time-field*
- */NXspm_bias_spectroscopy/CIRCUIT/animation_time-field*
- */NXspm_bias_spectroscopy/CIRCUIT/measurement_time-field*
- */NXspm_bias_spectroscopy/identifier_environment-field*
- */NXspm_bias_spectroscopy/measurement_type-field*
- */NXspm_bias_spectroscopy/SPM_POSITIONER-group*
- */NXspm_bias_spectroscopy/SPM_POSITIONER/z_controller-group*

NXDL Source:

https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_bias_spectroscopy.nxdl.xml

NXspm_cantilever

Status:

base class (contribution), extends *NXobject*

Description:

A base class to describe the cantilever used in Atomic Force Microscopy (AFM).

Symbols:

No symbol table

Groups cited:
NXpositioner, *NXspm_cantilever_config*, *NXspm_cantilever_oscillator*

Structure:

cantilever_oscillator: (optional) *NXspm_cantilever_oscillator*

The oscillator of the cantilever.

cantilever_config: (optional) *NXspm_cantilever_config*

The configuration parameters of the cantilever used in scanning probe microscopy.

phase_positioner: (optional) *NXpositioner*

The phase positioner of the cantilever.

amplitude_positioner: (optional) *NXpositioner*

The amplitude positioner of the cantilever.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_cantilever/amplitude_positioner-group*
- */NXspm_cantilever/cantilever_config-group*
- */NXspm_cantilever/cantilever_oscillator-group*
- */NXspm_cantilever/phase_positioner-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_cantilever.nxdl.xml

NXspm_cantilever_config

Status:

base class (contribution), extends *NXobject*

Description:

This file defines the NXspm_cantilever_config base class, which contains configuration information about the cantilever used in the AFM (atomic force microscopy) experiments.

Symbols:

No symbol table

Groups cited:
NXcalibration

Structure:

cantilever_coating: (optional) *NX_CHAR*

The coating material of the cantilever.

curvature_radiusN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The radius of curvature of the cantilever tip. The (substring) N denotes X or Y.

cantilever_type: (optional) *NX_CHAR*

The shape of the cantilever as a general text, such as A-shape, beam, or arrow.

cantilever_length: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Nominal length between base and end of the cantilever in micrometers.

cantilever_width: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Nominal width of the cantilever in microns.

cantilever_thickness: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Nominal thickness of the cantilever in microns.

resonance_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

Nominal free resonance frequency of the cantilever in air i.e., out of interaction force.

amplitude_excitation: (optional) *NX_NUMBER* {units=*NX_ANY*}

Either the drive amplitude in mv for the driving cantilever or the free-oscillation amplitude which is the resulted movement amplitude without interaction to the surface.

fermi_level: (optional) *NX_NUMBER* {units=*NX_VOLTAGE*}

The Fermi level of the cantilever material.

resonance_amplitude: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Nominal free resonance amplitude of the cantilever in air, in nm i.e., out of interaction force.

spring_constant: (optional) *NX_NUMBER* {units=*NX_ANY*}

The spring constant coefficient of the cantilever.

CALIBRATION: (optional) *NXcalibration*

The calibration information of the cantilever.

sensitivity: (optional) *NX_NUMBER* {units=*NX_ANY*}

A force applied to the cantilever tip will cause a change in cantilever's oscillation amplitude (in dynamic mode) or deflection of the cantilever. The sensitivity of the cantilever is calculated as the ratio of this change to the force causing it.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_cantilever_config/amplitude_excitation-field*
- */NXspm_cantilever_config/CALIBRATION-group*
- */NXspm_cantilever_config/CALIBRATION/sensitivity-field*
- */NXspm_cantilever_config/cantilever_coating-field*
- */NXspm_cantilever_config/cantilever_length-field*
- */NXspm_cantilever_config/cantilever_thickness-field*
- */NXspm_cantilever_config/cantilever_type-field*
- */NXspm_cantilever_config/cantilever_width-field*
- */NXspm_cantilever_config/curvature_radiusN-field*
- */NXspm_cantilever_config/fermi_level-field*

- */NXspm_cantilever_config/resonance_amplitude-field*
- */NXspm_cantilever_config/resonance_frequency-field*
- */NXspm_cantilever_config/spring_constant-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_cantilever_config.nxdl.xml

NXspm_cantilever_oscillator

Status:

base class (contribution), extends *NXobject*

Description:

In generally speaking a cantilever resembles a leaf-spring which can be treated as a harmonic oscillator as a first approximation.

Note: If any field data in this group comes in an array as input or output in the scan process they will be stored in NXdata in scan_control group.

Symbols:

No symbol table

Groups cited:
none

Structure:

reference_amplitude: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The reference amplitude (also called drive amplitude) of the cantilever. This is the amplitude of the cantilever oscillation when no external forces are acting on it.

Note: At least one from reference_amplitude, reference_frequency, or reference_phase is expected.

reference_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The reference frequency (also called drive frequency or resonance frequency) of the cantilever.

Note: At least one from reference_amplitude, reference_frequency, or reference_phase is expected.

reference_phase: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The reference phase of the cantilever oscillator.

Note: At least one from reference_amplitude, reference_frequency, or reference_phase is expected.

frequency_harmonic: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

The harmonic (e.g., second harmonic of the fundamental frequency) frequency of the cantilever.

phase_shift: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The phase difference between the reference signal of cantilever and response signal.

frequency_shift: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

Shift in the resonance frequency of the cantilever.

frequency_cutoff: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The cutoff frequency of the cantilever.

frequency_bandwidth: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The bandwidth of the resonance frequency.

target_amplitude: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The target amplitude of the cantilever to scan on each scan point. This field is same as the reference amplitude in the non-contact mode.

target_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The target frequency of the cantilever to scan on each scan point. This field is same as the reference frequency in the non-contact mode

active_frequency: (optional) *NX_NUMBER* {units=*NX_FREQUENCY*}

The active frequency of the cantilever to start the experiment.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_cantilever_oscillator/active_frequency-field*
- */NXspm_cantilever_oscillator/frequency_bandwidth-field*
- */NXspm_cantilever_oscillator/frequency_cutoff-field*
- */NXspm_cantilever_oscillator/frequency_harmonic-field*
- */NXspm_cantilever_oscillator/frequency_shift-field*
- */NXspm_cantilever_oscillator/phase_shift-field*
- */NXspm_cantilever_oscillator/reference_amplitude-field*
- */NXspm_cantilever_oscillator/reference_frequency-field*
- */NXspm_cantilever_oscillator/reference_phase-field*
- */NXspm_cantilever_oscillator/target_amplitude-field*
- */NXspm_cantilever_oscillator/target_frequency-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_cantilever_oscillator.nxdl.xml

NXspm_piezo_config

Status:

base class (contribution), extends *NXobject*

Description:

A base class describing piezo actuator settings for scanning probe microscopy.

Piezoelectric actuators work utilizing the inverse-piezoelectric effect, when a voltage is applied on the material and it deforms proportional to the applied voltage. Description below shows calibration coefficients

and other configuration parameters of open loop piezo actuators (that is actuators without capacitive sensor feedback systems).

Symbols:

No symbol table

Groups cited:
NXcalibration, *NXparameters*, *NXspm_piezoelectric_material*

Structure:

piezo_material: (optional) *NXspm_piezoelectric_material*

The material description and properties of the piezoelectric scanner materials.

curvature_radiusN: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The N (substring) denotes X or Y. There are 2 parameters in X and Y directions. It can be set approximately to the length of the piezo tube along X and Y axis.

calibration: (optional) *NXcalibration*

calibration_type: (optional) *NX_CHAR*

The name of the calibration type, sometimes it is called *active calibration*.

Any of these values: `active` | `passive`

calibration_name: (optional) *NX_CHAR*

A specific name of the calibration (e.g. active type with name 'LHe').

calibration_date: (optional) *NX_DATE_TIME*

The date of the calibration.

calibratedAXIS: (optional) *NX_NUMBER*

The AXIS (substring) denotes X, Y or Z, e.g., calibrated_x. There are three directions X, Y, and Z for calibration, along with three available parameters each: Calibration (m/V), Range (m), and HV gain. Only two of these parameters are required to define the calibration. Consequently, when any value is changed, one of the other values will be automatically updated.

hv_gainN: (optional) *NX_NUMBER*

The N (substring) denotes X or Y or Z, e.g., hv_gain_x. In some systems, there is an HV gain readout feature. For these systems, the HV gain should be automatically adjusted whenever the gain is changed at the high voltage amplifier.

rangeN: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The N (substring) denotes X or Y or Z, e.g., range_x. There are 3 parameters in X, Y and Z directions. The range is the maximum distance the piezo can move.

tiltN: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The N (substring) denotes X and Y directions (e.g., tilt_x), and for both directions tilt needs to be adjusted according to the actual surface.

drift_correction_status: (optional) *NX_BOOLEAN*

The drift correction status (true / false) in calibration step of piezo.

driftN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The N (substring) denotes X, Y and Z directions (e.g., drift_x). Define the drift speed [m/s] for all three axes. When the compensation is on, the piezo will start to move at that speed.

calibration_parameters: (optional) *NXparameters* <=

coefficientN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The calibration coefficient is the ratio of the actual distance moved by the piezo due to the voltage or external force applied to the piezo. It is also called first-order correction. The N (substring) denotes X and Y directions (e.g., coefficient_x).

second_order_correctionN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The N (substring) denotes X and Y directions (e.g., second_order_correction_x). If you know them, you can enter the 2nd order piezo characteristics to compensate the error for that axis. The following equation shows the interpretation of the 2nd order correction parameters, For the X-piezo: $U_x = \frac{1}{c_x} \cdot X + c_{xx} \cdot X^2$ with units: $[V] = \frac{[V]}{[m]} \cdot [m] + \frac{[V]}{[m^2]} \cdot [m^2]$ where cx is the calibration of the piezo X and cxx is the 2nd order correction. The unit for the second-order correction is ($\frac{V}{m^2}$).

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_piezo_config/calibration-group*
- */NXspm_piezo_config/calibration/calibratedAXIS-field*
- */NXspm_piezo_config/calibration/calibration_date-field*
- */NXspm_piezo_config/calibration/calibration_name-field*
- */NXspm_piezo_config/calibration/calibration_parameters-group*
- */NXspm_piezo_config/calibration/calibration_parameters/coefficientN-field*
- */NXspm_piezo_config/calibration/calibration_parameters/second_order_correctionN-field*
- */NXspm_piezo_config/calibration/calibration_type-field*
- */NXspm_piezo_config/calibration/drift_correction_status-field*
- */NXspm_piezo_config/calibration/driftN-field*
- */NXspm_piezo_config/calibration/hv_gainN-field*
- */NXspm_piezo_config/calibration/rangeN-field*
- */NXspm_piezo_config/calibration/tiltN-field*
- */NXspm_piezo_config/piezo_material-group*
- */NXspm_piezo_config/piezo_material/curvature_radiusN-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_piezo_config.nxdl.xml

NXspm_piezo_sensor

Status:

base class (contribution), extends *NXsensor*

Description:

This piezo sensor group refers to the height (or Z) piezo sensor.

Symbols:

No symbol table

Groups cited:
NXspm_piezo_config, *NXspm_piezoelectric_material*, *NXspm_positioner*

Structure:

x: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The x position (e.g., target or averaged target value) of the piezo.

y: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The y position (e.g., target or averaged target value) of the piezo.

z: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The z position (e.g., target or averaged target value) of the piezo.

AXIS**offset_value**: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The offset value for the piezo axis (X, Y, or Z) that will be added to the measured value.

piezo_configuration: (optional) *NXspm_piezo_config*

The piezo configuration information like piezoelectric calibration and material properties.

SPM_POSITIONER: (optional) *NXspm_positioner*

The positioner information like the position of the tip, the position of the sample, PID controller etc.

piezo_material: (optional) *NXspm_piezoelectric_material*

The material description and properties of the piezoelectric scanner materials.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_piezo_sensor/AXISoffset_value-field*
- */NXspm_piezo_sensor/piezo_configuration-group*
- */NXspm_piezo_sensor/piezo_material-group*
- */NXspm_piezo_sensor/SPM_POSITIONER-group*
- */NXspm_piezo_sensor/x-field*
- */NXspm_piezo_sensor/y-field*
- */NXspm_piezo_sensor/z-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_piezo_sensor.nxdl.xml

NXspm_piezoelectric_material

Status:

base class (contribution), extends *NXcrystal*

Description:

Description and properties of the piezoelectric actuator materials. The piezoelectric actuator is usually composed of polycrystalline solids and attached at the head of the tip or cantilever. The material deforms when the external electric field (e.g., electric force) is applied.

Symbols:

No symbol table

Groups cited:
none

Structure:

name: (optional) *NX_CHAR* <=

The name of the material of the piezo scanner such as Lead Zirconate Titanates (PZTs).

identifier_piezo_material: (optional) *NX_CHAR* <=

The identifier of the piezo material.

chemical_formula: (optional) *NX_CHAR* <=

type: (optional) *NX_CHAR* <=

The type of the material of the piezo scanner (e.g. piezoelectric ceramics, polymer-film piezoelectrics).

density: (optional) *NX_NUMBER* {units=*NX_ANY*} <=

The density of the piezo material.

relative_permittivity: (optional) *NX_NUMBER* {units=*NX_ANY*}

The relative permittivity (dielectric constant) of the piezo material.

D_**piezoelectric_constant**: (optional) *NX_NUMBER* {units=*NX_ANY*}

The D_piezoelectric_constant (with substitutable part D) piezoelectric charge coefficients of the material. The coefficients describe the electric polarization generated by the applied stress on the material. Different coefficients correspond to different relative directions of the polarization and the stress (e.g., d33, d31).

G_**voltage_constant**: (optional) *NX_NUMBER* {units=*NX_ANY*}

The constants (with substitutable part G) define the electric field produced by the external mechanical strain. Different coefficients correspond to different relative directions of the electric field and the strain (e.g., g33, g31).

K_electromechanical_constant: (optional) *NX_NUMBER* {units=*NX_ANY*}

The electromechanical constant measures the efficiency of the conversion of mechanical energy into electrical energy.

P_pyroelectric_constant: (optional) *NX_NUMBER* {units=*NX_ANY*}

The pyroelectric constant defines the change of the polarization vector of the piezoelectric material per unit change in temperature.

acoustic_impedance: (optional) *NX_NUMBER* {units=*NX_ANY*}

The acoustic impedance of the piezo material.

young_modulus: (optional) *NX_NUMBER* {units=*NX_ANY*}

The Young's modulus of the piezo material.

surface_resistivity: (optional) *NX_NUMBER* {units=*NX_ANY*}

The surface resistivity of the piezo material.

temperature_range: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*}

The temperature range of the piezo material. This field is expecting a range of temperatures in an array [min temperature, max temperature].

glass_transition_temperature: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*}

The range of temperature where a piezoelectric hard material transforms into the viscous state.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_piezoelectric_material/acoustic_impedance-field*
- */NXspm_piezoelectric_material/chemical_formula-field*
- */NXspm_piezoelectric_material/D_piezoelectric_constant-field*
- */NXspm_piezoelectric_material/density-field*
- */NXspm_piezoelectric_material/G_voltage_constant-field*
- */NXspm_piezoelectric_material/glass_transition_temperature-field*
- */NXspm_piezoelectric_material/identifier_piezo_material-field*
- */NXspm_piezoelectric_material/K_electromechanical_constant-field*
- */NXspm_piezoelectric_material/name-field*
- */NXspm_piezoelectric_material/P_pyroelectric_constant-field*
- */NXspm_piezoelectric_material/relative_permittivity-field*
- */NXspm_piezoelectric_material/surface_resistivity-field*
- */NXspm_piezoelectric_material/temperature_range-field*
- */NXspm_piezoelectric_material/type-field*
- */NXspm_piezoelectric_material/young_modulus-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_piezoelectric_material.nxdl.xml

NXspm_positioner

Status:

base class (contribution), extends *NXpositioner*

Description:

An extension of positioner, used to maintain a measurement signal through a feedback loop, specialized for scanning probe microscopy applications.

The component positions the spm head or cantilever tip on the surface of the sample thus maps 2D scan of the sample.

Symbols:

No symbol table

Groups cited:
NXactuator, *NXpid_controller*

Structure:

z_controller: (optional) *NXpid_controller*

This controller's task is to continuously adjust the Z position of tip (e.g., in topography scan in STM/STS experiment) in order to keep the selected control signal as close as possible to the Set Point. Different control signals lead to different controller's behavior.

The second PID feedback loop intends to position the tip in the Z direction.

p_gain (proportional gain) from z_controller refers to K_p value from PID controller. i_gain (integral gain) from z_controller refers to K_i value from PID controller. setpoint from z_controller refers to setpoint from PID controller.

Usually, the same controller, (z_controller) will be used for positioning the tip in three dimensional space. In this case, the controller coefficients (proportional, integral, differential) and other characteristic constants will be the same. Otherwise, for separate controllers positioning the tip in 3D space use the *feedback* controller in actuator.

D_t: (optional) *NX_NUMBER* {units=*NX_TIME*}

The derivative time constant of the derivative part of the PID controller. Proportional Derivative constant K_d can be expressed as $K_d = K_p D_t$.

I_t: (optional) *NX_NUMBER* {units=*NX_TIME*}

The integral time constant of the integral part of the PID controller. Proportional Integral constant K_i can be expressed as $K_i = \frac{K_p}{I_t}$.

z: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

To indicate the relative tip position z between tip and sample. The tip position can also be varied when the z_controller is not running.

set_point: (optional) *NX_NUMBER* {units=*NX_ANY*}

The set point for the z-controller to be fixed and the target value could be height or current.

feedback_on: (recommended) *NX_BOOLEAN*

The status of the controller PID feedback system in z-axis is ON / OFF. E.g., for constant height mode, the z-controller feedback system is off.

tip_lift: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

If the tip is lifted from the stable point.

switch_off_delay: (optional) *NX_CHAR_OR_NUMBER* {units=*NX_TIME*}

The switch-off delay of the controller from its stable point.

z_offset_value: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Offset added to the initial averaged tip position on Z-axis before starting scan.

z_average_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

The averaging time for stabilizing the Z position. Usually, the signal for z-position contains noise, to reduce the noise impacts, z-position is being averaged over this time window.

z_controller_time: (optional) *NX_NUMBER* {units=*NX_TIME*}

Time taken by the Z controller to switch mode e.g., feedback off from on and vice versa. In this time window, controller fixed its position if feedback changed to off mode.

z_controller_hold: (optional) *NX_BOOLEAN*

Controller feedback system status: OFF / ON. The *z_controller_hold-field* is True, refers feedback system off (mostly case in STS bias spectra) and False refers feedback system on (mostly case in STM constant current spectra).

final_z: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The final Z position of the tip after the scan. This parameter is used to indicate any unexpected displacement of the tip.

controller_label: (optional) *NX_CHAR*

Controller label. This label which will be displayed at places where you can select a channel or controller.

actuator: (optional) *NXactuator*

feedback: (optional) *NXpid_controller* <=

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_positioner/actuator-group*
- */NXspm_positioner/actuator/feedback-group*
- */NXspm_positioner/z_controller-group*
- */NXspm_positioner/z_controller/controller_label-field*
- */NXspm_positioner/z_controller/D_t-field*
- */NXspm_positioner/z_controller/feedback_on-field*
- */NXspm_positioner/z_controller/final_z-field*
- */NXspm_positioner/z_controller/I_t-field*
- */NXspm_positioner/z_controller/set_point-field*
- */NXspm_positioner/z_controller/switch_off_delay-field*

- */NXspm_positioner/z_controller/tip_lift-field*
- */NXspm_positioner/z_controller/z-field*
- */NXspm_positioner/z_controller/z_average_time-field*
- */NXspm_positioner/z_controller/z_controller_hold-field*
- */NXspm_positioner/z_controller/z_controller_time-field*
- */NXspm_positioner/z_controller/z_offset_value-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_positioner.nxdl.xml

NXspm_scan_control

Status:

base class (contribution), extends *NXobject*

Description:

A scan is performed inside an N-dimensional phase space, where each dimension can correspond not only to real space coordinates (x,y) but also to any other parameter. This class contains detailed information about controlling the scan in such a phase space (or its subspace).

scan_types: Trajectory: A list of N-dimensional sequential vectors describes the trajectory line for a full scan. Mesh: For each dimension a range and a direction are chosen. When a scan along a dimension is done, a single step in the next dimension is taken, and then the scan in the previous dimension is repeated. As such we can speak about the fastest and the slowest scan axes. Snake: Similar to a mesh scan but with the scanning direction reversed after each line. Spiral: A scan taken along a spiral trajectory. Linear: A scan where the scanning will be performed along a single independent axis. Tilt: At each step, a proportional movement is done in all dimensions (an special case of Trajectory scan).

Scan_control_types: Stepping: At each step, a movement to the next point is performed; correction (for example backlash) or active regulation (feedback loop) may or may not be applied. After the movement is done, the measurement is performed without the movement. Continuous: The scanning of each line in an N-dimensional phase space is done without stopping; measurements are done simultaneously with the movement. Oscillating: Scanning over a scan point continuously and then moving to start scanning at the next position.

Symbols:

No symbol table

Groups cited:
NXspm_scan_pattern, *NXspm_scan_region*

Structure:

scan_time_start: (optional) *NX_DATE_TIME* {units=*NX_TIME*}

The start time of the scan.

scan_time_end: (optional) *NX_DATE_TIME* {units=*NX_TIME*}

The end time of the scan.

independent_scan_axes: (optional) *NX_CHAR*

An array of scan axes which are controlled independently of each other. (e.g. X, Y, Z, or other physical dimensions)

The array elements are in the order of axes of the scan from the fastest to the slowest.

scan_resolutionN: (optional) *NX_NUMBER* {units=*NX_ANY*}

Define the scan resolution along each dimension as the number of steps per unit of the dimension parameters.

Rename the field according to the name of the independent dimension (e.g. scan_resolution_x, scan_resolution_voltage).

accuracyN: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

Define the accuracy of the scan probe.

scan_type: (optional) *NX_CHAR*

This group specifies how the trajectory of the scan is defined.

Any of these values:

- `trajectory`
- `mesh`
- `snake`
- `spiral`
- `linear`
- `tilt`

scan_control_type: (optional) *NX_CHAR*

This string describes how the scan was performed.

Any of these values: `stepping` | `continuous` | `oscillating`

scan_region: (optional) *NXspm_scan_region*

The scan region is the area of phase space or sub-phase space where the scan is performed. The region could be N-dimensional and is defined by the minimum and maximum values of the scan axes.

meshSCAN: (optional) *NXspm_scan_pattern*

For each dimension a range and a direction are chosen. When a scan along a dimension is done, a single step in the next dimension is taken, and then the scan in the previous dimension is repeated. As such we can speak about the fastest and the slowest scan axes.

spiralSCAN: (optional) *NXspm_scan_pattern*

To define the spiral or circular scan, use this group.

spiral_radiusN: (optional) *NX_NUMBER* {units=*NX_ANY*} <=

Define the radius of the spiral circle of scanning.

Rename the field, according to the circle order, the nearest circle to the center is 0.

scan_pointsN: (optional) *NX_NUMBER* <=

Define the total number of points in a given circle scan to be performed.

Rename the field, according to the circle order, the nearest circle to the center is 0 (e.g. scan_points_2).

snakeSCAN: (optional) *NXspm_scan_pattern*

To define the snake scan, use this group.

trajSCAN: (optional) *NXspm_scan_pattern*

To define the trajectory scan, use this group.

linearSCAN: (optional) *NXspm_scan_pattern*

To define the linear scan, use this group.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_scan_control/accuracyN-field*
- */NXspm_scan_control/independent_scan_axes-field*
- */NXspm_scan_control/linearSCAN-group*
- */NXspm_scan_control/meshSCAN-group*
- */NXspm_scan_control/scan_control_type-field*
- */NXspm_scan_control/scan_region-group*
- */NXspm_scan_control/scan_resolutionN-field*
- */NXspm_scan_control/scan_time_end-field*
- */NXspm_scan_control/scan_time_start-field*
- */NXspm_scan_control/scan_type-field*
- */NXspm_scan_control/snakeSCAN-group*
- */NXspm_scan_control/spiralSCAN-group*
- */NXspm_scan_control/spiralSCAN/scan_pointsN-field*
- */NXspm_scan_control/spiralSCAN/spiral_radiusN-field*
- */NXspm_scan_control/trajSCAN-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_scan_control.nxdl.xml

NXspm_scan_pattern

Status:

base class (contribution), extends *NXobject*

Description:

Basic base class to define the pattern of a scan in a given scan region.

The base class is intended to handle the following scan types or patterns:

Trajectory: A list of N-dimensional sequential vectors, representing a point in phase space, describes
the trajectory line for a full scan.

Mesh: For each dimension, a range and a direction are chosen. When a scan along a dimension is done, a
single step in the next dimension is taken, and then the scan in the previous dimension is repeated. As such we can speak about the fastest and the slowest scan axes.

Snake: Similar to a mesh scan but the current scanning direction reversed after each line completed. Spiral: A scan taken along a spiral trajectory. Tilt: At each step, a proportional movement is done in all dimensions (an special case of Trajectory scan). Linear: A scan where the scanning will be performed along a single independent axis.

Symbols:

No symbol table

Groups cited:
NXdata

Structure:

scan_speedN: (optional) *NX_NUMBER* {units=*NX_ANY*}

Define the scan speed in the forward direction along the axis (except the spiral and trajectory scans), if forward and backward speeds below are not specified.

If the scan goes in the negative direction, the speed should be negative.

Rename the field, according to the name of the dimension (e.g. scan_speed_x, scan_speed_voltage).

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

forward_speedN: (optional) *NX_NUMBER* {units=*NX_ANY*}

Define the scan speed in the forward directions (except the spiral and trajectory scans). Rename the field, according to the name of the dimension.

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

backward_speedN: (optional) *NX_NUMBER* {units=*NX_ANY*}

Define the scan speed in the backward directions (except the spiral and trajectory scans). Rename the field, according to the name of the dimension.

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

channelNAME: (optional) *NX_CHAR*

Name of the channel that leads the scan data. The ending annotation 'NAME' is to represent the name of the channel and/or dimension.

Rename the field, according to the name of the channel and dimension (e.g. channel_piezo_scanner_x).

scan_pointsN: (optional) *NX_NUMBER*

Define the total number of points in the given axis the scan to be performed (except the spiral and trajectory scans). Rename the field, according to the name of the dimension (e.g. scan_points_x, scan_points_voltage).

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

steppingN: (optional) *NX_NUMBER*

The number of steps the probe jumps over the scan steps or points in the scanning process. This comes into picture, when not every point from the scan_points is scanned along an axis (except the spiral and trajectory scans).

Rename the field, according to the name of the dimension (e.g. stepping_x, stepping_voltage).

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

step_sizeN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The size of each step in the scan on each dimension (except the spiral and trajectory scans).

Rename the field, according to the name of the dimension (e.g. step_size_x, step_size_voltage).

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

continuousN: (optional) *NX_BOOLEAN*

If the scan probe moves continuously over the scan points or steps, use True. The default value is True. Usually, continuous scanning is possible in one dimension. On other dimensions, the scan probe moves in steps.

Rename the field, according to the name of the dimension (e.g. continuous_voltage).

Trajectory scan: N refers to the nth trajectory line (line between two trajectory points). Spiral scan: N refers to the nth circle from center.

oscillating: (optional) *NX_BOOLEAN*

If the scan probe oscillates over the scan point, use True. The default value is False.

oscillation_frequency: (optional) *NX_NUMBER*

The number of oscillations on each scanning point per second.

number_of_trajectory_points: (optional) *NX_NUMBER*

The number of trajectory points in the entire scan that defines the path or lines of the trajectory. Each trajectory line is defined by two subsequent trajectory points.

trajectory_points: (optional) *NX_NUMBER* (Rank: 2, Dimensions: [nTraj, nD])

The trajectory points (nTraj) are the N-dimensional vectors (nD), each vector refers to a point in N-dimensional phase space.

spiral_radiusN: (optional) *NX_NUMBER* {units=*NX_ANY*}

Define the radius of the spiral circle of scanning.

Rename the field, according to the circle order, the nearest circle to the center is 0.

DATA: (optional) *NXdata* <=

The scan data is the data collected during the scan. If the scan has several channels or derivatives from the channel data, please duplicate this NXdata group for each.

DATA: (optional) *NX_NUMBER* {units=*NX_ANY*} <=

The data (e.g. current, voltage, temperature) field that can be plotted against the axes.

AXISNAME: (optional) *NX_CHAR_OR_NUMBER* {units=*NX_ANY*} <=

The name of the axis that corresponds to the data field.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_scan_pattern/backward_speedN-field*
- */NXspm_scan_pattern/channelNAME-field*
- */NXspm_scan_pattern/continuousN-field*
- */NXspm_scan_pattern/DATA-group*
- */NXspm_scan_pattern/DATA/AXISNAME-field*
- */NXspm_scan_pattern/DATA/DATA-field*
- */NXspm_scan_pattern/forward_speedN-field*
- */NXspm_scan_pattern/number_of_trajectory_points-field*
- */NXspm_scan_pattern/oscillating-field*
- */NXspm_scan_pattern/oscillation_frequency-field*
- */NXspm_scan_pattern/scan_pointsN-field*
- */NXspm_scan_pattern/scan_speedN-field*
- */NXspm_scan_pattern/spiral_radiusN-field*
- */NXspm_scan_pattern/step_sizeN-field*
- */NXspm_scan_pattern/steppingN-field*
- */NXspm_scan_pattern/trajectory_points-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_scan_pattern.nxdl.xml

NXspm_scan_region

Status:

base class (contribution), extends *NXobject*

Description:

The scan region is the area of phase space or sub-phase space where the scan is performed. The region could be N-dimensional and is defined by the minimum and maximum values of the scan axes.

Symbols:

No symbol table

Groups cited:
none

Structure:

scan_offset_valueN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The offset of center of the scan region from the origin along the specific scan axis.

'N' denotes the name of the specific scan axis. (Offset, start and end positions are related)

scan_rangeN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The range of the scan is the difference start and end values of the scan region along the dimension 'N'.

scan_angleN: (optional) *NX_NUMBER* {units=*NX_ANGLE*}

The orientation of the scan region or subspace. Usually, the scan_offset and scan_range are enough to define the scan region. This field defines how the spatial space is oriented with respect to the frame of reference.

Rename the field describing the angle with an axis of the spatial space (e.g. scan_angle_x).

scan_startN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The start of the scan is the starting point of the scan region (phase space or sub-phase space) for each independent scan axis.

For N-dimensional, it is a list of N numbers.

scan_endN: (optional) *NX_NUMBER* {units=*NX_ANY*}

The end of the scan is the ending point of the scan region (phase space or sub-phase space) for each independent scan axis.

Note: The scan_offset and scan_range are equivalent to the scan_start and scan_end.

For N-dimensional, it is a list of N numbers.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_scan_region/scan_angleN-field*
- */NXspm_scan_region/scan_endN-field*
- */NXspm_scan_region/scan_offset_valueN-field*
- */NXspm_scan_region/scan_rangeN-field*
- */NXspm_scan_region/scan_startN-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_scan_region.nxdl.xml

NXspm_temperature_sensor

Status:

base class (contribution), extends *NXsensor*

Description:

The *NXspm_temperature_sensor* class describes a temperature sensor used in scanning probe microscopy (SPM) experiments. It includes fields for the temperature measurement, calibration, and associated data.

Symbols:

No symbol table

Groups cited:
NXcalibration, *NXdata*, *NXparameters*

Structure:

temperature: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*}

The obtained average temperature of the sensor.

target_temperature: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*}

The target temperature of the sensor, which is the desired temperature to be maintained during the experiment.

temp_offset_value: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*}

The offset temperature of the sensor, which will be added to obtained correct the measured temperature.

TEMPERATURE**channel**: (optional) *NX_CHAR*

The name of the channel that handles the temperature measurement.

calibration: (optional) *NXcalibration*

Calibration of the temperature measurement.

calibration_parameters: (optional) *NXparameters* <=

coefficient: (optional) *NX_NUMBER* {units=*NX_ANY*}

The coefficient of the calibration.

DATA: (optional) *NXdata* <=

Data related to the temperature measurement, such as time series data or voltage-temperature data.

DATA: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*} <=

Temperature data collected during the scan.

AXISNAME: (optional) *NX_NUMBER* {units=*NX_ANY*}

independent axis data like time or position or bias voltage.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXspm_temperature_sensor/calibration-group*
- */NXspm_temperature_sensor/calibration/calibration_parameters-group*
- */NXspm_temperature_sensor/calibration/calibration_parameters/coefficient-field*
- */NXspm_temperature_sensor/DATA-group*
- */NXspm_temperature_sensor/DATA/AXISNAME-field*
- */NXspm_temperature_sensor/DATA/DATA-field*
- */NXspm_temperature_sensor/target_temperature-field*
- */NXspm_temperature_sensor/temp_offset_value-field*
- */NXspm_temperature_sensor/temperature-field*
- */NXspm_temperature_sensor/TEMPERATUREchannel-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXspm_temperature_sensor.nxdl.xml

NXstm

Status:

application definition (contribution), extends *NXspm*

Description:

An application definition to describe Scanning Tunneling Microscopy (STM).

Symbols:

No symbol table

Groups cited:
NXcollection, *NXentry*, *NXenvironment*, *NXinstrument*, *NXlockin*, *NXsensor*, *NXspm_scan_control*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Name of the definition that is used for the STM technique.

Obligatory value: `NXstm`

scan_mode: (required) *NX_CHAR* <=

The mode of the scan that is performed. Two commonly used modes are constant height mode and constant current mode.

Any of these values: `constant height|constant current`

experiment_technique: (required) *NX_CHAR* <=

The specific to STM experiment.

Obligatory value: `STM`

INSTRUMENT: (required) *NXinstrument* <=

lockin_amplifier: (required) *NXlockin* <=

The lock-in amplifier information. The device is being used to extract the very weak signal buried in noisy signals.

modulation_signal: (recommended) *NX_CHAR* {units=*NX_UNITLESS*} <=

The type of the signal (voltage or current) subject to modulation.

SCAN_ENVIRONMENT: (required) *NXenvironment* <=

The environment information for stm or sts experiment.

Note: At least one field from head_temperature, cryo_bottom_temperature and cryo_shield_temperature must be provided.

lockin_status: (recommended) *NX_BOOLEAN*

The status of the lock-in amplifier. It is well established convention that for topographic measurements lockin amplifier is turned off state while current scan lockin amplifier is turned on.

SPM_SCAN_CONTROL: (required) *NXspm_scan_control* <=

The scan control information like scan region or phase space, e.g. pattern of scan (e.g. mesh), scanner speed, etc.

head_temperature_sensor: (optional) *NXsensor* <=

This should be a link to /entry/instrument/head_temperature_sensor

cryo_bottom_temperature_sensor: (optional) *NXsensor* <=

This should be a link to /entry/instrument/cryo_bottom_temperature_sensor

cryo_shield_temperature_sensor: (optional) *NXsensor* <=

This should be a link to /entry/instrument/cryo_shield_temperature_sensor

bias_spectroscopy_environment: (optional) *NXenvironment* <=

To explain bias and current behavior (sweep measurement) due to voltage applied to the sample.

reproducibility_indicators: (optional) *NXcollection* <=

The group's concepts hold the link to the related concepts that define the reproducibility of the STM experiment.

current: (optional) *NX_NUMBER*

The tunneling current between tip and sample after application of bias voltage. This should be a link to /entry/instrument/current_sensor/current

current_offset: (optional) *NX_NUMBER*

The offset in tunneling current between tip and sample after application of bias voltage. This should be a link to /entry/instrument/current_sensor/current_offset

modulation_signal_type: (optional) *NX_CHAR* <=

This is the signal on which the modulation voltage or current will be added. This should be a link to /entry/instrument/phase_lock_loop/modulation_signal

reference_frequency: (optional) *NX_NUMBER*

The frequency of the sine modulation that is used as a carrier signal of input signal in lock-in.

This should be a link to /entry/instrument/lockin_amplifier/reference_frequency

BIAS_SWEEP: (optional) *NXspm_scan_control*

Bias sweep measurement in bias spectroscopy. This should be a link to /entry/instrument/bias_spectroscopy_environment/bias_spectroscopy/bias_sweep

resolution_indicators: (optional) *NXcollection* <=

The group's concepts hold the link to the related concepts that define the resolution of the STM experiment.

head_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/head_temperature

cryo_bottom_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_bottom_temperature

cryo_shield_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_shield_temperature

modulation_signal_type: (optional) *NX_CHAR* <=

This is the signal on which the modulation voltage or current will be added. This should be a link to /entry/instrument/lockin_amplifier/modulation_signal_type

reference_frequency: (optional) *NX_NUMBER*

The frequency of the sine modulation that is used to modulate the signal in lock-in. This should be a link to /entry/instrument/lockin_amplifier/reference_frequency

BIAS_SWEEP: (optional) *NXspm_scan_control*

This should be a link to /entry/instrument/bias_spectroscopy_environment/bias_spectroscopy/bias_sweep

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXstm/ENTRY-group*
- */NXstm/ENTRY/definition-field*
- */NXstm/ENTRY/experiment_technique-field*
- */NXstm/ENTRY/INSTRUMENT-group*
- */NXstm/ENTRY/INSTRUMENT/bias_spectroscopy_environment-group*
- */NXstm/ENTRY/INSTRUMENT/lockin_amplifier-group*
- */NXstm/ENTRY/INSTRUMENT/lockin_amplifier/modulation_signal-field*
- */NXstm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT-group*
- */NXstm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/cryo_bottom_temperature_sensor-group*
- */NXstm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/cryo_shield_temperature_sensor-group*
- */NXstm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/head_temperature_sensor-group*
- */NXstm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/lockin_status-field*
- */NXstm/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL-group*
- */NXstm/ENTRY/reproducibility_indicators-group*
- */NXstm/ENTRY/reproducibility_indicators/BIAS_SWEEP-group*
- */NXstm/ENTRY/reproducibility_indicators/current-field*
- */NXstm/ENTRY/reproducibility_indicators/current_offset-field*
- */NXstm/ENTRY/reproducibility_indicators/modulation_signal_type-field*
- */NXstm/ENTRY/reproducibility_indicators/reference_frequency-field*

- */NXstm/ENTRY/resolution_indicators-group*
- */NXstm/ENTRY/resolution_indicators/BIAS_SWEEP-group*
- */NXstm/ENTRY/resolution_indicators/cryo_bottom_temperature-field*
- */NXstm/ENTRY/resolution_indicators/cryo_shield_temperature-field*
- */NXstm/ENTRY/resolution_indicators/head_temperature-field*
- */NXstm/ENTRY/resolution_indicators/modulation_signal_type-field*
- */NXstm/ENTRY/resolution_indicators/reference_frequency-field*
- */NXstm/ENTRY/scan_mode-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXstm.nxdl.xml

NXsts

Status:

application definition (contribution), extends *NXspm*

Description:

An application definition to describe Scanning Tunneling Spectroscopy (STS).

NXsts is an extension of NXspm.

Symbols:

No symbol table

Groups cited:
NXcollection, *NXentry*, *NXenvironment*, *NXinstrument*, *NXlockin*, *NXpid_controller*, *NXsensor*, *NXspm_bias_spectroscopy*, *NXspm_positioner*, *NXspm_scan_control*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Name of the definition that is used for the application.

Obligatory value: `NXsts`

experiment_technique: (required) *NX_CHAR* <=

The specific to STM experiment.

Obligatory value: `STS`

scan_mode: (required) *NX_CHAR* <=

The mode between the tip and sample of the tunneling spectroscopy experiment.

Any of these values or a custom value (if you use a custom value, also set @custom=True):

- `constant height`
- `constant current`
- `constant spacing`

INSTRUMENT: (required) *NXinstrument* <=

lockin_amplifier: (recommended) *NXlockin* <=

The lock-in amplifier information.

current_sensorTAG: (recommended) *NXsensor* <=

Information for current sensor.

SCAN_ENVIRONMENT: (optional) *NXenvironment* <=

SPM_SCAN_CONTROL: (recommended) *NXspm_scan_control* <=

bias_spectroscopy_environment: (required) *NXenvironment* <=

To explain bias (sweep measurement) voltage applied to the sample.

SPM_BIAS_SPECTROSCOPY: (required) *NXspm_bias_spectroscopy* <=

Setup and scan data for continuous measurement of bias-voltage on the subject of experiment vs tunneling current from probe.
SPM_POSITIONER: (recommended) *NXspm_positioner* <=

z_controller: (recommended) *NXpid_controller* <=

reproducibility_indicators: (optional) *NXcollection* <=

The group's concepts hold the link to the related concepts that define the reproducibility of the STM experiment.

current: (optional) *NX_NUMBER*

The tunneling current between tip and sample after application of bias voltage. This should be a link to /ENTRY[*]/INSTRUMENT[*]/current_sensorTAG/current

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

current_offset: (optional) *NX_NUMBER*

The offset in tunneling current between tip and sample after application of bias voltage. This should be a link to /ENTRY[*]/INSTRUMENT[*]/current_sensorTAG[current_sensor*]/current_offset

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

current_gain: (optional) *NX_NUMBER*

Proportional relationship between the probe output voltage and the actual tunneling current when measuring the tunneling current. This should be a link to /ENTRY[*]/INSTRUMENT[*]/current_sensorTAG[current_sensor*]/amplifier/current_gain

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

modulation_signal_type: (optional) *NX_CHAR* <=

This is the signal on which the modulation voltage or current will be added. This should be a link to /ENTRY[*]/INSTRUMENT[*]/lockin_amplifier/modulation_signal_type

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

reference_frequency: (optional) *NX_NUMBER*

The frequency of the sine modulation that is used as carrier signal of input signal in lock-in. This should be a link to /ENTRY[*]/INSTRUMENT[*]/lockin_amplifier/reference_frequency

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

BIAS_SWEEP: (optional) *NXspm_scan_control*

Bias sweep measurement in bias spectroscopy. This should be a link to /ENTRY[*]/INSTRUMENT[*]/bias_spectroscopy_environment/BIAS_SPECTROSCOPY[bias_spectroscopy]/bias_sw

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

resolution_indicators: (optional) *NXcollection* <=

The group's concepts hold the link to the related concepts that define the resolution of the STM experiment.

head_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/head_temperature

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

cryo_bottom_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_bottom_temperature

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

cryo_shield_temperature: (optional) *NX_NUMBER*

This should be a link to /entry/instrument/scan_environment/cryo_shield_temperature

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

modulation_signal_type: (optional) *NX_CHAR* <=

This is the signal on which the modulation voltage or current will be added. This should be a link to /ENTRY[*]/INSTRUMENT[*]/lockin_amplifier/modulation_signal_type

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

modulation_frequency: (optional) *NX_NUMBER*

The frequency of the sine modulation that is used to modulate the signal in lock-in. This should be a link to /ENTRY[*]/INSTRUMENT[*]/lockin_amplifier/modulation_frequency

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the

BIAS_SWEEP: (optional) *NXspm_scan_control*

This should be a link to /ENTRY[*]/INSTRUMENT[*]/bias_spectroscopy_environment/SPM_BIAS_SPECTROSC

Note: group name (could be any meaningful and relevant name e.g. entry in ENTRY[entry]) inside the square bracket would be the *exact* name of the NXentry group.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsts/ENTRY-group*
- */NXsts/ENTRY/definition-field*
- */NXsts/ENTRY/experiment_technique-field*
- */NXsts/ENTRY/INSTRUMENT-group*
- */NXsts/ENTRY/INSTRUMENT/bias_spectroscopy_environment-group*
- */NXsts/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY-group*
- */NXsts/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/SPM_POSITIONER-group*
- */NXsts/ENTRY/INSTRUMENT/bias_spectroscopy_environment/SPM_BIAS_SPECTROSCOPY/SPM_POSITIONER/z_controller-group*
- */NXsts/ENTRY/INSTRUMENT/current_sensorTAG-group*
- */NXsts/ENTRY/INSTRUMENT/lockin_amplifier-group*
- */NXsts/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT-group*
- */NXsts/ENTRY/INSTRUMENT/SCAN_ENVIRONMENT/SPM_SCAN_CONTROL-group*
- */NXsts/ENTRY/reproducibility_indicators-group*
- */NXsts/ENTRY/reproducibility_indicators/BIAS_SWEEP-group*
- */NXsts/ENTRY/reproducibility_indicators/current-field*
- */NXsts/ENTRY/reproducibility_indicators/current_gain-field*
- */NXsts/ENTRY/reproducibility_indicators/current_offset-field*
- */NXsts/ENTRY/reproducibility_indicators/modulation_signal_type-field*
- */NXsts/ENTRY/reproducibility_indicators/reference_frequency-field*
- */NXsts/ENTRY/resolution_indicators-group*

- */NXsts/ENTRY/resolution_indicators/BIAS_SWEEP-group*
- */NXsts/ENTRY/resolution_indicators/cryo_bottom_temperature-field*
- */NXsts/ENTRY/resolution_indicators/cryo_shield_temperature-field*
- */NXsts/ENTRY/resolution_indicators/head_temperature-field*
- */NXsts/ENTRY/resolution_indicators/modulation_frequency-field*
- */NXsts/ENTRY/resolution_indicators/modulation_signal_type-field*
- */NXsts/ENTRY/scan_mode-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXsts.nxdl.xml

NXsubsampling_filter

Status:

base class (contribution), extends *NXparameters*

Description:

Base class of a filter to sample members in a set based on their indices.

The filter defines three parameters: The minimum, the increment, and the maximum index of values to include of a sequence $[i_0, i_0+1, i_0+2, \ldots, i_0+\mathcal{N}] with i_0 \in \mathcal{Z}$ of indices. The increment controls which n-th index (value) to take.

Take as an example a dataset with 100 indices (aka entries). Assume that the indices start at zero, i.e., index_offset is 0. Assume further that min, increment, max are set to 0, 1, and 99, respectively. In this case the filter will yield all indices. Setting min, increment, max to 0, 2, and 99, respectively will yield each second index value. Setting min, increment, max to 90, 3, and 99 respectively will yield each third index value beginning from index values 90 up to 99.

Symbols:

No symbol table

Groups cited:
none

Structure:

min: (optional) *NX_INT* {units=*NX_UNITLESS*}

Minimum index.

increment: (optional) *NX_INT* {units=*NX_UNITLESS*}

Increment.

max: (optional) *NX_INT* {units=*NX_UNITLESS*}

Maximum index.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXsubsampling_filter/increment-field*
- */NXsubsampling_filter/max-field*
- */NXsubsampling_filter/min-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*, *NXresolution*, *NXsample*, *NXslit*, *NXsource*, *NXuser*

Structure:

ENTRY: (required) *NXentry*

definition: (required) *NX_CHAR* <=

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.

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.

instrument: (required) *NXinstrument* <=

common_beam_depolarizer: (required) *NX_BOOLEAN*

If true, the incident beam is depolarized.

polarizer: (required) *NX_NUMBER* {units=*NX_ANGLE*}

Polarizer value inside the beam path

time_points: (optional) *NX_NUMBER* (Rank: 1, Dimensions: [N_scans]) {units=*NX_TIME*}

An array of relative scan start time points.

measured_data: (required) *NX_NUMBER* (Rank: 2, Dimensions: [N_scans, N_wavelengths])

Resulting data from the measurement. The length of the 2nd dimension is the number of time points. If it has length one the time_points may be empty.

manufacturer: (recommended) *NXfabrication* <=

Manufacturer of the instrument.

common_beam_mask: (required) *NXslit*

Common beam mask to shape the incident beam

y_gap: (required) *NX_NUMBER* {units=*NX_UNITLESS*} <=

The height of the common beam in percentage of the beam

ref_attenuator: (required) *NXattenuator* <=

Attenuator in the reference beam

attenuator_transmission: (required) *NX_FLOAT* <=

sample_attenuator: (required) *NXattenuator* <=

Attenuator in the sample beam

attenuator_transmission: (required) *NX_FLOAT* <=

spectrometer: (required) *NXmonochromator* <=

wavelength: (required) *NX_NUMBER* (Rank: 1, Dimensions: [N_wavelengths]) {units=*NX_LENGTH*}

Wavelength value(s) used for the measurement. An array of 1 or more elements. Length defines N_wavelenghts

spectral_resolution: (optional) *NXresolution*

Overall spectral resolution of this spectrometer. If several gratings are employed the spectral resolution should rather be specified for each grating inside the NXgrating group of this spectrometer.

resolution: (required) *NX_NUMBER* {units=*NX_WAVENUMBER*}

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* {units=*NX_DIMENSIONLESS*}

Dispersion of the grating in nm/mm used.

blaze_wavelength: (optional) *NX_NUMBER* {units=*NX_LENGTH*}

The blaze wavelength of the grating used.

wavelength_range: (required) *NX_NUMBER* (Rank: 1, Dimensions: [2]) {units=*NX_LENGTH*}

Wavelength range in which this grating was used

spectral_resolution: (optional) *NXresolution*

Overall spectral resolution of the instrument when this grating is used.

resolution: (required) *NX_NUMBER* {units=*NX_WAVENUMBER*}

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 or a custom value (if you use a custom value, also set @custom=True): `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-group*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING/spectral_resolution/resolution-field*
- */NXtransmission/ENTRY/instrument/spectrometer/GRATING/wavelength_range-field*
- */NXtransmission/ENTRY/instrument/spectrometer/spectral_resolution-group*
- */NXtransmission/ENTRY/instrument/spectrometer/spectral_resolution/resolution-field*
- */NXtransmission/ENTRY/instrument/spectrometer/wavelength-field*
- */NXtransmission/ENTRY/instrument/time_points-field*
- */NXtransmission/ENTRY/SAMPLE-group*
- */NXtransmission/ENTRY/SAMPLE/name-field*
- */NXtransmission/ENTRY/start_time-field*
- */NXtransmission/ENTRY/USER-group*
- */NXtransmission/ENTRY/USER/affiliation-field*
- */NXtransmission/ENTRY/USER/name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXtransmission.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

$$g_2(\boldsymbol{Q}, t) = \frac{\langle I(\boldsymbol{Q}, t\prime) I(\boldsymbol{Q}, t\prime + t) \rangle}{\langle I(\boldsymbol{Q}, t\prime) \rangle^2}; t > 0$$

Typically, g_2 is a quantity calculated for a group of pixels representing a specific region of reciprocal space. These groupings, or bins, are generically described as q. Some open-source XPCS libraries refer to these bins as "rois", which are not to be confused with EPICS AreaDetector ROI. See usage guidelines for q_lists and roi_maps within a mask.[1]

In short, g_2 should be ordered according to the roi_map value. In principle, any format is acceptable if the data and its axes are self-describing as per NeXus recommendations. However, the data is preferred in one of the following two formats:

- iterable list of linked files (or keys) for each g_2 with 1 file (key) per q, where q is called by the nth roi_map value
- 2D array[2] with shape (g_2, q), where q is represented by the nth roi_map value, not the value q value

Note it is expected that "g2" and all quantities following it will be of the same length.

Other formats are acceptable with sufficient axes description.

See references below for related implementation information:

@storage_mode: (required) *NX_CHAR*

storage_mode describes the format of the data to be loaded

We encourage the documentation of other formats not represented here.

- one array representing entire data set ("one_array")
- data exchange format with each key representing one `q` by its corresponding roi_map value ("data_exchange_keys")

Any of these values: `one_array` | `data_exchange_keys` | `other`

g2_derr: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

error values for the g_2 values.

The derivation of the error is left up to the implemented code. Symmetric error will be expected ($\pm$ error). The data should be in the same format as `g2`.

@storage_mode: (required) *NX_CHAR*

Any of these values: `one_array` | `data_exchange_keys` | `other`

G2_unnormalized: (optional) *NX_NUMBER* {units=*NX_ANY*}

unnormalized intensity auto-correlation function.

Specifically, `g2` without the denominator. The data should be in the same format as `g2`.

[1] mask: `NXxpcs:/entry/instrument/masks-group`
[2] NeXus 2-D data and axes: https://manual.nexusformat.org/classes/base_classes/NXdata.html#nxdata

@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:

$$C(\boldsymbol{Q}, t_1, t_2) = \frac{\langle I(\boldsymbol{Q}, t_1) I(\boldsymbol{Q}, t_2) \rangle}{\langle I(\boldsymbol{Q}, t_1) \rangle \langle I(\boldsymbol{Q}, t_2) \rangle}$$

in which time is quantized by frames. In principle, any data format is acceptable if the data and its axes are self-describing as per NeXus recommendations. However, the data is preferred in one of the following two formats:

- iterable list of linked files (or keys) for each q-bin called by the nth roi_map value. data for each bin is a 2D array
- 3D array with shape (frames, frames, q) or (q, frames, frames), where q is represented by the nth roi_map value, not the value q value

The computation of this result can be customized. These customizations can affect subsequently derived results (below). The following attributes will be used to manage the customization.

- Other normalization methods may be applied, but the method will not be specified in this definition. Some of these normalization methods result in a baseline value of `0`, not `1`.
- The various software libraries use different programming languages. Therefore, we need to specify the `time = 0` origin location of the 2D array for each q.
- A method to reduce data storage needs is to only record half of the 2D array by populating array elements above or below the array diagonal.

@storage_mode: (required) *NX_CHAR*

storage_mode describes the format of the data to be loaded

We encourage the 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, 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.

[3] Coherent X-ray Imaging Data Bank: https://cxidb.org/cxi.html

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.

"roi_maps" provide for representation of pixel binning that are arbitrary and irregular, which is geometry scattering agnostic and most flexible. The maps work as a labeled array for N rois.

"Dynamic" represents quantities directly related to XPCS and NXxcps/entry/data and NXxpcs/entry/two_time.

"Static" refers to finer binning used for computation not strictly used for the final XPCS results. Implementation of _static_ binning is left for individual libraries to document. We encourage usage of *NXcanSAS* to represent standard SAXS results or development of new NeXus definitions for GI-SAXS or other reciprocal space intensity mapping.

dynamic_roi_map: (required) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

roi index array or labeled array

The values of this mask index (or map to) the Q value from the the `dynamic_q_list` field. Not that the value of `0` represents in-action. XPCS computations are performed on all pixels with a value > 0.

The `units` attribute should be set to `"au"` indicating arbitrary units.

dynamic_q_list: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

1-D list of Q values, one for each roi index value.

List order is determined by the index value of the associated roi map starting at `1`.

The only requirement for the list is that it may be iterable. Some expected formats are:

- iterable list of floats (i.e., $Q(r)$)
- iterable list of tuples (i.e., $Q(r)$, φ), but preferable use the separate φ field below

- iterable list of tuples (e.g., (H, K, L); (qx, qy, qz); (horizontal_pixel, vertical_pixel))
- iterable list of integers (for Nth roi_map value) or strings

This format is chosen because results plotting packages are not common and simple I/O is required by end user. The lists can be accessed as lists, arrays or via keys

dynamic_phi_list: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

Array of φ value for each pixel.

List order is determined by the index value of the associated roi map starting at 1.

static_roi_map: (optional) *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

roi index array.

The values of this mask index the $|Q|$ value from the the `static_q_list` field.

The `units` attribute should be set to `"au"` indicating arbitrary units.

static_q_list: (optional) *NX_NUMBER* {units=*NX_PER_LENGTH*}

1-D list of $|Q|$ values, 1 for each roi.

sample: (optional) *NXsample* <=

temperature_set: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*} <=

Sample temperature setpoint, (C or K).

temperature: (optional) *NX_NUMBER* {units=*NX_TEMPERATURE*}

Sample temperature actual, (C or K).

position_x: (optional) *NXpositioner* <=

position_y: (optional) *NXpositioner* <=

position_z: (optional) *NXpositioner* <=

NOTE: (optional) *NXnote* <=

Any other notes.

NAME: The NeXus convention, to use all upper case to indicate the name (here `NOTE`), is left to the file writer. In our case, follow the suggested name pattern and sequence: note_1, note_2, note_3, … Start with `note_1` if the first one, otherwise pick the next number in this sequence.

PROCESS: (required) *NXprocess*

Describe the computation process that produced these results.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxpcs/ENTRY-group*
- */NXxpcs/ENTRY/data-group*
- */NXxpcs/ENTRY/data/delay_difference-field*
- */NXxpcs/ENTRY/data/delay_difference@storage_mode-attribute*
- */NXxpcs/ENTRY/data/frame_average-field*
- */NXxpcs/ENTRY/data/frame_sum-field*
- */NXxpcs/ENTRY/data/g2-field*
- */NXxpcs/ENTRY/data/g2@storage_mode-attribute*
- */NXxpcs/ENTRY/data/g2_derr-field*
- */NXxpcs/ENTRY/data/g2_derr@storage_mode-attribute*
- */NXxpcs/ENTRY/data/G2_unnormalized-field*
- */NXxpcs/ENTRY/data/G2_unnormalized@storage_mode-attribute*
- */NXxpcs/ENTRY/definition-field*
- */NXxpcs/ENTRY/end_time-field*
- */NXxpcs/ENTRY/entry_identifier-field*
- */NXxpcs/ENTRY/entry_identifier_uuid-field*
- */NXxpcs/ENTRY/instrument-group*
- */NXxpcs/ENTRY/instrument/DETECTOR-group*
- */NXxpcs/ENTRY/instrument/DETECTOR/beam_center_x-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/beam_center_y-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/count_time-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/description-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/distance-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/frame_time-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/x_pixel_size-field*
- */NXxpcs/ENTRY/instrument/DETECTOR/y_pixel_size-field*
- */NXxpcs/ENTRY/instrument/incident_beam-group*
- */NXxpcs/ENTRY/instrument/incident_beam/extent-field*
- */NXxpcs/ENTRY/instrument/incident_beam/incident_energy-field*
- */NXxpcs/ENTRY/instrument/incident_beam/incident_energy_spread-field*
- */NXxpcs/ENTRY/instrument/incident_beam/incident_polarization_type-field*
- */NXxpcs/ENTRY/instrument/masks-group*
- */NXxpcs/ENTRY/instrument/masks/dynamic_phi_list-field*
- */NXxpcs/ENTRY/instrument/masks/dynamic_q_list-field*

- */NXxpcs/ENTRY/instrument/masks/dynamic_roi_map-field*
- */NXxpcs/ENTRY/instrument/masks/static_q_list-field*
- */NXxpcs/ENTRY/instrument/masks/static_roi_map-field*
- */NXxpcs/ENTRY/NOTE-group*
- */NXxpcs/ENTRY/sample-group*
- */NXxpcs/ENTRY/sample/position_x-group*
- */NXxpcs/ENTRY/sample/position_y-group*
- */NXxpcs/ENTRY/sample/position_z-group*
- */NXxpcs/ENTRY/sample/temperature-field*
- */NXxpcs/ENTRY/sample/temperature_set-field*
- */NXxpcs/ENTRY/scan_number-field*
- */NXxpcs/ENTRY/start_time-field*
- */NXxpcs/ENTRY/twotime-group*
- */NXxpcs/ENTRY/twotime/g2_err_from_two_time_corr_func-field*
- */NXxpcs/ENTRY/twotime/g2_err_from_two_time_corr_func@storage_mode-attribute*
- */NXxpcs/ENTRY/twotime/g2_err_from_two_time_corr_func_partials-field*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func-field*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func@baseline_reference-attribute*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func@first_point_for_fit-attribute*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func@storage_mode-attribute*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func_partials-field*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func_partials@baseline_reference-attribute*
- */NXxpcs/ENTRY/twotime/g2_from_two_time_corr_func_partials@storage_mode-attribute*
- */NXxpcs/ENTRY/twotime/two_time_corr_func-field*
- */NXxpcs/ENTRY/twotime/two_time_corr_func@baseline_reference-attribute*
- */NXxpcs/ENTRY/twotime/two_time_corr_func@populated_elements-attribute*
- */NXxpcs/ENTRY/twotime/two_time_corr_func@storage_mode-attribute*
- */NXxpcs/ENTRY/twotime/two_time_corr_func@time_origin_location-attribute*
- */NXxpcs/PROCESS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXxpcs.nxdl.xml

NXxrd

Status:

application definition (contribution), extends *NXmonopd*

Description:

NXxrd on top of NXmonopd

Symbols:

No symbol table

Groups cited:
NXbeam, *NXdata*, *NXdetector*, *NXentry*, *NXinstrument*, *NXprocess*

Structure:

ENTRY: (required) *NXentry* <=

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXxrd`

INSTRUMENT: (optional) *NXinstrument* <=

BEAM: (required) *NXbeam* <=

incident_energy: (required) *NX_FLOAT* {units=*NX_ENERGY*} <=

DETECTOR: (required) *NXdetector* <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*} <=

The 2-theta range of the diffractogram

@units: (required) *NX_CHAR*

Obligatory value: `deg`

raw_data: (optional) *NXdata* <=

raw detector signal (usually counts) as collected it can be a multi-dimensional dataset depending on the detector type (faster axes) and the scanning method (slower axes)

DATA: (required) *NXdata* <=

polar_angle: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])

link (suggested target:/NXentry/NXinstrument/NXdetector/polar_angle) Link to polar ale in /NXentry/NXinstrument/NXdetector

data: (required) *NX_NUMBER* (Rank: 1, Dimensions: [nDet])

link (suggested target:/NXentry/NXinstrument/NXdetector/data) Link to data in /Nxentry/Nxinstrument/Nxdetector

PROCESS: (optional) *NXprocess* <=

Description of a processing or analysis step, such as the baseline extraction or azimuth integration. Add additional fields as needed to describe value(s) of any variable, parameter, or term related to the NXprocess step. Be sure to include units attributes for all numerical fields.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxrd/ENTRY-group*
- */NXxrd/ENTRY/DATA-group*
- */NXxrd/ENTRY/DATA/data-field*
- */NXxrd/ENTRY/DATA/polar_angle-field*
- */NXxrd/ENTRY/definition-field*
- */NXxrd/ENTRY/INSTRUMENT-group*
- */NXxrd/ENTRY/INSTRUMENT/BEAM-group*
- */NXxrd/ENTRY/INSTRUMENT/BEAM/incident_energy-field*
- */NXxrd/ENTRY/INSTRUMENT/DETECTOR-group*
- */NXxrd/ENTRY/INSTRUMENT/DETECTOR/polar_angle-field*
- */NXxrd/ENTRY/INSTRUMENT/DETECTOR/polar_angle@units-attribute*
- */NXxrd/ENTRY/INSTRUMENT/DETECTOR/raw_data-group*
- */NXxrd/ENTRY/PROCESS-group*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXxrd.nxdl.xml

NXxrd_pan

Status:

application definition (contribution), extends *NXxrd*

Description:

NXxrd_pan is a specialization of NXxrd with extra properties for the PANalytical XRD data format.

Symbols:

No symbol table

Groups cited:
NXdata, *NXdetector*, *NXentry*, *NXinstrument*, *NXobject*, *NXsample*, *NXsource*

Structure:

ENTRY: (required) *NXentry* <=

data_file: (optional) *NX_CHAR*

Name of the data file.

measurement_type: (required) *NX_CHAR*

Type of measurement.

definition: (required) *NX_CHAR* <=

Official NeXus NXDL schema to which this file conforms.

Obligatory value: `NXxrd_pan`

method: (required) *NX_CHAR*

Method used to collect the data default='X-Ray Diffraction (XRD)'

Any of these values or a custom value (if you use a custom value, also set @custom=True): `X-Ray Diffraction (XRD)`

INSTRUMENT: (required) *NXinstrument* <=

SOURCE: (required) *NXsource* <=

xray_tube_material: (required) *NX_CHAR*

Type of the X-ray tube.

Any of these values or a custom value (if you use a custom value, also set @custom=True): `Cu` | `Cr` | `Mo` | `Fe` | `Ag` | `In` | `Ga`

xray_tube_current: (required) *NX_FLOAT* {units=*NX_CURRENT*}

Current of the X-ray tube.

xray_tube_voltage: (required) *NX_FLOAT* {units=*NX_VOLTAGE*}

Voltage of the X-ray tube.

k_alpha_one: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength of the Ku03b1 1 line.

@**units**: (required) *NX_CHAR*

Any of these values or a custom value (if you use a custom value, also set @custom=True): `angstrom`

k_alpha_two: (required) *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength of the Ku03b1 2 line.

@**units**: (required) *NX_CHAR*

Any of these values or a custom value (if you use a custom value, also set @custom=True): `angstrom`

ratio_k_alphatwo_k_alphaone: (required) *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Ku03b1 2/Ku03b1 1 intensity ratio.

kbeta: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength of the Ku00df line.

@**units**: (required) *NX_CHAR*

Any of these values or a custom value (if you use a custom value, also set @custom=True): `angstrom`

source_peak_wavelength: (optional) *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength of the X-ray source. Used to convert from 2-theta to Q.

DETECTOR: (required) *NXdetector* <=

scan_axis: (required) *NX_CHAR*

Axis scanned.

scan_mode: (required) *NX_CHAR*

Mode of scan.

integration_time: (optional) *NX_FLOAT* {units=*NX_TIME*}

Integration time per channel.

experiment_config: (optional) *NXobject*

Collect user inputs e.g. name or path of the input file.

beam_attenuation_factors: (required) *NX_CHAR*

Beam attenuation factors over the path.

goniometer_x: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Goniometer position X.

goniometer_y: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Goniometer position Y.

goniometer_z: (optional) *NX_FLOAT* {units=*NX_LENGTH*}

Goniometer position Z

count_time: (required) *NX_FLOAT* {units=*NX_TIME*}

Total time of count.

two_theta: (required) *NXobject*

start: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Starting value of the diffraction angle.

end: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Ending value of the diffraction angle.

step: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Minimum step size in-between two diffraction angles.

omega: (required) *NXobject*

start: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Starting value of the incident angle.

end: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Ending value of the incident angle.

step: (required) *NX_FLOAT* {units=*NX_ANGLE*}

Minimum step size in the between two incident angles.

experiment_result: (required) *NXdata* <=

All experiment results data such as scattering angle (2theta), intensity, incident angle, scattering vector, etc will be stored here.

intensity: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet])

Number of scattered electrons per unit time.

two_theta: (required) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*}

Two-theta (scattering angle) of the diffractogram.

omega: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*}

Incident angle of the diffractogram.

phi: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*}

The phi range of the diffractogram.

chi: (optional) *NX_FLOAT* (Rank: 1, Dimensions: [nDet]) {units=*NX_ANGLE*}

The chi range of the diffractogram

q_parallel: (optional) *NX_FLOAT* {units=*NX_ANY*}

The scattering vector component, which is parallel to the sample surface.

q_perpendicular: (optional) *NX_FLOAT* {units=*NX_ANY*}

The scattering vector component, which is perpendicular to the sample surface.

q_norm: (optional) *NX_FLOAT* {units=*NX_ANY*}

The norm value of the scattering vector, q. The scattering vector is defined as a difference between the incident and scattered wave vectors. For details: https://en.wikipedia.org/wiki/Powder_diffraction and https://theory.labster.com/scattering-vector/

q_data: (optional) *NXdata* <=

The desired view for scattering vectors.

q: (optional) *NX_FLOAT*

This concept corresponds to the norm value of the scattering vector(q). The concept is the same as 'q_norm' of 'experiment_result' and should be linked to /entry[ENTRY]/experiment_result/q_norm.

intensity: (optional) *NX_FLOAT*

Number of scattered electrons per unit time at each scattering vector (q) value. The concept is the same as the 'intensity' of experiment_result and should be linked to /entry[ENTRY]/experiment_result/intensity.

q_parallel: (optional) *NX_FLOAT*

The scattering vector (q) component, which is parallel to the sample surface. This component is used in the Reciprocal Space Mapping (RSM) technique of X-ray diffraction method.

The concept is the same as 'q_parallel' of experiment_result, and should be linked to /entry[ENTRY]/experiment_result/q_parallel.

q_perpendicular: (optional) *NX_FLOAT*

The scattering vector component, which is perpendicular to the sample surface.

The concept is the same as 'q_perpendicular' of experiment_result, and should be linked to /entry[ENTRY]/experiment_result/q_perpendicular.

SAMPLE: (optional) *NXsample* <=

Description on sample.

sample_mode: (required) *NX_CHAR*

Mode of sample.

sample_id: (required) *NX_CHAR*

Id of sample.

sample_name: (required) *NX_CHAR*

Usually in xrd sample are being analyzed, but sample might be identified by assumed name or given name.

Hypertext Anchors

List of hypertext anchors for all groups, fields, attributes, and links defined in this class.

- */NXxrd_pan/ENTRY-group*
- */NXxrd_pan/ENTRY/data_file-field*
- */NXxrd_pan/ENTRY/definition-field*
- */NXxrd_pan/ENTRY/experiment_config-group*
- */NXxrd_pan/ENTRY/experiment_config/beam_attenuation_factors-field*
- */NXxrd_pan/ENTRY/experiment_config/count_time-field*
- */NXxrd_pan/ENTRY/experiment_config/goniometer_x-field*
- */NXxrd_pan/ENTRY/experiment_config/goniometer_y-field*
- */NXxrd_pan/ENTRY/experiment_config/goniometer_z-field*
- */NXxrd_pan/ENTRY/experiment_config/omega-group*
- */NXxrd_pan/ENTRY/experiment_config/omega/end-field*
- */NXxrd_pan/ENTRY/experiment_config/omega/start-field*
- */NXxrd_pan/ENTRY/experiment_config/omega/step-field*
- */NXxrd_pan/ENTRY/experiment_config/two_theta-group*
- */NXxrd_pan/ENTRY/experiment_config/two_theta/end-field*
- */NXxrd_pan/ENTRY/experiment_config/two_theta/start-field*
- */NXxrd_pan/ENTRY/experiment_config/two_theta/step-field*
- */NXxrd_pan/ENTRY/experiment_result-group*
- */NXxrd_pan/ENTRY/experiment_result/chi-field*
- */NXxrd_pan/ENTRY/experiment_result/intensity-field*
- */NXxrd_pan/ENTRY/experiment_result/omega-field*
- */NXxrd_pan/ENTRY/experiment_result/phi-field*
- */NXxrd_pan/ENTRY/experiment_result/q_norm-field*
- */NXxrd_pan/ENTRY/experiment_result/q_parallel-field*
- */NXxrd_pan/ENTRY/experiment_result/q_perpendicular-field*

- */NXxrd_pan/ENTRY/experiment_result/two_theta-field*
- */NXxrd_pan/ENTRY/INSTRUMENT-group*
- */NXxrd_pan/ENTRY/INSTRUMENT/DETECTOR-group*
- */NXxrd_pan/ENTRY/INSTRUMENT/DETECTOR/integration_time-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/DETECTOR/scan_axis-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/DETECTOR/scan_mode-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE-group*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/k_alpha_one-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/k_alpha_one@units-attribute*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/k_alpha_two-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/k_alpha_two@units-attribute*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/kbeta-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/kbeta@units-attribute*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/ratio_k_alphatwo_k_alphaone-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/source_peak_wavelength-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/xray_tube_current-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/xray_tube_material-field*
- */NXxrd_pan/ENTRY/INSTRUMENT/SOURCE/xray_tube_voltage-field*
- */NXxrd_pan/ENTRY/measurement_type-field*
- */NXxrd_pan/ENTRY/method-field*
- */NXxrd_pan/ENTRY/q_data-group*
- */NXxrd_pan/ENTRY/q_data/intensity-field*
- */NXxrd_pan/ENTRY/q_data/q-field*
- */NXxrd_pan/ENTRY/q_data/q_parallel-field*
- */NXxrd_pan/ENTRY/q_data/q_perpendicular-field*
- */NXxrd_pan/ENTRY/SAMPLE-group*
- */NXxrd_pan/ENTRY/SAMPLE/sample_id-field*
- */NXxrd_pan/ENTRY/SAMPLE/sample_mode-field*
- */NXxrd_pan/ENTRY/SAMPLE/sample_name-field*

NXDL Source:
https://github.com/nexusformat/definitions/blob/main/contributed_definitions/NXxrd_pan.nxdl.xml

3.3.4 Downloads

See this table for the different formats available:

download file	description
`/_static/nxdl_vocabulary.html`	Human-readable HTML list of anchors, by vocabulary term, with links to the manual.
`/_static/nxdl_vocabulary.json`	vocabulary list by key in JSON format[1]
`/_static/nxdl_vocabulary.txt`	list of all anchors, sorted alphabetically
`/_static/nxdl_vocabulary.yml`	vocabulary list by key in YAML format[2]

[1] JSON: https://www.w3schools.com/whatis/whatis_json.asp
[2] YAML https://yaml.org

CHAPTER

FOUR

NAPI: NEXUS APPLICATION PROGRAMMER INTERFACE (FROZEN)

4.1 Status

This application program interface (API) was developed to support the reading and writing of NeXus files through unified function calls, regardless of the physical data format (XML, HDF4, HDF5).

In the meantime it has been decided that active development of NeXus definitions and tools will concentrate on HDF5 as the only supported physical data format. It is expected that most application developers will use standard HDF5 tools to read and write NeXus. Two examples are provided in *HDF5 in C with libhdf5*.

Therefore, the decision has been taken to freeze the NAPI. Maintenance is reduced to bug fixes.

4.2 Overview

The core routines have been written in C but wrappers are available for a number of other languages including C++, Fortran 77, Fortran 90, Java, Python and IDL. The API makes the reading and writing of NeXus files transparent; the user doesn't even need to know the underlying format when reading a file since the API calls are the same.

The NeXus Application Programming Interface for the various language backends is available on-line from https://github.com/nexusformat/code/

The NeXusIntern.pdf document (https://github.com/nexusformat/code/blob/master/doc/api/NeXusIntern.pdf) describes the internal workings of the NeXus-API. You are very welcome to read it, but it will not be of much use if all you want is to read and write files using the NAPI.

The NeXus Application Program Interface call routines in the appropriate backend (HDF4, HDF5 or XML) to read and write files with the correct structure. The API serves a number of purposes:

1. It simplifies the reading and writing of NeXus files.
2. It ensures a certain degree of compliance with the NeXus standard.
3. It hides the implementation details of the format. In particular, the API can read and write HDF4, HDF5, and XML files using the same routines.

4.3 Core API

The core API provides the basic routines for reading, writing and navigating NeXus files. Operations are performed using a handle that keeps a record of its current position in the file hierarchy. All are read or write requests are then implicitly performed on the currently *open* entity. This limits number of parameters that need to be passed to API calls, at the cost of forcing a certain mode of operation. It is very similar to navigating a directory hierarchy; NeXus groups are the directories, which can contain data sets and/or other directories.

The core API comprises the following functional groups:

- General initialization and shutdown: opening and closing the file, creating or opening an existing group or dataset, and closing them.
- Reading and writing data and attributes to previously opened datasets.
- Routines to obtain meta-data and to iterate over component datasets and attributes.
- Handling of linking and group hierarchy.
- Routines to handle memory allocation. (Not required in all language bindings.)

4.3.1 NAPI C and C++ Interface

Documentation is provided online:

C
https://manual.nexusformat.org/doxygen/html-c/

C++
https://manual.nexusformat.org/doxygen/html-cpp/ https://github.com/nexusformat/code/tree/master/bindings/cpp

4.3.2 NAPI Fortran 77 Interface

The bindings are listed at https://github.com/nexusformat/code/tree/master/bindings/f77 and can be built as part of the API distribution https://github.com/nexusformat/code/releases

4.3.3 NAPI Fortran 90 Interface

The Fortran 90 interface is a wrapper to the C interface with nearly identical routine definitions. As with the Fortran 77 interface, it is necessary to reverse the order of indices in multidimensional arrays, compared to an equivalent C program, so that data are stored in the same order in the NeXus file.

Any program using the F90 API needs to put the following line at the top (after the `PROGRAM` statement):

```
use NXmodule
```

Use the following table to convert from the C data types listed with each routine to the Fortran 90 data types.

C data type	F90 data type
`int, int`	`integer`
`char*`	`character(len=*)`
`NXhandle, NXhandle*`	`type(NXhandle)`
`NXstatus`	`integer`
`int[]`	`integer(:)`
`void*`	`real(:)` or `integer(:)` or `character(len=*)`
`NXlink a, NXlink* a`	`type(NXlink)`

The parameters in the next table, defined in `NXmodule`, may be used in defining variables.

Name	Description	Value
`NX_MAXRANK`	Maximum number of dimensions	`32`
`NX_MAXNAMELEN`	Maximum length of NeXus name	`64`
`NXi1`	Kind parameter for a 1-byte integer	`selected_int_kind(2)`
`NXi2`	Kind parameter for a 2-byte integer	`selected_int_kind(4)`
`NXi4`	Kind parameter for a 4-byte integer	`selected_int_kind(8)`
`NXr4`	Kind parameter for a 4-byte real	`kind(1.0)`
`NXr8`	Kind parameter for an 8-byte real	`kind(1.0D0)`

The bindings are listed at https://github.com/nexusformat/code/tree/master/bindings/f90 and can be built as part of the API distribution https://github.com/nexusformat/code/releases

4.3.4 NAPI Java Interface

This section includes installation notes, instructions for running NeXus for Java programs and a brief introduction to the API.

The Java API for NeXus (`jnexus`) was implemented through the Java Native Interface (JNI) to call on to the native C library. This has a number of disadvantages over using pure Java, however the most popular file backend HDF5 is only available using a JNI wrapper anyway.

Acknowledgement

This implementation uses classes and native methods from NCSA's Java HDF Interface project. Basically all conversions from native types to Java types is done through code from the NCSA HDF group. Without this code the implementation of this API would have taken much longer. See NCSA's copyright for more information.

Installation

Requirements

Caution

Documentation is old and may need revision.

For running an application with `jnexus` an recent Java runtime environment (JRE) will do.

In order to compile the Java API for NeXus a Java Development Kit is required on top of the build requirements for the C API.

Installation under Windows

1. Copy the HDF DLL's and the file `jnexus.dll` to a directory in your path. For instance `C:\\Windows\\system32`.
2. Copy the `jnexus.jar` to the place where you usually keep library jar files.

Note that the location or the naming of these files in the binary Nexus distributions have changed over the years. In the Nexus 4.3.0 Windows 64-bit distribution (see Assets in https://github.com/nexusformat/code/releases/tag/4.3.0), By default, the DLL is at: `C:\Program Files\NeXus Data Format\bin\libjnexus-0.dll`. Please rename this file to `jnexus.dll` before making it available in your path. This is important, otherwise, JVM runtime will not be able to locate this file.

For the same distribution, the location of `jnexus.jar` is at: `C:\Program Files\NeXus Data Format\share\java`.

Installation under Unix

The `jnexus.so` shared library as well as all required file backend `.so` libraries are required as well as the `jnexus.jar` file holding the required Java classes. Copy them wherever you like and see below for instructions how to run programs using `jnexus`.

Running Programs with the NeXus API for Java

In order to successfully run a program with `jnexus`, the Java runtime systems needs to locate two items:

1. The shared library implementing the native methods.
2. The `nexus.jar` file in order to find the Java classes.

Locating the shared libraries

The methods for locating a shared library differ between systems. Under Windows32 systems the best method is to copy the `jnexus.dll` and the HDF4, HDF5 and/or XML-library DLL files into a directory in your path.

On a UNIX system, the problem can be solved in three different ways:

1. Make your system administrator copy the `jnexus.so` file into the systems default shared library directory (usually `/usr/lib` or `/usr/local/lib`).
2. Put the `jnexus.so` file wherever you see fit and set the `LD_LIBRARY_PATH` environment variable to point to the directory of your choice.
3. Specify the full pathname of the jnexus shared library on the java command line with the `-Dorg.nexusformat.JNEXUSLIB=full-path-2-shared-library` option.

Locating `jnexus.jar`

This is easier, just add the the full pathname to `jnexus.jar` to the classpath when starting java. Here are examples for a UNIX shell and the Windows shell.

UNIX example shell script to start `jnexus.jar`

```
#!/sbin/sh
java -classpath /usr/lib/classes.zip:../jnexus.jar:. \
    -Dorg.nexusformat.JNEXUSLIB=../libjnexus.so TestJapi
```

Windows 32 example batch file to start `jnexus.jar`

```
set JL=-Dorg.nexusformat.JNEXUSLIB=..\jnexus\bin\win32\jnexus.dll
java -classpath C:\jdk1.5\lib\classes.zip;..\jnexus.jar;. %JL% TestJapi
```

Programming with the NeXus API for Java

The NeXus C-API is good enough but for Java a few adaptions of the API have been made in order to match the API better to the idioms used by Java programmers. In order to understand the Java-API, it is useful to study the NeXus C-API because many methods work in the same way as their C equivalents. A full API documentation is available in Java documentation format. For full reference look especially at:

- The interface `NeXusFileInterface` first. It gives an uncluttered view of the API.
- The implementation `NexusFile` which gives more details about constructors and constants. However this documentation is interspersed with information about native methods which should not be called by an application programmer as they are not part of the standard and might change in future.

See the following code example for opening a file, opening a vGroup and closing the file again in order to get a feeling for the API:

fragment for opening and closing

```
    try{
        NexusFile nf = new NexusFile(filename, NexusFile.NXACC_READ);
        nf.opengroup("entry1","NXentry");
        nf.finalize();
    }catch(NexusException ne) {
        // Something was wrong!
    }
```

Some notes on this little example:

- Each NeXus file is represented by a `NexusFile` object which is created through the constructor.
- The `NexusFile` object takes care of all file handles for you. So there is no need to pass in a handle anymore to each method as in the C language API.
- All error handling is done through the Java exception handling mechanism. This saves all the code checking return values in the C language API. Most API functions return void.

- Closing files is tricky. The Java garbage collector is supposed to call the finalize method for each object it decides to delete. In order to enable this mechanism, the `NXclose()` function was replaced by the `finalize()` method. In practice it seems not to be guaranteed that the garbage collector calls the `finalize()` method. It is safer to call `finalize()` yourself in order to properly close a file. Multiple calls to the `finalize()` method for the same object are safe and do no harm.

Data Writing and Reading

Again a code sample which shows how this looks like:

fragment for writing and reading

```
    int idata[][] = new idata[10][20];
    int iDim[] = new int[2];

    // put some data into idata.......

    // write idata
    iDim[0] = 10;
    iDim[1] = 20;
    nf.makedata("idata",NexusFile.NX_INT32,2,iDim);
    nf.opendata("idata");
    nf.putdata(idata);

    // read idata
    nf.getdata(idata);
```

The dataset is created as usual with `makedata()` and opened with `putdata()`. The trick is in `putdata()`. Java is meant to be type safe. One would think then that a `putdata()` method would be required for each Java data type. In order to avoid this, the data to `write()` is passed into `putdata()` as type `Object`. Then the API proceeds to analyze this object through the Java introspection API and convert the data to a byte stream for writing through the native method call. This is an elegant solution with one drawback: An array is needed at all times. Even if only a single data value is written (or read) an array of length one and an appropriate type is the required argument.

Another issue are strings. Strings are first class objects in Java. HDF (and NeXus) sees them as dumb arrays of bytes. Thus strings have to be converted to and from bytes when reading string data. See a writing example:

String writing

```
    String ame = "Alle meine Entchen";
    nf.makedata("string_data",NexusFile.NX_CHAR,
            1,ame.length()+2);
    nf.opendata("string_data");
    nf.putdata(ame.getBytes());
```

And reading:

String reading

```
byte bData[] = new byte[132];
nf.opendata("string_data");
nf.getdata(bData);
String string_data = new String(bData);
```

The aforementioned holds for all strings written as SDS content or as an attribute. SDS or vGroup names do not need this treatment.

Inquiry Routines

Let us compare the C-API and Java-API signatures of the `getinfo()` routine (C) or method (Java):

C API signature of `getinfo()`

```
/* C -API */
NXstatus NXgetinfo(NXhandle handle, int *rank, int iDim[],
                   int *datatype);
```

Java API signature of `getinfo()`

```
// Java
void getinfo(int iDim[], int args[]);
```

The problem is that Java passes arguments only by value, which means they cannot be modified by the method. Only array arguments can be modified. Thus `args` in the `getinfo()` method holds the rank and datatype information passed in separate items in the C-API version. For resolving which one is which, consult a debugger or the API-reference.

The attribute and vGroup search routines have been simplified using Hashtables. The `Hashtable` returned by `groupdir()` holds the name of the item as a key and the classname or the string SDS as the stored object for the key. Thus the code for a vGroup search looks like this:

vGroup search

```
nf.opengroup(group,nxclass);
h = nf.groupdir();
e = h.keys();
System.out.println("Found in vGroup entry:");
while(e.hasMoreElements())
{
    vname = (String)e.nextElement();
    vclass = (String)h.get(vname);
    System.out.println("     Item: " + vname + " class: " + vclass);
}
```

For an attribute search both at global or SDS level the returned Hashtable will hold the name as the key and a little class holding the type and size information as value. Thus an attribute search looks like this in the Java-API:

attribute search

```
    Hashtable h = nf.attrdir();
    Enumeration e = h.keys();
    while(e.hasMoreElements())
    {
        attname = (String)e.nextElement();
        atten = (AttributeEntry)h.get(attname);
        System.out.println("Found global attribute: " + attname +
            " type: "+ atten.type + " ,length: " + atten.length);
    }
```

For more information about the usage of the API routines see the reference or the NeXus C-API reference pages. Another good source of information is the source code of the test program which exercises each API routine.

Known Problems

These are a couple of known problems which you might run into:

Memory
As the Java API for NeXus has to convert between native and Java number types a copy of the data must be made in the process. This means that if you want to read or write 200MB of data your memory requirement will be 400MB! This can be reduced by using multiple `getslab()`/`putslab()` to perform data transfers in smaller chunks.

`Java.lang.OutOfMemoryException`
By default the Java runtime has a low default value for the maximum amount of memory it will use. This ceiling can be increased through the `-mxXXm` option to the Java runtime. An example: `java -mx512m ...` starts the Java runtime with a memory ceiling of 512MB.

Maximum 8192 files open
The NeXus API for Java has a fixed buffer for file handles which allows only 8192 NeXus files to be open at the same time. If you ever hit this limit, increase the `MAXHANDLE` define in `native/handle.h` and recompile everything.

On-line Documentation

The following documentation is browsable online:

1. The API source code
2. A verbose tutorial for the NeXus for Java API.
3. The API Reference.
4. Finally, the source code for the test driver for the API which also serves as a documented usage example.

4.3.5 NAPI IDL Interface

IDL is an interactive data evaluation environment developed by Research Systems - it is an interpreted language for data manipulation and visualization. The NeXus IDL bindings allow access to the NeXus API from within IDL - they are installed when NeXus is compiled from source after being configured with the following options:

```
configure \
        --with-idlroot=/path/to/idl/installation \
        --with-idldlm=/path/to/install/dlm/files/to
```

For further details see the README (https://htmlpreview.github.com/?https://github.com/nexusformat/code/blob/master/bindings/idl/README.html) for the NeXus IDL binding. The source code is stored at https://github.com/nexusformat/code/tree/master/bindings/idl

4.4 Utility API

The NeXus F90 Utility API provides a number of routines that combine the operations of various core API routines in order to simplify the reading and writing of NeXus files. At present, they are only available as a Fortran 90 module but a C version is in preparation.

The utility API comprises the following functional groups:

- Routines to read or write data.
- Routines to find whether or not groups, data, or attributes exist, and to find data with specific signal or axis attributes, i.e. to identify valid data or axes.
- Routines to open other groups to which `NXdata` items are linked, and to return again.

line required for use with F90 API

Any program using the F90 Utility API needs to put the following line near the top of the program:

```
use NXUmodule
```

Note

Do not put `USE` statements for both `NXmodule` and `NXUmodule`. The former is included in the latter

4.4.1 List of F90 Utility Routines

name	description
Reading and Writing	
`NXUwriteglobals`	Writes all the valid global attributes of a file.
`NXUwritegroup`	Opens a group (creating it if necessary).
`NXUwritedata`	Opens a data item (creating it if necessary) and writes data and its units.
`NXUreaddata`	Opens and reads a data item and its units.
`NXUwritehistogra`	Opens one dimensional data item (creating it if necessary) and writes histogram centers and their units.
`NXUreadhistogram`	Opens and reads a one dimensional data item and converts it to histogram bin boundaries.
`NXUsetcompress`	Defines the compression algorithm and minimum dataset size for subsequent write operations.
Finding Groups, Data, and Attributes	
`NXUfindclass`	Returns the name of a group of the specified class if it is contained within the currently open group.
`NXUfinddata`	Checks whether a data item of the specified name is contained within the currently open group.
`NXUfindattr`	Checks whether the currently open data item has the specified attribute.
`NXUfindsignal`	Searches the currently open group for a data item with the specified `SIGNAL` attribute.
`NXUfindaxis`	Searches the currently open group for a data item with the specified `AXIS` attribute.
Finding Linked Groups	
`NXUfindlink`	Finds another link to the specified NeXus data item and opens the group it is in.
`NXUresumelink`	Reopens the original group from which `NXUfindlink` was used.

Currently, the F90 utility API will only write character strings, 4-byte integers and reals, and 8-byte reals. It can read other integer sizes into four-byte integers, but does not differentiate between signed and unsigned integers.

4.5 Building Programs

The install kit provides a utility call `nxbuild` that can be used to build simple programs:

```
nxbuild -o test test.c
```

This script links in the various libraries for you and reading its contents would provide the necessary information for creating a separate Makefile. You can also use `nxbuild` with the example files in the NeXus distribution kit which are installed into `/usr/local/nexus/examples`

Note that the executable name is important in this case as the test program uses it internally to determine the `NXACC_CREATE*` argument to pass to `NXopen`.

building and running a simple NeXus program

```
#  builds HDF5 specific test
nxbuild -o napi_test-hdf5 napi_test.c

# runs the test
./napi_test-hdf5
```

NeXus is also set up for pkg-config so the build can be done as:

```
gcc `pkg-config --cflags` `pkg-config --libs` -o test test.c
```

4.6 Reporting Bugs in the NeXus API

If you encounter any bugs in the installation or running of the NeXus API, please report them online using our Issue Reporting system. (https://www.nexusformat.org/IssueReporting.html)

CHAPTER

FIVE

NEXUS COMMUNITY

NeXus began as a group of scientists with the goal of defining a common data storage format to exchange experimental results and to exchange ideas about how to analyze them.

The NeXus Scientific Community provides the scientific data, advice, and continued involvement with the NeXus standard. NeXus provides a forum for the scientific community to exchange ideas in data storage.

The NeXus International Advisory Committee (NIAC) supervises the development and maintenance of the NeXus common data format for neutron, X-ray, and muon science through the NeXus class definitions and oversees the maintenance of the NeXus Application Programmer Interface (NAPI) as well as the technical infrastructure.

There are several mechanisms in place in order to coordinate the development of NeXus with the larger community.

5.1 NeXus Webpage

First of all, there is the NeXus webpage, https://www.nexusformat.org/, which provides all kinds of information, including membership, minutes, and discussions from the meetings of the NIAC, Code Camps, and Tele Conferences, as well as some proposed designs for consideration by NeXus.

The webpage is kept with a number of other repositories in the nexusformat.org Github organisation https://github.com/nexusformat/. As for all of these repositories, pull requests to correct or improve the content or code are always welcome!

5.2 Contributed Definitions

The community is encouraged to provide new definitions (*Base Class Definitions* or *Application Definitions*) for consideration in the NeXus standard. These community contributions will be entered in the *Contributed Definitions* and will be curated according to procedures set forth by the *NIAC: The NeXus International Advisory Committee*.

5.3 Other Ways NeXus Coordinates with the Scientific Community

5.3.1 NIAC: The NeXus International Advisory Committee

The purpose of the NeXus International Advisory Committee (NIAC)[1] is to supervise the development and maintenance of the NeXus common data format for neutron, X-ray, and muon science. This purpose includes, but is not limited to, the following activities.

[1] For more details about the NIAC constitution, procedures, and meetings, refer to the NIAC web page: https://www.nexusformat.org/NIAC.html The members of the NIAC may be reached by email: nexus-committee@nexusformat.org

1. To establish policies concerning the definition, use, and promotion of the NeXus format.
2. To ensure that the specification of the NeXus format is sufficiently complete and clear for its use in the exchange and archival of neutron, X-ray, and muon data.
3. To receive and examine all proposed amendments and extensions to the NeXus format. In particular, to ratify proposed instrument and group class definitions, to ensure that the data structures conform to the basic NeXus specification, and to ensure that the definitions of data items (fields) are clear and unambiguous and conform to accepted scientific usage.
4. To ensure that documentation of the NeXus format is sufficient, current, and available to potential users both on the internet and in other forms.
5. To coordinate the maintenance of the NeXus Application Programming Interface and to promote other software development that will benefit users of the NeXus format.
6. To coordinate with other organizations that maintain and develop related data formats to reach compatibility.

The committee will meet at least once every other calendar year according to the following plan:

- In years coinciding with the NOBUGS series of conferences (once every two years), members of the entire NIAC will meet as a satellite meeting to NOBUGS, along with interested members of the community.
- In intervening years, the executive officers of the NIAC will attend, along with interested members of the NIAC. This is intended to be a working meeting with a small group.

5.3.2 NeXus Mailing List

We invite anyone who is associated with neutron and/or X-ray synchrotron science and who wishes to be involved in the development and testing of the NeXus format to subscribe to this list. It is a public list for the free discussion of all aspects of the design and operation of the NeXus format.

- List Address: nexus@nexusformat.org
- Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus
- Archive: http://lists.nexusformat.org/pipermail/nexus

Note

Subscription to nexus@nexusformat.org does not subscribe you automatically to any other NeXus mailing list.

5.3.3 NeXus International Advisory Committee (NIAC) Mailing List

This list contains discussions of the *NIAC: The NeXus International Advisory Committee*, which oversees the development of the NeXus data format. Its members represent many of the major neutron and synchrotron scattering sources in the world. Membership and posting to this list are limited to the committee members, but the archives are public. The NIAC mailing list is for communications specific to NIAC and not for public contribution. General discussions should be held in the public mailing list.

- List Address: nexus-committee@nexusformat.org
- Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-committee
- Archive: http://lists.nexusformat.org/pipermail/nexus-committee

Note

Subscription to nexus-committee@nexusformat.org does not subscribe you automatically to any other NeXus mailing list.

5.3.4 NeXus Video Conference Announcements

There are video conferences on NeXus roughly twice a month. Agenda and joining details are posted on the webpage: https://www.nexusformat.org/Teleconferences.html In addition calendar invites are sent to this list. NeXus-Tech used to be used for discussions in the past. Now the list is moderated to only allow communication related to holding meetings. All other traffic should go to the main list nexus@nexusformat.org

- List Address: nexus-tech@nexusformat.org
- Subscriptions: http://lists.nexusformat.org/mailman/listinfo/nexus-tech

5.3.5 NeXus Developers Mailing List (retired)

This mailing list was for discussions concerning the technical development of NeXus (the Definitions, NXDL, and the NeXus Application Program Interface). There was, however, much overlap with the general NeXus mailing list and so this separate list was closed in October 2012, but the archive of previous posting is still available.

- Archive: http://lists.nexusformat.org/pipermail/nexus-developers

5.3.6 NeXus Repositories

NeXus NXDL class definitions (both base classes and application definitions) and the NeXus code library source are held in a pair of git repositories on GitHub.

The repositories are world readable. You can browse them directly:

NeXus code library and applications
https://github.com/nexusformat/code

NeXus NXDL class definitions
https://github.com/nexusformat/definitions

NeXus GitHub *organization*
https://github.com/nexusformat

If you would like to contribute (thank you!), the normal GitHub procedure of forking the repository and generating pull requests should be used.

Please report any problems via the *Issue Reporting* system.

5.3.7 NeXus Issue Reporting

NeXus is using GitHUb (https://github.com) as source code repository and for problem reporting. The issue reports (see *View current issues* below) are used to guide the NeXus developers in resolving problems as well as implementing new features.

NeXus Code (NAPI, Library, and Applications)

Report a new issue
https://github.com/nexusformat/code/issues/new

View current issues
https://github.com/nexusformat/code/issues

Timeline (recent ticket and code changes)
https://github.com/nexusformat/code/pulse

NeXus Definitions (NXDL base classes and application definitions)

Report a new issue
https://github.com/nexusformat/definitions/issues/new

View current issues
https://github.com/nexusformat/definitions/issues

Timeline (recent ticket and definition changes)
https://github.com/nexusformat/definitions/pulse

CHAPTER

SIX

INSTALLATION

This section describes how to install the NeXus API and details the requirements. The NeXus API is distributed under the terms of the GNU Lesser General Public License version 3.

The source distribution of NAPI can be downloaded from the release page of the associated GitHub project. Instructions how to build the code can be found in the *INSTALL.rst* file shipped with the source distribution. In case you need help, feel free to contact the NeXus mailing list: http://lists.nexusformat.org/mailman/listinfo/nexus

6.1 Precompiled Binary Installation

6.1.1 Linux RPM Distribution Kits

An installation kit (source or binary) can be downloaded from: https://github.com/nexusformat/code/releases/tag/4.3.0

A NeXus binary RPM (nexus-*.i386.rpm) contains ready compiled NeXus libraries whereas a source RPM (nexus-*.src.rpm) needs to be compiled into a binary RPM before it can be installed. In general, a binary RPM is installed using the command

```
rpm -Uvh file.i386.rpm
```

or, to change installation location from the default (e.g. /usr/local) area, using

```
rpm -Uvh --prefix /alternative/directory file.i386.rpm
```

If the binary RPMS are not the correct architecture for you (e.g. you need x86_64 rather than i386) or the binary RPM requires libraries (e.g. HDF4) that you do not have, you can instead rebuild a source RPM (.src.rpm) to generate the correct binary RPM for you machine. Download the source RPM file and then run

```
rpmbuild --rebuild file.src.rpm
```

This should generate a binary RPM file which you can install as above. Be careful if you think about specifying an alternative buildroot for rpmbuild by using `--buildroot` option as the "buildroot" directory tree will get remove (so `--buildroot /` is a really bad idea). Only change buildroot it if the default area turns out not to be big enough to compile the package.

If you are using Fedora, then you can install all the dependencies by typing

```
yum install hdf hdf-devel hdf5 hdf5-devel mxml mxml-devel
```

6.1.2 Microsoft Windows Installation Kit

A Windows MSI based installation kit is available and can be downloaded from: https://github.com/nexusformat/code/releases/tag/4.3.0

6.1.3 Mac OS X Installation Kit

An installation disk image (.dmg) can be downloaded from: https://github.com/nexusformat/code/releases/tag/4.3.0

6.2 Source Installation

6.2.1 NeXus Source Code Distribution

The source code distribution can be obtained from GitHub. One can either checkout the git repositories to get access to the most recent development code. To clone the definitions repository use

```
$ git clone https://github.com/nexusformat/definitions.git definitions
```

or for the NAPI

```
$ git clone https://github.com/nexusformat/code.git code
```

For release tarballs go to the release page for the NAPI or the definitions. For the definitions it is currently recommended to work directly with the Git repository as the actual release is rather outdated.

Instructions how to build the NAPI code can be found either on the GitHub project website or in the *README.rst* file shipped with the source distribution.

6.3 Releases

The NeXus definitions are expected to evolve. The evolution is marked as a series of *releases* which are snapshots of the repository (and current state of the NeXus standard). Each new *release* of the definitions will be posted to the definitions GitHub repository and announced to the community via the NeXus mailing list: *nexus@nexusformat.org*

6.3.1 NeXus definitions

Releases of the NeXus definitions are listed on the GitHub web site: https://github.com/nexusformat/definitions/releases

Release Notes

Detailed notes about each release (start with v3.3) are posted to the definitions GitHub wiki: https://github.com/nexusformat/definitions/wiki/Release-Notes

Release Process

The process to make a new release of the NeXus definitions repository is documented in the repository's GitHub wiki: https://github.com/nexusformat/definitions/wiki/Release-Procedure.

The release process starts by creating a GitHub [Milestone](https://help.github.com/articles/tracking-the-progress-of-your-work-with-milestones/) for the new release. Milestones for the NeXus definitions repository are available on the GitHub site: https://github.com/nexusformat/definitions/milestones

Versioning (Tags)

Versioning of each of the NXDL files, as well as the complete set of NXDL files is now described in the wiki[1] of the NeXus definitions repository[2]. Please see that wiki for complete information.

In case you need help, feel free to contact the *NeXus Mailing List*:

Archives
http://lists.nexusformat.org/mailman/listinfo/nexus

email
nexus@nexusformat.org

[1] Release Procedure: https://github.com/nexusformat/definitions/wiki/Release-Procedure
[2] Definitions repository: https://github.com/nexusformat/definitions

CHAPTER

SEVEN

NEXUS UTILITIES

There are many utilities available to read, browse, write, and use NeXus data files. Some are provided by the NeXus technical group while others are provided by the community. Still, other tools listed here can read or write one of the low-level file formats used by NeXus (HDF5, HDF4, or XML).

Furthermore, there are specific examples of code that can read, write, (or both) NeXus data files, given in the section *Language APIs for NeXus and HDF5*.

The NIAC welcomes your continued contributions to this documentation.

Please note that NeXus maintains a repository of example data files[1] which you may browse and download. There is a cursory analysis[2] of every file in this repository as to whether it can be read as HDF5 or NeXus HDF5. The analysis code[3], which serves as yet another example reader, is made using python and h5py.

7.1 Utilities supplied with NeXus

Most of these utility programs are run from the command line. It will be noted if a program provides a graphical user interface (GUI). Short descriptions are provided here with links to further information, as available.

nxbrowse
NeXus Browser

nxconvert
Utility to convert a NeXus file into HDF4/HDF5/XML/…

nxdir
`nxdir` is a utility for querying a NeXus file about its contents. Full documentation can be found by running this command:

```
nxdir -h
```

nxingest
`nxingest` extracts the metadata from a NeXus file to create an XML file according to a mapping file. The mapping file defines the structure (names and hierarchy) and content (from either the NeXus file, the mapping file or the current time) of the output file. See the man page for a description of the mapping file. This tool uses the NAPI. Thus, any of the supported formats (HDF4, HDF5 and XML) can be read.

nxsummary
Use `nxsummary` to generate summary of a NeXus file. This program relies heavily on a configuration file. Each `item` tag in the file describes a node to print from the NeXus file. The `path` attribute describes where in the NeXus file to get information from. The `label` attribute will be printed when showing the value of the specified

[1] https://github.com/nexusformat/exampledata
[2] https://github.com/nexusformat/exampledata/blob/master/critique.md
[3] https://github.com/nexusformat/exampledata/blob/master/critique.py

field. The optional `operation` attribute provides for certain operations to be performed on the data before printing out the result. See the source code documentation for more details.

nxtranslate
`nxtranslate` is an anything to NeXus converter. This is accomplished by using translation files and a plugin style of architecture where `nxtranslate` can read from new formats as plugins become available. The documentation for `nxtranslate` describes its usage by three types of individuals:

- the person using existing translation files to create NeXus files
- the person creating translation files
- the person writing new *retrievers*

All of these concepts are discussed in detail in the documentation provided with the source code.

NXplot
An extendable utility for plotting any NeXus file. `NXplot` is an Eclipse-based GUI project in Java to plot data in NeXus files. (The project was started at the first NeXus Code Camp in 2009.)

7.2 Validation

The list of applications below are for *validating* NeXus files. The list is not intended to be a complete list of all available packages.

cnxvalidate
NeXus validation tool written in C (not via NAPI).

Its dependencies are libxml2 and the HDF5 libraries, version 1.8.9 or better. Its purpose is to validate HDF5 files against NeXus application definitions.

See the program documentation for more details: https://github.com/nexusformat/cnxvalidate.git

punx
Python Utilities for NeXus HDF5 files

punx can validate both NXDL files and NeXus HDF5 data files, as well as print the structure of any HDF5 file, even non-NeXus files.

NOTE: project is under initial construction, not yet released for public use, but is useful in its present form (version 0.2.5).

punx can show the tree structure of any HDF5 file. The output is more concise than that from *h5dump*.

See the program documentation for more details: https://punx.readthedocs.io

7.3 Other Utilities

NeXus Constructor (https://github.com/ess-dmsc/nexus-constructor)
The NeXus Constructor facilitates constructing NeXus files in which to record data from experiments at neutron science facilities. This includes all supporting metadata typically required to perform analysis of such experiments, including instrument geometry information.

nxdl_to_hdf5.py (https://github.com/nexusformat/exampledata/tree/master/nxdl)
`nxdl_to_hdf5.py` is a Python script that reads the NeXus definition files (files ending with `.nxdl.xml`) and creates example Python scripts as well as HDF5 files for each definition. There are generated example scripts of each application definition for both *h5py* and *nexusformat*. Currently, only application definitions and some contributed_definitions are supported as the code depends on the existence of an *NXentry* in the definition.

7.4 Data Analysis

The list of applications below are some of the utilities that have been developed (or modified) to read/write NeXus files as a data format. It is not intended to be a complete list of all available packages.

DAVE (http://www.ncnr.nist.gov/dave/)
DAVE is an integrated environment for the reduction, visualization and analysis of inelastic neutron scattering data. It is built using IDL (Interactive Data Language) from ITT Visual Information Solutions.

DAWN (http://www.dawnsci.org)
The Data Analysis WorkbeNch (DAWN) project is an eclipse based workbench for doing scientific data anaylsis. It offers generic visualisation, and domain specific processing.

GDA (http://www.opengda.org)
The GDA project is an open-source framework for creating customised data acquisition software for science facilities such as neutron and X-ray sources. It has elements of the DAWN analysis workbench built in.

Gumtree (https://archive.ansto.gov.au/ResearchHub/OurInfrastructure/ACNS/Facilities/Computing/GumTree/index.htm)
Gumtree is an open source project, providing a graphical user interface for instrument status and control, data acquisition and data reduction.

IDL (https://www.harrisgeospatial.com/docs/using_idl_home.html)
IDL is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation.

IgorPro (http://www.wavemetrics.com/)
IGOR Pro is an extraordinarily powerful and extensible scientific graphing, data analysis, image processing and programming software tool for scientists and engineers.

ISAW (ftp://ftp.sns.gov/ISAW/)
The Integrated Spectral Analysis Workbench software project (ISAW) is a Platform-Independent system Data Reduction/Visualization. ISAW can be used to read, manipulate, view, and save neutron scattering data. It reads data from IPNS run files or NeXus files and can merge and sort data from separate measurements.

LAMP (http://www.ill.eu/data_treat/lamp/>)
LAMP (Large Array Manipulation Program) is designed for the treatment of data obtained from neutron scattering experiments at the Institut Laue-Langevin. However, LAMP is now a more general purpose application which can be seen as a GUI-laboratory for data analysis based on the IDL language.

Mantid (http://www.mantidproject.org/)
The Mantid project provides a platform that supports high-performance computing on neutron and muon data. It is being developed as a collaboration between Rutherford Appleton Laboratory and Oak Ridge National Laboratory.

MATLAB (http://www.mathworks.com/)
MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation.

NeXpy (http://nexpy.github.io/nexpy/)
The goal of NeXpy is to provide a simple graphical environment, coupled with Python scripting capabilities, for the analysis of X-Ray and neutron scattering data. (It was decided at the NIAC 2010 meeting that a large portion of this code would be adopted in the future by NeXus and be part of the distribution)

silx (http://www.silx.org/doc/silx/latest/)
The silx project aims to provide a collection of Python packages to support the development of data assessment, reduction and analysis at synchrotron radiation facilities. In particular it provides tools to read, write and visualize NeXus HDF5 files.

OpenGENIE (http://www.opengenie.org/)
A general purpose data analysis and visualisation package primarily developed at the ISIS Facility, Rutherford Appleton Laboratory.

PyMCA (http://pymca.sourceforge.net/)
PyMca is a ready-to-use, and in many aspects state-of-the-art, set of applications implementing most of the needs of X-ray fluorescence data analysis. It also provides a Python toolkit for visualization and analysis of energy-dispersive X-ray fluorescence data. Reads, browses, and plots data from NeXus HDF5 files.

spec2nexus (https://spec2nexus.readthedocs.io)
(Python code) Converts SPEC data files and scans into NeXus HDF5 files. (Note the *h5toText* tool mentioned here previously is no longer available from the *spec2nexus* project. The code has been moved into the *punx* project: https://punx.readthedocs.io/.)

spec2nexus provides libraries:

- *spec2nexus.spec*: python binding to read SPEC[4] data files
- *spec2nexus.eznx*: (Easy NeXus) supports writing NeXus HDF5 files using h5py

7.5 HDF Tools

Here are some of the generic tools that are available to work with HDF files. In addition to the software listed here there are also APIs for many programming languages that will allow low level programmatic access to the data structures.

h5wasm (https://github.com/usnistgov/h5wasm):

A WebAssembly port of the HDF5 C library, which allows reading and writing HDF5 files from JavaScript (i.e. no need for a back-end server at all).

H5Web (https://github.com/silx-kit/h5web):

H5Web is a toolkit for exploring and visualising HDF5 files and, more generally, for visualizing data. It is based on React, and WebGL. These projects make use of H5Web:

- jupyterlab-h5web: extension for JupyterLab
- vscode-h5web: extension for Microsoft Visual Studio Code Editor
- On-line visualization with NeXus file (using `h5wasm`): simple_example_basic.nexus.hdf5
- H5Web demonstration site

HDF Group tools (https://portal.hdfgroup.org/display/support/Downloads)
Various tools are available from the HDF Group. These are usually shipped with the HDF5 kits but are also available for download separately. The HDF5 source code (https://github.com/HDFGroup/hdf5) is available on GitHub.

HDFexplorer (http://www.space-research.org/)
A data visualization program that reads Hierarchical Data Format files (HDF, HDF-EOS and HDF5) and also netCDF data files.

HDFview (http://www.hdfgroup.org)
A Java based GUI for browsing (and some basic plotting) of HDF files.

tiled (https://blueskyproject.io/tiled/)
A *data access service* for data-aware portals and data science tools, provides a way to browse and visualize HDF5 files.

[4] SPEC: http://www.certif.com

7.6 Language APIs for NeXus and HDF5

Collected here are some of the tools identified[5] as a result of a simple question asked at the 2018 Nobugs conference: *Are there examples of code that reads NeXus data?* Some of these are very specific to an instrument or application definition while others are more generic. The lists below are organized by programming language, yet some collections span several languages so they are listed in the section *Language API: mixed*.

Note these example listed in addition to the many examples described here in the manual, in section :*Examples*.

7.6.1 Language API: *F77*

- **POLDI**: `poldi.zip`[6] contains: - A F77 reading routine using NAPI for POLDI at SINQ PSI - an example of a file which it reads

7.6.2 Language API: *IDL*

- **aXis2000**[7], with the NeXus-specific IDL code in the `read_nexus.pro`[8], reads *NXstxm*

7.6.3 Language API: *IgorPro*

- **HDF5gateway**[9] makes it easy to read a HDF5 file (including NeXus) into an IgorPro[10] folder, including group and dataset attributes, such as a NeXus data file, modify it, and then write the folder structure back out.

7.6.4 Language API: *Java*

- **Dawn**[11] has java code to read[12] and write[13] HDF5 NeXus files (generic NeXus, not tied to specific application definitions).
- `NXreader.zip`[14] is java code which reads NeXus files into **ImageJ.** It uses the Java-hdf interface to HDF5. It tries to do a good job locating the image dataset by NeXus conventions. But it uses the old style conventions.

[5] https://github.com/nexusformat/definitions/issues/630
[6] https://github.com/nexusformat/definitions/files/4107360/poldi.zip
[7] http://unicorn.chemistry.mcmaster.ca/aXis2000.html
[8] `read_nexus.pro`: http://unicorn.chemistry.mcmaster.ca/axis/aXis2000.zip
[9] https://github.com/prjemian/hdf5gateway
[10] IgorPro: https://wavemetrics.com
[11] https://dawnsci.org/
[12] read: https://github.com/DawnScience/scisoft-core/blob/master/uk.ac.diamond.scisoft.analysis/src/uk/ac/diamond/scisoft/analysis/io/NexusHDF5Loader.java
[13] write: https://github.com/DawnScience/dawnsci/blob/master/org.eclipse.dawnsci.hdf5/src/org/eclipse/dawnsci/hdf5/nexus/NexusFileHDF5.java
[14] https://github.com/nexusformat/definitions/files/4107439/NXreader.zip

7.6.5 Language API: *Python*

- **Dials**[15] has python (and some C++) code for reading *NXmx*[16]
  - *cctbx.xfel* code for writing[17] *NXmx* master files for JF16M at SwissFEL
- **h5py**[18]
  HDF5 for Python (h5py) is a general-purpose Python interface to HDF5.
- **Mantis**[19], with NeXus-specific python code[20], reads *NXstxm*
- **nexusformat**[21] **NeXus package for Python**
  Provides an API to open, create, plot, and manipulate NeXus data.
- **SasView**[22] has python code to read[23] and write[24] *NXcanSAS*

7.6.6 Language API: *mixed*

- **FOCUS**: `focus.zip`[25] contains:
  - An example FOCUS file
  - focusreport: A h5py program which skips through a list of files and prints statistics
  - focusreport.tcl, same as above but in Tcl using the Swig generated binding to NAPI
  - i80.f contains a F77 routine for reading FOCUS files into Ida. The routine is RRT_in_Foc.
- **ZEBRA**: `zebra.zip`[26] contains:
  - an example file
  - zebra-to-ascii, a h5py script which dumps a zebra file to ASCII
  - `TRICSReader.*` for reading ZEBRA files in C++ using C-NAPI calls

[15] https://dials.github.io/
[16] read: https://github.com/cctbx/dxtbx/blob/master/format/nexus.py
[17] write: https://github.com/cctbx/cctbx_project/blob/master/xfel/swissfel/jf16m_cxigeom2nexus.py
[18] http://docs.h5py.org
[19] Mantis: http://spectromicroscopy.com/
[20] python code: https://bitbucket.org/mlerotic/spectromicroscopy/src/default/
[21] https://github.com/nexpy/nexusformat
[22] https://www.sasview.org/
[23] read: https://github.com/SasView/sasview/blob/master/src/sas/sascalc/dataloader/readers/cansas_reader_HDF5.py
[24] write: https://github.com/SasView/sasview/blob/master/src/sas/sascalc/file_converter/nxcansas_writer.py
[25] https://github.com/nexusformat/definitions/files/4107386/focus.zip
[26] https://github.com/nexusformat/definitions/files/4107416/zebra.zip

CHAPTER

EIGHT

BRIEF HISTORY OF NEXUS

Two things to note about the development and history of NeXus:

- All efforts on NeXus have been voluntary except for one year when we had one full-time worker.
- The NIAC has already discussed many matters related to the format.

2022-07:

- *release v2022.06 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2022.06>* of NeXus Definitions

2020-10:

- *release v2020.10 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2020.10>* of NeXus Definitions

2020-01:

- *release v2020.1 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2020.1>* of NeXus Definitions

2018-05:

- *release v2018.5 <https://github.com/nexusformat/definitions/wiki/releasenotes__v2018.5>* of NeXus Definitions
- **#597**
  changed versioning scheme and procedures

2017-07:
release 3.3 <https://github.com/nexusformat/definitions/wiki/releasenotes__v3.3> of NeXus Definitions

2016-10:
release 3.2 <https://github.com/nexusformat/definitions/releases/tag/v3.2> of NeXus Definitions

2014-12:
The NIAC approves a new method to identify the default data to be plotted, applying attributes at the group level to the root of the HDF5 tree, and the NXentry and NXdata groups. See the description in *Associating plottable data using attributes applied to the NXdata group* and the proposal: https://www.nexusformat.org/2014_How_to_find_default_data.html

2012-05:
first release (3.1.0) of NXDL (NeXus Definition Language)

2010-01:
NXDL presented to ESRF HDF5 workshop on hyperspectral data

2009-09:
NXDL and draft NXsas (base class) presented to canSAS at SAS2009 conference

2009-04:
NeXus API version 4.2.0 is released with additional C++, IDL, and python/numpy interfaces.

2008-10:
NXDL: The NeXus Definition Language is defined. Until now, NeXus used another XML format, meta-DTD, for defining base classes and application definitions. There were several problems with meta-DTD, the biggest one being that it was not easy to validate against it. NXDL was designed to circumvent these problems. All current base classes and application definitions were ported into the NXDL.

2007-10:
NeXus API version 4.1.0 is released with many bug-fixes.

2007-05:
NeXus API version 4.0.0 is released with broader support for scripting languages and the feature to link with external files.

2005-07:
The community asked the NeXus team to provide an ASCII based physical file format which allows them to edit their scientific results in emacs. This lead to the development of a XML NeXus physical format. This was released with NeXus API version 3.0.0.

2003-10:
In 2003, NeXus had arrived at a stage where informal gatherings of a group of people were no longer good enough to oversee the development of NeXus. This lead to the formation of the NeXus International Advisory Committee (NIAC) which strives to include representatives of all major stake holders in NeXus. A first meeting was held at CalTech. Since 2003, the NIAC meets every year to discuss all matters NeXus.

2003-06:
Przemek Klosowski, Ray Osborn, and Richard Riedel received the only known grant explicitly for working on NeXus from the Systems Integration for Manufacturing Applications (SIMA) program of the National Institute of Standards and Technology (NIST). The grant funded a person for one year to work on community wide infrastructure in NeXus.

2002-09:
NeXus API version 2.0.0 is released. This version brought support for the new version of HDF, HDF5, released by the HDF group. HDF4 imposed limits on file sizes and the number of objects in a file. These issues were resolved with HDF5. The NeXus API abstracted the difference between the two physical file formats away form the user.

2001-summer:
MLNSC at LANL started writing NeXus files to store raw data

1997-07:
SINQ at PSI started writing NeXus files to store raw data.

1996-10:
At *SoftNeSS 1996* (at ANL), after reviewing the different scientific data formats discussed, it was decided to use HDF4 as it provided the best grouping support. The basic structure of a NeXus file was agreed upon. the various data format proposals were combined into a single document by Przemek Klosowski (NIST), Mark Könnecke (then ISIS), Jonathan Tischler (ORNL and APS/ANL), and Ray Osborn (IPNS/ANL) coauthored the first proposal for the NeXus scientific data standard.[1]

1996-08:
The HDF-4 API is quite complex. Thus a NeXus Abstract Programmer Interface NAPI was released which simplified reading and writing NeXus files.

1995-09:
At *SoftNeSS 1995* (at NIST), three individual data format proposals by Przemek Klosowski (NIST), Mark Kön-

[1] https://www.nexusformat.org/pdfs/NeXus_Proposal.pdf

necke (then ISIS), and Jonathan Tischler (ORNL and APS/ANL) were joined to form the basis of the current NeXus format. At this workshop, the name *NeXus* was chosen.

1994-10:
Ray Osborn convened a series of three workshops called *SoftNeSS*. In the first meeting, Mark Könnecke and Jon Tischler were invited to meet with representatives from all the major U.S. neutron scattering laboratories at Argonne National Laboratory to discuss future software development for the analysis and visualization of neutron data. One of the main recommendations of *SoftNeSS'94* was that a common data format should be developed.

1994-08:
Jonathan Tischler (ORNL) proposed an HDF-based format[2] as a standard for data storage at APS

1994-06:
Mark Könnecke (then ISIS, now PSI) made a proposal using netCDF[3] for the European neutron scattering community while working at ISIS

[2] https://www.nexusformat.org/pdfs/Proposed_Data_Standard_for_the_APS.pdf
[3] https://www.nexusformat.org/pdfs/European-Formats.pdf

CHAPTER

NINE

ABOUT THESE DOCS

9.1 Authors

Pete R. Jemian, Documentation Editor:
<jemian@anl.gov>, Advanced Photon Source, Argonne National Laboratory, Argonne, IL, USA,

Frederick Akeroyd:
<freddie.akeroyd@stfc.ac.uk>, Rutherford Appleton Laboratory, Didcot, UK,

Stuart I. Campbell:
<campbellsi@ornl.gov>, Oak Ridge National Laboratory, Oak Ridge, TN, USA,

Przemek Klosowski:
<przemek.klosowski@nist.gov>, U. of Maryland and NIST, Gaithersburg, MD, USA,

Mark Könnecke:
<Mark.Koennecke@psi.ch>, Paul Scherrer Institut, CH-5232 Villigen PSI, Switzerland,

Ray Osborn:
<rosborn@anl.gov>, Argonne National Laboratory, Argonne, IL, USA,

Peter F. Peterson:
<petersonpf@ornl.gov>, Spallation Neutron Source, Oak Ridge, TN, USA,

Tobias Richter:
<Tobias.Richter@esss.se>, European Spallation Source, Lund, Sweden,

Joachim Wuttke:
<j.wuttke@fz-juelich.de>, Forschungszentrum Jülich, Jülich Centre for Neutron Science at Heinz Maier-Leibnitz Zentrum Garching, Germany.

9.2 Colophon

These docs (manual and reference) were produced using Sphinx (http://sphinx-doc.org). The source for the manual shows many examples of the structures used to create the manual. If you have any questions about how to contribute to this manual, please contact the NeXus Documentation Editor (Pete Jemian <jemian@anl.gov>).

Note

The indentation is very important to the syntax of the restructured text manual source. Be careful not to mix tabs and spaces in the indentation or the manual may not build properly.

9.3 Revision History

Browse the most recent Issues on the GitHub repository: https://github.com/nexusformat/definitions/pulse/monthly

9.4 Copyright and Licenses

Published by NeXus International Advisory Committee, https://www.nexusformat.org

Copyright (C) 1996-2026 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 Feb 19, 2026.

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*), 797
a (*field*), 526, 659, 793
a_axis_direction (*field*), 1038
a_b_c (*field*), 526
Abbe_number (*field*), 450
aberration (*base class*), *see* NXaberration, **196**
absorbing_material (*field*), 275, 392
absorption_cross_section (*field*), 224
acceleration_time (*field*), 465
acceptance_angle (*field*), 1070, 1073
accepted_photon_beam_divergence (*field*), 234
accepting_aperture (*field*), 240
accuracyN (*field*), 1144
acoustic_impedance (*field*), 1140
acquisition_mode (*field*), 325, 588
acquisition_time (*field*), 270, 1082, 1128
active_channel (*field*), 1117
active_channels (*field*), 876
active_frequency (*field*), 1135
activity (*base class*), *see* NXactivity, **197**
actuation_target (*field*), 199, 430–432, 705, 706
actuator (*base class*), *see* NXactuator, **198**
additional_detector_hardware (*field*), 749
additional_phase_information (*field*), 998
address (*field*), 529, 632, 1087
address, absolute, 21
address, relative, 21
aequatorial_angle (*field*), 773, 776, 777
affiliation (*field*), 529, 632, 695, 1087, 1162
afm (*application definition*), *see* NXafm, **872**
algorithm (*field*), 444, 553, 563–569, 631, 645, 649, 658, 878, 879, 882, 890, 891, 897, 903, 904, 910–912, 918–920, 948, 949, 960, 968, 975, 1007, 1021, 1050, 1060
alias (*field*), 197, 373, 553, 555, 558, 635–637, 639, 657
alpha (*field*), 241, 526, 659, 839, 892, 956
alpha_beta_gamma (*field*), 526
alpha_value_choice (*field*), 953
alpha_values (*field*), 954
amount (*field*), 269
amplifier (*base class*), *see* NXamplifier, **876**
amplifier_bias (*field*), 348
amplifier_type (*field*), 348, 701
amplifier_voltage (*field*), 348
amplitude_excitation (*field*), 875, 1133
amplitude_modulation_bandwidth (*field*), 1014
aN (*field*), 238
analyze_coprecipitation (*field*), 902
analyze_intersection (*field*), 902
analyze_proximity (*field*), 902
analyzer_dispersion (*field*), 724
analyzer_elevation (*field*), 723
analyzer_rotation (*field*), 723
analyzer_take_off_azimuth_angle (*field*), 847
analyzer_take_off_polar_angle (*field*), 846
angle (*field*), 197
angle_of_detection (*field*), 747
angle_of_in_plane_sample_rotation (*field*), 747
angle_of_incidence (*field*), 747
angle_of_incident_and_detection_beam (*field*), 747
angle_reference_frame (*field*), 746
angles (*field*), 404, 589
angular0 (*field*), 704, 713, 726
angular0_indices (*group attribute*), 726
angular1 (*field*), 704, 713, 726
angular1_indices (*group attribute*), 726
angular_acceptance (*field*), 274, 700, 724
angular_calibration (*field*), 325, 734
angular_calibration_applied (*field*), 325, 734
angular_dispersion (*field*), 1163
angular_spread (*field*), 623
animation_time (*field*), 1082, 1128
anode_material (*field*), 506
AOI_range (*field*), 979
aperture (*base class*), *see* NXaperture, **199**
aperture (*field*), 534
aperture_type (*field*), 207, 558
API, *see* NAPI
    F77; POLDI, 1209
    IDL; aXis2000, 1209
    IgorPro; HDF5gateway, 1209
    java; Dawn, 1209

java; NXreader.zip, 1209
mixed; FOCUS, 1210
mixed; ZEBRA, 1210
Python; Dials, 1210
Python; h5py, 1210
Python; Mantis, 1210
Python; nexusformat, 1210
Python; SasView, 1210
apm (*application definition*), *see* NXapm, **547**
apm_charge_state_analysis (*base class*), *see* NXapm_charge_state_analysis, **201**
apm_compositionspace_config (*application definition*), *see* NXapm_compositionspace_config, **878**
apm_compositionspace_results (*application definition*), *see* NXapm_compositionspace_results, **881**
apm_event_data (*base class*), *see* NXapm_event_data, **203**
apm_instrument (*base class*), *see* NXapm_instrument, **205**
apm_measurement (*base class*), *see* NXapm_measurement, **212**
apm_paraprobe_clusterer_config (*application definition*), *see* NXapm_paraprobe_clusterer_config, **889**
apm_paraprobe_clusterer_results (*application definition*), *see* NXapm_paraprobe_clusterer_results, **893**
apm_paraprobe_distancer_config (*application definition*), *see* NXapm_paraprobe_distancer_config, **896**
apm_paraprobe_distancer_results (*application definition*), *see* NXapm_paraprobe_distancer_results, **899**
apm_paraprobe_intersector_config (*application definition*), *see* NXapm_paraprobe_intersector_config, **901**
apm_paraprobe_intersector_results (*application definition*), *see* NXapm_paraprobe_intersector_results, **905**
apm_paraprobe_nanochem_config (*application definition*), *see* NXapm_paraprobe_nanochem_config, **908**
apm_paraprobe_nanochem_results (*application definition*), *see* NXapm_paraprobe_nanochem_results, **924**
apm_paraprobe_ranger_config (*application definition*), *see* NXapm_paraprobe_ranger_config, **943**
apm_paraprobe_ranger_results (*application definition*), *see* NXapm_paraprobe_ranger_results, **944**
apm_paraprobe_selector_config (*application definition*), *see* NXapm_paraprobe_selector_config, **945**
apm_paraprobe_selector_results (*application definition*), *see* NXapm_paraprobe_selector_results, **945**
apm_paraprobe_spatstat_config (*application definition*), *see* NXapm_paraprobe_spatstat_config, **946**
apm_paraprobe_spatstat_results (*application definition*), *see* NXapm_paraprobe_spatstat_results, **951**
apm_paraprobe_surfacer_config (*application definition*), *see* NXapm_paraprobe_surfacer_config, **953**
apm_paraprobe_surfacer_results (*application definition*), *see* NXapm_paraprobe_surfacer_results, **955**
apm_paraprobe_tessellator_config (*application definition*), *see* NXapm_paraprobe_tessellator_config, **959**
apm_paraprobe_tessellator_results (*application definition*), *see* NXapm_paraprobe_tessellator_results, **961**
apm_paraprobe_tool_common (*base class*), *see* NXapm_paraprobe_tool_common, **965**
apm_paraprobe_tool_config (*application definition*), *see* NXapm_paraprobe_tool_config, **967**
apm_paraprobe_tool_parameters (*base class*), *see* NXapm_paraprobe_tool_parameters, **971**
apm_paraprobe_tool_process (*base class*), *see* NXapm_paraprobe_tool_process, **973**
apm_paraprobe_tool_results (*application definition*), *see* NXapm_paraprobe_tool_results, **974**
apm_ranging (*base class*), *see* NXapm_ranging, **213**
apm_reconstruction (*base class*), *see* NXapm_reconstruction, **215**
apm_simulation (*base class*), *see* NXapm_simulation, **220**
AppDef-Apm-Definitions, 536
AppDef-Apm-Introduction, 536
AppDef-Diff-Definitions, 537
AppDef-Diff-Introduction, 537
AppDef-Em-Definitions, 537
AppDef-Em-Design, 537
AppDef-Em-Introduction, 537
AppDef-Imaging-Definitions, 542
AppDef-Imaging-Introduction, 542
AppDef-Mpes-Definitions, 543
AppDef-Mpes-Introduction, 543
AppDef-Opt-Spec-Definitions, 543
AppDef-Opt-Spec-Ellipsometry, 543
AppDef-Opt-Spec-Raman, 543
AppDef-Tof-Definitions, 544

AppDef-Tof-Introduction, 544
application definition, **535**
applied (*field*), 207, 237, 277, 282, 341, 344, 479, 558, 653, 655, 656
apply (*field*), 1055
archive (*application definition*), *see* NXarchive, **583**
area (*field*), 253, 263, 527, 795, 850, 930, 935, 1023, 1025, 1036
area_errors (*field*), 795
arpes (*application definition*), *see* NXarpes, **587**
arrival_time_pairs (*field*), 562
arrow_char (*field*), 1039
as (*field*), 797
as_errors (*field*), 797
aspect (*field*), 933
associated_beam (*field*), 696, 697, 707, 750
associated_source (*field*), 698, 699, 748
atom (*base class*), *see* NXatom, **220**
atom_types (*field*), 364, 555, 633, 659, 709, 756, 883, 998, 1006, 1012, 1048
attached_to (*field*), 498
attenuation (*field*), 1070
attenuator (*base class*), *see* NXattenuator, **223**
attenuator_transmission (*field*), 224, 732, 857, 1163
attribute, *see* field attribute, *see* group attribute, *see* file attribute, **143**, *see* NXDL attribute
  HDF, 59
  NXclass, 41
attribute (*NXDL element*), **146**
attributeType (*NXDL data type*), **154**
author (*field*), 443, 553, 631, 788, 1095, 1104
authors, **1215**
automatic plotting, *see* plotting
auxiliary_signals (*file attribute*), 314
auxiliary_signals (*group attribute*), 798
average_power (*field*), 228
average_value (*field*), 429, 1094, 1103
average_value_error (*field*), 429, 1094, 1103
average_value_errors (*field*), 429, 1094, 1103
axes (*attribute*), 57, 312
axes (*field attribute*), 317, 987
axes (*file attribute*), 315
axes (*group attribute*), 329, 352, 407, 561, 568, 569, 592, 597, 645–652, 659, 660, 708, 726, 758, 798, 884, 885, 987, 988, 1008, 1035, 1053, 1062, 1165
axis, 57
axis (*field attribute*), 316, 321, 322
axis_aic (*field*), 885
axis_angle_convention (*field*), 557, 635
axis_bic (*field*), 885
axis_dimension (*field*), 886
axis_energy (*field*), 365, 510–514, 649–652, 659
axis_explained_variance (*field*), 885
axis_feature_importance (*field*), 884
axis_feature_indices (*field*), 884
axis_i (*field*), 414–420, 510–514, 561, 562, 646–652, 660
axis_j (*field*), 415–417, 419, 420, 511, 513, 514, 561, 562, 646–652, 660
axis_k (*field*), 416, 417, 420, 511, 514, 646, 647, 649, 650, 652
axis_m (*field*), 417, 647
axis_mass_to_charge (*field*), 569
axis_pca_dimension (*field*), 885
axis_x (*field*), 362, 569, 659, 1033, 1034, 1047
axis_y (*field*), 362, 569, 659, 1033, 1047
axis_z (*field*), 568, 659, 1033
AXISNAME (*field*), 316, 523, 1123, 1147, 1150
AXISNAME_end (*field*), 524
AXISNAME_increment_set (*field*), 524
AXISNAME_indices (*file attribute*), 315
AXISNAME_indices (*group attribute*), 561, 568, 569, 645–652, 659, 660, 884, 885, 1008, 1035
AXISoffset_value (*field*), 1138
azimuth (*field*), 751, 796
azimuthal (*field*), 781
azimuthal_angle (*field*), 289, 323, 476, 688, 773, 776, 814, 817, 819, 1096, 1105
azimuthal_axis (*field*), 598
azimuthal_axis_edges (*field*), 598
azimuthal_width (*field*), 781
azint1d (*application definition*), *see* NXazint1d, **590**
azint2d (*application definition*), *see* NXazint2d, **595**

B

b (*field*), 526, 659, 793, 797
b_axis_direction (*field*), 1039
b_errors (*field*), 797
background (*field*), 570, 799
background_area (*field*), 797
background_area_interval (*field*), 797
background_mean (*field*), 475
backside_roughness (*field*), 624
backward_speedN (*field*), 1146
backward_sweep (*field*), 1130
bandwidth (*field*), 270, 424, 877
base class, **176**
base_pressure (*field*), 468
baseline_reference (*field attribute*), 1170, 1171
basicComponent (*NXDL data type*), **168**
BC-Apm-Classes, 185
BC-Apm-Introduction, 185
BC-Cgeometry-Base-Classes, 183
BC-Cgeometry-Introduction, 183
BC-Computer-Base-Classes, 181
BC-Core-Base-Classes, 177
BC-Danalysis-Base-Classes, 182

BC-Em-Classes, 186
BC-Em-Introduction, 186
BC-Mpes-Classes, 187
BC-Mpes-Introduction, 187
BC-MpesCommonBC, 187
BC-Opt-Spec-Introduction, 189
BC-Sample-Base-Classes, 181
BC-Tech-Base-Classes, 178
beam (*base class*), *see* NXbeam, **225**
beam_attenuation_factors (*field*), 1180
beam_azimuth_angle (*field*), 846
beam_center_derived (*field*), 733
beam_center_x (*field*), 324, 612, 733, 773, 776, 857, 987, 1173
beam_center_y (*field*), 324, 612, 734, 773, 776, 857, 987, 1173
beam_direction (*field*), 845
beam_evaluation (*field*), 789
beam_polar_angle_of_incidence (*field*), 845
beam_polarization_type (*field*), 748
beam_position (*field*), 339
beam_sample_relation (*field*), 753
beam_shape (*field*), 613
beam_size_x (*field*), 614
beam_size_y (*field*), 614
beam_splitter (*base class*), *see* NXbeam_splitter, **977**
beam_stop (*base class*), *see* NXbeam_stop, **231**
beam_transfer_matrix_table (*base class*), *see* NXbeam_transfer_matrix_table, **232**
BEAMdirection (*field*), 229
beamline (*field*), 1095, 1104
beamline_distance (*field*), 1004, 1017, 1080, 1090, 1111, 1114
bend_angle_x (*field*), 406, 434
bend_angle_y (*field*), 406, 434
bending_magnet (*base class*), *see* NXbending_magnet, **234**
bending_radius (*field*), 234
beta (*field*), 527, 659
bias_divider (*field*), 1014
bias_offset_value (*field*), 1122
bias_voltage (*field*), 1122
bibtex (*field*), 272
binary data, 48, *see* NX_BINARY
binary executable, *see* NAPI installation
binding_energy (*field*), 708
bins (*field*), 794
bit_depth_readout (*field*), 327, 735
bitdepth (*field*), 293, 565, 900, 925, 956, 963, 964, 969, 974, 975
blade_spacing (*field*), 275
blade_thickness (*field*), 275
blaze_wavelength (*field*), 1163
block (*field*), 1084
boundaries (*field*), 248, 927, 1061
boundary_conditions (*field*), 248, 928, 1061
boundary_contact (*field*), 1023, 1024, 1026, 1028
bounding_box (*field*), 474
bragg_angle (*field*), 289
brightness (*field*), 411
browser, 16, 1205
bunch_distance (*field*), 506
bunch_length (*field*), 506
bunge_euler (*field*), 1036, 1050, 1051, 1053, 1064
burgers_vector (*field*), 1049

C

C
    code examples, **83**
c (*field*), 526, 659, 793
c1 (*field*), 1052
c2 (*field*), 1052
c3 (*field*), 1052
calibrated_axis (*field*), 237, 707, 708, 758
calibrated_tof (*field*), 565
calibratedAXIS (*field*), 1136
calibration (*base class*), *see* NXcalibration, **236**
calibration_date (*field*), 324, 1136
calibration_name (*field*), 1136
calibration_status (*field*), 755
calibration_time (*field*), 755, 1088
calibration_type (*field*), 787, 1136
camera_length (*field*), 377
canSAS, 600
canSAS (*application definition*), *see* NXcanSAS, **600**
canSAS_class (*group attribute*), 602, 603, 610–612, 614–616
cantilever_coating (*field*), 1132
cantilever_config (*field*), 875
cantilever_length (*field*), 1133
cantilever_thickness (*field*), 1133
cantilever_type (*field*), 1132
cantilever_width (*field*), 1133
capability (*field*), 391
capillary (*base class*), *see* NXcapillary, **239**
capital_phi (*field*), 1045
cardinality (*field*), 262, 883, 894, 925, 929, 935, 937, 962, 968, 969, 1061, 1092
cas_name (*field attribute*), 1160
cas_number (*field attribute*), 1160
categorical_label (*field*), 895, 1092
category (*field*), 570
category (*NXDL attribute*), 69
CC-Apm-Definitions, 859
CC-Apm-Introduction, 859
CC-Apm-Paraprobe-Application-Definitions, 859
CC-Apm-Paraprobe-German-NFDI, 859
CC-Apm-Paraprobe-Introduction, 859

CC-Apm-Paraprobe-Status-Quo, 859
CC-Cgeometry-Definitions, 859
CC-Danalysis-Definitions, 859
CC-Micro-Definitions, 865
CC-Micro-Introduction, 865
CC-Opt-Spec-Definitions, 862
CC-Opt-Spec-DispersiveMaterial, 862
CC-Opt-Spec-Introduction, 862
CC-Sample-Definitions, 859
CC-SpmAppDef, 866
CC-SpmNewBC, 866
CC-Transport-Definitions, 865
CC-Transport-Introduction, 865
cdeform_field (*field*), 342
cell_dimensions (*field*), 248, 883, 925, 1035, 1054, 1061
center (*field*), 262, 795, 800, 921, 930, 933, 937, 969, 1008
center_energy (*field*), 380
center_errors (*field*), 795, 800
center_frequency (*field*), 877, 1076
center_type (*field*), 800
central_stop_diameter (*field*), 402
central_stop_material (*field*), 402
central_stop_thickness (*field*), 402
cg_alpha_complex (*base class*), *see* NXcg_alpha_complex, **240**
cg_cylinder (*base class*), *see* NXcg_cylinder, **242**
cg_ellipsoid (*base class*), *see* NXcg_ellipsoid, **243**
cg_face_list_data_structure (*base class*), *see* NXcg_face_list_data_structure, **244**
cg_grid (*base class*), *see* NXcg_grid, **247**
cg_half_edge_data_structure (*base class*), *see* NXcg_half_edge_data_structure, **249**
cg_hexahedron (*base class*), *see* NXcg_hexahedron, **251**
cg_parallelogram (*base class*), *see* NXcg_parallelogram, **254**
cg_point (*base class*), *see* NXcg_point, **255**
cg_polygon (*base class*), *see* NXcg_polygon, **256**
cg_polyhedron (*base class*), *see* NXcg_polyhedron, **258**
cg_polyline (*base class*), *see* NXcg_polyline, **259**
cg_primitive (*base class*), *see* NXcg_primitive, **261**
cg_roi (*base class*), *see* NXcg_roi, **264**
cg_tetrahedron (*base class*), *see* NXcg_tetrahedron, **265**
cg_triangle (*base class*), *see* NXcg_triangle, **266**
cg_unit_normal (*base class*), *see* NXcg_unit_normal, **267**
changer_position (*field*), 488, 1098, 1107
channelNAME (*field*), 1146
characteristics_type (*group attribute*), 753
charge (*field*), 221
charge_state (*field*), 202, 221, 571, 572, 934
charge_state_whitelist (*field*), 912
check_sum (*field attribute*), 321
checksum (*field*), 444, 553, 563–569, 631, 645, 649, 658, 878, 879, 882, 890, 891, 897, 903, 904, 910–912, 918–920, 948, 949, 960, 968, 975, 1007, 1050, 1060
chemical_composition (*base class*), *see* NXchemical_composition, **268**
chemical_formula (*field*), 286, 394, 487, 494, 586, 709, 756, 792, 983, 998, 1139
chemical_symbol (*field*), 555
chi (*field*), 747, 837, 1181
chirp_GDD (*field*), 229
chirp_type (*field*), 229
choice, **144**
choice (*NXDL element*), **146**
choiceType (*NXDL data type*), **165**
chromatic (*field*), 449
CIF, 29
cif (*field*), 1041
circuit (*base class*), *see* NXcircuit, **269**
cite (*base class*), *see* NXcite, **271**
class definitions, **175**, *see* base class, *see* application definition, *see* contributed definition
class path, **22**
classification (*field*), 876
clear_aperture (*field*), 532, 978
clock_speed (*field*), 296
closest_corner (*field*), 961
cluster_composition (*field*), 907
cluster_id (*field*), 907
cluster_selection_epsilon (*field*), 892
cluster_statistics (*field*), 907
cnxvalidate (*utility*), 1206
coating_material (*field*), 395, 404, 407, 435, 450, 533, 980
coating_roughness (*field*), 395, 405, 407, 435
coating_thickness (*field*), 450, 533, 980
coating_type (*field*), 450, 533, 980
coefficient (*field*), 1120–1122, 1150
coefficientN (*field*), 1137
collection (*base class*), *see* NXcollection, **272**
collection_description (*field*), 383, 518, 584, 786
collection_identifier (*field*), 383, 518, 584, 786, 1093, 1102
collection_time (*field*), 384, 519, 584
collection_title (*field*), 1093, 1102
collectioncolumn (*base class*), *see* NXcollectioncolumn, **273**
collimator (*base class*), *see* NXcollimator, **275**
colloquial_name (*field*), 998
color_map (*field*), 1039
color_model (*field*), 1032
color_palette (*field*), 1039
command1 (*field*), 1095, 1104

command_line_call (*field*), 297
comment (*field attribute*), 373, 374, 384, 519
comment (*field*), 207, 218, 568, 1021
common_beam_depolarizer (*field*), 1162
community, 1197
comp_current (*field*), 401
comp_turns (*field*), 400
complex (*field*), 414, 415, 417–420, 646–649
component (*base class*), *see* NXcomponent, **276**
component (*field*), 490
component_index (*field*), 403
components (*field*), 269
composition (*field*), 269, 464, 555
composition_errors (*field*), 269, 555
concentration (*field*), 490
configuration (*field attribute*), 384, 519
connections (*field*), 270
constitution, 79, 1197
construction_date (*field*), 390
construction_year (*field*), 754
container (*base class*), *see* NXcontainer, **982**
context (*field*), 414, 510, 645, 649
continuousN (*field*), 1147
contribute, 79
contributed definition, **858**, 1197
control_action (*field*), 462
control_points (*field*), 918
controller_label (*field*), 1142
controller_record (*field*), 465
convention (*field*), 993, 997, 999–1001
conversion, 1205
cooler_or_heater (*field*), 754
coordinate (*field*), 248, 883, 1064
coordinate systems, 32
    CIF, 32
    IUCr, 27
    McStas, 27, 32
    NeXus, **26**
    NeXus polar coordinate, 32
    spherical polar, 33
    transformations, 27
coordinate_system (*base class*), *see* NXcoordinate_system, **278**
coordinates, 312
copyright, 1216
core_sample_indices (*field*), 895
correction_peak (*field*), 566
corrector_cs (*base class*), *see* NXcorrector_cs, **281**
count (*field*), 934, 1084
count_time (*field*), 324, 440, 733, 789, 1172, 1180
countrate_correction_applied (*field*), 326, 735
countrate_correction_lookup_table (*field*), 326, 735
coupled (*field*), 437
coupling_material (*field*), 437, 1097, 1106
crate (*field*), 323
creator (*file attribute*), 483
creator_version (*file attribute*), 483
critical_energy (*field*), 234
crosstalk_compensation (*field*), 877
crosstalk_factor (*field*), 877
cryo_bottom_temperature (*field*), 874, 1118, 1153, 1156
cryo_shield_temperature (*field*), 874, 1118, 1153, 1156
crystal (*base class*), *see* NXcrystal, **285**
crystal_symmetry (*field*), 485
crystal_symmetry_point_group (*field*), 1043, 1046
crystal_system (*field*), 527
crystallographic_calibration (*field*), 218, 568
cs_computer (*base class*), *see* NXcs_computer, **291**
cs_filter_boolean_mask (*base class*), *see* NXcs_filter_boolean_mask, **292**
cs_memory (*base class*), *see* NXcs_memory, **293**
cs_prng (*base class*), *see* NXcs_prng, **294**
cs_processor (*base class*), *see* NXcs_processor, **295**
cs_profiling (*base class*), *see* NXcs_profiling, **296**
cs_profiling_event (*base class*), *see* NXcs_profiling_event, **298**
cs_storage (*base class*), *see* NXcs_storage, **300**
csg (*base class*), *see* NXcsg, **984**
cue_index (*field*), 390, 429
cue_timestamp_zero (*field*), 389, 429
cumulated (*field*), 952
cumulated_normalized (*field*), 952
current (*field*), 319, 347, 411, 505, 706, 1012, 1120, 1152, 1155
current_gain (*field*), 1120, 1155
current_offset (*field*), 1152, 1155
current_set_feature_to_cluster (*field*), 907
current_to_next_link (*field*), 906
current_to_next_link_type (*field*), 906
current_working_directory (*field*), 297, 882
curvature (*field*), 534
curvature_horizontal (*field*), 288
curvature_radius_FACE (*field*), 450
curvature_radiusN (*field*), 1132, 1136
curvature_vertical (*field*), 288
cut_angle (*field*), 287
cxi_ptycho (*application definition*), *see* NXcxi_ptycho, **985**
cylinders (*field*), 301
cylindrical (*field*), 534
cylindrical_geometry (*base class*), *see* NXcylindrical_geometry, **300**
cylindrical_orientation_angle (*field*), 289

D

d (*field*), 471
D_piezoelectric_constant (*field*), 1139
d_spacing (*field*), 288
D_t (*field*), 1141
data
    multi-dimensional, 54
data (*base class*), *see* NXdata, **301**
DATA (*field*), 316, 1123, 1147, 1150
data (*field*), 321, 362, 408, 440, 444, 588, 592, 597, 659, 683, 687–689, 691, 712, 726, 730, 733, 767, 769, 770, 772, 776, 777, 779, 781, 783, 787, 807, 808, 810, 811, 814, 816, 817, 819, 820, 822, 825, 828, 830, 832, 834, 840, 987, 988, 1032, 1033, 1041, 1098, 1105, 1107, 1177
data analysis software, 1206
data field, *see* field
data group, *see* group
data item, *see* field
data object, *see* field
data set, *see* field
data type, **170**
data_correction (*field*), 623
data_errors (*field*), 322
data_file (*field*), 1178
data_identifier (*field*), 624
data_offset (*field*), 731
data_origin (*field*), 336, 737
data_scaling_factor (*field*), 730
data_size (*field*), 336, 737
data_stride (*field*), 737
data_type (*field*), 624
data_x_time_of_flight (*field*), 1105
data_x_y (*field*), 1096, 1105
data_y_time_of_flight (*field*), 1105
dataset, *see* field
datatype_N (*field*), 233
date (*field*), 443, 466, 592, 597, 615, 788, 794, 827, 832, 1021, 1095, 1104
date and time, 48
DAVE (*data analysis software*), 1207
DAWN (*data analysis software*), 1207
DAXIS (*field*), 798
dc_offset_valueN (*field*), 1015
dead_time (*field*), 323, 733, 789
decorator_multiplicity (*field*), 935
default, **50**
default (*file attribute*), 236, 383, 445, 483, 518
default (*group attribute*), 591, 595, 602, 1086
default attribute value, **50**, 383, 483, 518
default plot, *see* plotting
default_color_map (*field*), 1039
default_slice (*file attribute*), 314
definition (*field*), 383, 519, 551, 584, 588, 591, 596, 603, 620, 622, 630, 683, 685, 686, 688, 691, 693, 722, 730, 744, 764, 767, 769, 772, 775, 779, 780, 782, 786, 806, 810, 813, 816, 819, 821, 824, 827, 829, 832, 833, 837, 838, 840–843, 857, 872, 878, 882, 889, 894, 897, 899, 901, 906, 909, 925, 944–947, 951, 953, 955, 960, 961, 968, 975, 986, 998, 1006, 1012, 1035, 1048, 1060, 1086, 1093, 1102, 1116, 1151, 1154, 1161, 1167, 1177, 1178
definition (*NXDL data type*), **155**
definition (*NXDL element*), 69, **146**
definition_local (*field*), 384, 519
definitionType (*NXDL data type*), **155**
definitionTypeAttr (*NXDL data type*), **157**
deflection_angle (*field*), 404, 979
deflector (*base class*), *see* NXdeflector, **319**
defocus (*field*), 377
deformed_grain_id (*field*), 1064
defragment (*field*), 1055
defragment_x (*field*), 1055
degree_char (*field*), 1039
delay (*field*), 340, 704, 714
delay_difference (*field*), 1169
delocalization (*base class*), *see* NXdelocalization, **990**
delta_time (*field*), 197, 204, 1007
demodulated_amplitude (*field*), 1014
demodulated_frequency (*field*), 1014
demodulated_phase (*field*), 1015
demodulated_signal (*field*), 1014
demodulator_channels (*field*), 1015
density (*field*), 288, 393, 489, 495, 634, 983, 1139
depends on (*field attribute*), **28**
depends_on (*field attribute*), 216, 222, 229, 337, 524, 563, 723–726, 738, 751, 845–847, 852, 853, 1032
depends_on (*field*), 200, 224, 229, 232, 235, 240, 260, 261, 275, 277, 281, 289, 293, 328, 337, 340, 357, 358, 387, 392, 395, 401, 402, 405, 407, 424, 435, 437, 440, 442, 463–465, 479, 500, 502, 507, 530, 535, 637, 639, 658, 722–724, 731, 733, 746, 789, 792, 793, 844–846, 852, 1032, 1080
depolarization (*field*), 758
deprecated, 201, 232, 235, 276, 289, 316–318, 321, 322, 328, 340, 384, 385, 392, 395, 403, 405, 407, 424, 429, 435, 438–442, 482, 491, 492, 500, 507, 531, 613, 739, 740, 1094, 1103, 1167
depth (*field*), 404
description (*field attribute*), 400, 455, 469–476, 1004, 1017, 1088
description (*field*), 198, 200, 214, 231, 236, 263, 272, 277, 282, 298, 323, 342, 347, 349, 365, 387,

393, 398, 399, 403, 406, 411, 428, 430, 434, 444, 461, 465, 479, 490, 495, 554, 555, 570, 585, 615, 634, 700, 733, 756, 788, 850, 852, 972, 974, 983, 995, 996, 1004, 1017, 1035, 1048, 1060, 1069, 1080, 1081, 1090, 1094–1097, 1103–1106, 1111, 1114, 1172
description (*file attribute*), 469
descriptor (*field*), 361, 659
design (*field*), 373, 468, 644, 645, 656
design principles, 4
det_module (*field*), 470
details (*field*), 614
detection_gas_path (*field*), 323
detector (*base class*), *see* NXdetector, **320**
detector_channel (*base class*), *see* NXdetector_channel, **332**
detector_channel_type (*field*), 749
detector_faces (*field*), 448
detector_group (*base class*), *see* NXdetector_group, **334**
detector_identifier (*field*), 414, 509, 645, 649
detector_module (*base class*), *see* NXdetector_module, **336**
detector_number (*field*), 301, 321, 814, 816
detector_readout_time (*field*), 327, 736
detector_type (*field*), 348, 702, 749
detector_voltage (*field*), 348
device_path (*field*), 755
diameter (*field*), 325, 381, 463, 532, 724, 841, 1049, 1071
dictionary of terms, 14
dielectric_function (*field*), 997, 999, 1000
diffraction_order (*field*), 404
diffractogram (*field*), 798
diffractogram_errors (*field*), 798
dim (*NXDL element*), 70
dimension, 20, 54
    dimension scales, 54, 57, 58
    fastest varying, 45, 57, 337
    slowest varying, 45, 337
    storage order, 45
dimension scale, 25, 52, 53
dimensionality (*field*), 250, 261, 526, 883, 925, 929, 935–937, 956, 957, 962, 968, 969, 1011, 1021, 1048, 1061
dimensions (*NXDL element*), 70, **147**
dimensionsType (*NXDL data type*), **157**
direction, *see* vector (field attribute)
direction (*field attribute*), 488, 792
direction (*field*), 502
directtof (*application definition*), *see* NXdirecttof, **620**
disk_chopper (*base class*), *see* NXdisk_chopper, **338**
dislocation_density (*field*), 1064
disorientation_angle (*field*), 486
disorientation_axis (*field*), 486
disorientation_quaternion (*field*), 486
disorientation_threshold (*field*), 1021
dispersion (*base class*), *see* NXdispersion, **992**
dispersion (*field*), 344, 653, 1070
dispersion_function (*base class*), *see* NXdispersion_function, **993**
dispersion_repeated_parameter (*base class*), *see* NXdispersion_repeated_parameter, **995**
dispersion_single_parameter (*base class*), *see* NXdispersion_single_parameter, **995**
dispersion_table (*base class*), *see* NXdispersion_table, **996**
dispersion_type (*field*), 998, 1070
dispersive_material (*application definition*), *see* NXdispersive_material, **997**
DispersiveMaterial-BC, 189
distance (*field*), 224, 226, 322, 340, 392, 437, 439, 491, 503, 611, 685, 689, 698, 699, 733, 769, 773, 776, 781, 788, 814, 816, 817, 819, 820, 822, 825, 834, 835, 891, 899, 911, 919, 920, 948, 949, 951, 952, 960, 973, 987, 1096–1098, 1105–1107, 1172
distance_derived (*field*), 733
distance_to_detector (*field*), 231
distances (*field*), 525
distancing_model (*field*), 919
distortion (*base class*), *see* NXdistortion, **341**
distribution (*field attribute*), 316
divergence_x (*field*), 275
divergence_x_minus (*field*), 235
divergence_x_plus (*field*), 235
divergence_y (*field*), 275
divergence_y_minus (*field*), 235
divergence_y_plus (*field*), 235
dld_wire_names (*field*), 562
doc (*NXDL element*), 70, **147**
docType (*NXDL data type*), **159**
documentation editor, **1215**
doi (*field*), 272, 553, 631, 998
dose_management (*field*), 378
dose_rate (*field*), 378
download location, *see* NAPI installation
dQl (*field*), 610
dQw (*field*), 609
drift_correction_status (*field*), 1136
drift_energy (*field*), 380, 701
driftN (*field*), 1136
dspacing (*field*), 361
duration (*field*), 384, 429, 519, 584, 816, 819, 1094, 1102, 1103
duty_cycle (*field*), 404
dwell_time (*field*), 497, 655, 656
dynamic_focus_correction (*field*), 378

dynamic_phi_list (*field*), 1174
dynamic_q_list (*field*), 1173
dynamic_refocusing (*field*), 378
dynamic_roi_map (*field*), 1173

E

ebeam_column (*base class*), *see* NXebeam_column, **342**
ebsd (*field*), 1041
ebsd_extensions (*field*), 1041
edge_contact (*field*), 937
edge_distance (*field*), 948
edge_length (*field*), 257, 259, 265, 266, 879, 930, 935
edge_method (*field*), 912
edge_threshold (*field*), 913
edges (*field*), 246
edges_are_unique (*field*), 246
ef (*field*), 810
efficiency (*field*), 217, 329, 440, 464, 568, 788
ei (*field*), 810
elapsed_time (*field*), 299, 552
electric_field (*field*), 488, 586
electromagnetic_lens (*base class*), *see* NXelectromagnetic_lens, **346**
electron_detector (*base class*), *see* NXelectron_detector, **348**
electronanalyzer (*base class*), *see* NXelectronanalyzer, **349**
electrostatic_kicker (*base class*), *see* NXelectrostatic_kicker, **1004**
ellipsometer_type (*field*), 623
ellipsometry (*application definition*), *see* NXellipsometry, **621**
ellipsometry_experiment_type (*field*), 622
em (*application definition*), *see* NXem, **627**
em_calorimetry (*application definition*), *see* NXem_calorimetry, **1005**
em_ebsd (*base class*), *see* NXem_ebsd, **353**
em_eds (*base class*), *see* NXem_eds, **364**
em_eels (*base class*), *see* NXem_eels, **366**
em_event_data (*base class*), *see* NXem_event_data, **367**
em_img (*base class*), *see* NXem_img, **370**
em_instrument (*base class*), *see* NXem_instrument, **371**
em_interaction_volume (*base class*), *see* NXem_interaction_volume, **375**
em_measurement (*base class*), *see* NXem_measurement, **376**
em_optical_system (*base class*), *see* NXem_optical_system, **377**
em_simulation (*base class*), *see* NXem_simulation, **379**
email (*field*), 529, 632, 1087
emission_current (*field*), 343, 506, 652
emittance_x (*field*), 505
emittance_y (*field*), 505
emitter_material (*field*), 343
emitter_type (*field*), 343, 410, 641, 643
en (*field*), 783, 810
end (*field*), 1180
end_settling_time (*field*), 1129
end_time (*field*), 198, 204, 297, 298, 369, 384, 439, 519, 552, 560, 584, 630, 645, 693, 709, 730, 745, 767, 769, 772, 779, 786, 806, 821, 824, 882, 966, 970, 976, 986, 1006, 1035, 1060, 1086, 1094, 1102, 1167
end_time_estimated (*field*), 730
endnote (*field*), 272
energies (*field*), 589
energy (*field*), 365, 392, 424, 441, 505, 588, 591, 596, 620, 683, 685, 702, 707, 712, 726, 781, 807, 808, 830, 832, 853, 986, 987, 997, 1028
energy_dispersion (*field*), 442
energy_error (*field*), 442
energy_errors (*field*), 442
energy_identifier (*field*), 993, 999–1001
energy_indices (*group attribute*), 712, 726, 853
energy_interval (*field*), 380
energy_max (*field*), 993
energy_min (*field*), 993
energy_range (*field*), 365, 660
energy_scan_mode (*field*), 381, 701, 846
energy_transfer (*field*), 226
energy_unit (*field*), 993, 999–1001
energydispersion (*base class*), *see* NXenergydispersion, **380**
entering (*field*), 470
enthalpy (*field*), 1052
entrance_slit_setting (*field*), 588
entrance_slit_shape (*field*), 588
entrance_slit_size (*field*), 589
entry (*base class*), *see* NXentry, **382**
entry_identifier (*field*), 383, 519, 584, 1094, 1102, 1167
entry_identifier_uuid (*field*), 383, 1167
enumeration, 49
enumeration (*NXDL element*), **147**
enumerationType (*NXDL data type*), **159**
environment (*base class*), *see* NXenvironment, **387**
EPICS
    instrument examples, **125**
epoch_start (*field attribute*), 357
eps (*field*), 880, 891, 894, 1040
epsilon (*field*), 886
equipment_component (*field attribute*), 524
error (*field attribute*), 455
error (*field*), 781
errors (*field*), 317
estimated (*field attribute*), 198
euler (*field*), 1044
euler_angle (*field*), 1039

euler_angle_convention (*field*), 556, 634
eulerian cradle, 28, 537, 546, 836
evaporation_control (*field*), 210, 561
evaporation_field (*field*), 217, 568
evaporation_id (*field*), 565
evaporation_id_offset (*field*), 565
even_layer_density (*field*), 435
even_layer_material (*field*), 435
event_data (*base class*), *see* NXevent_data, **388**
event_id (*field*), 389
event_index (*field*), 389, 1096
event_pixel_id (*field*), 1096
event_time_of_flight (*field*), 1096
event_time_offset (*field*), 389
event_time_zero (*field*), 389
example (*field*), 1041
examples
    NeXus file, 5
    NeXus file; minimal, 6
experiment_alias (*field*), 551, 630
experiment_description (*field*), 383, 518, 551, 584, 630, 745, 786, 1086, 1162
experiment_identifier (*field*), 383, 518, 584, 786, 1094, 1102, 1162
experiment_sub_type (*field*), 745
experiment_technique (*field*), 872, 1116, 1151, 1154
experiment_type (*field*), 622, 745, 765
experiments (*field*), 469
exposure_time (*field*), 282
expression (*field attribute*), 455
extends (*NXDL attribute*), 69
extent (*field*), 227, 248, 698, 699, 748, 883, 925, 987, 1035, 1049, 1054, 1061, 1172
external_DAC (*field*), 491
external_field_brief (*field*), 499
external_material (*field*), 407, 434
extinction_ratio (*field*), 1073
extraction_voltage (*field*), 343, 652
extractor_current (*field*), 274
extractor_voltage (*field*), 273
extrema (*field*), 1044

F

fabrication (*base class*), *see* NXfabrication, **390**
fabrication (*field*), 402
face_area (*field*), 253, 259, 265
face_half_edge (*field*), 250
face_index_offset (*field*), 927, 929
face_normal (*field*), 936
face_normal_orientation (*field*), 936
face_normals (*field*), 898, 934
faces (*field*), 246, 448, 927, 929, 934, 936, 957
faces_are_unique (*field*), 246
facet_normals (*field*), 920
facility_user_id (*field*), 529, 585, 1098, 1107
FAQ, **78**
farthest_corner (*field*), 961
fast_axes (*field*), 350, 700
fast_pixel_direction (*field*), 337, 738
fax_number (*field*), 529
FDL, 1216
feature_distance (*field*), 949
feature_type (*field*), 903, 904, 932
feature_type_dict_keyword (*field*), 932
feature_type_dict_value (*field*), 932
features (*field*), 383
feed_forward_model (*field*), 461
feedback_on (*field*), 1141
fermi_chopper (*base class*), *see* NXfermi_chopper, **391**
fermi_level (*field*), 1133
fft_accuracy (*field*), 1040
field, 4, **19**
    HDF, 59
field (*NXDL element*), 70, **150**
field attribute, 4, 16, **20**
field_of_view (*field*), 216, 377, 567
FIELDNAME_errors (*field*), 317, 445
FIELDNAME_mask (*field*), 445
FIELDNAME_offset (*field*), 317
FIELDNAME_scaling_factor (*field*), 317
FIELDNAME_set (*field*), 445
FIELDNAME_weights (*field*), 445
fieldType (*NXDL data type*), **161**
figure_of_merit (*field*), 517
figure_of_meritMETRIC (*field*), 396, 847
figure_size (*field*), 1039
filament_current (*field*), 343, 506, 652
file
    read and write, 14
    validate, 1206
file attribute, **21**
file format, 58
    HDF, 59
file_name (*field*), 443, 553, 563–569, 631, 645, 649, 658, 787, 878, 879, 882, 890, 891, 897, 903, 904, 910–912, 917–920, 948, 949, 960, 968, 975, 1007, 1050, 1060
file_name (*file attribute*), 482
file_time (*file attribute*), 482
file_update_time (*file attribute*), 482
filenames (*field*), 686, 783
filter (*base class*), *see* NXfilter, **393**
filter_mechanism (*field*), 752
filter_type (*field*), 752
final_beam_divergence (*field*), 228
final_energy (*field*), 226
final_polarization (*field*), 227
final_polarization_stokes (*field*), 228

final_wavelength (*field*), 227
final_wavelength_spread (*field*), 228
final_z (*field*), 1129, 1142
find the default plottable data, **50**, 314, 383, 483, 518
first_good (*field attribute*), 316
first_point_for_fit (*field attribute*), 1171
first_settling_time (*field*), 1129
fit (*base class*), *see* NXfit, **396**
fit (*field*), 798
fit_errors (*field*), 799
fit_formula_description (*field*), 237
fit_function (*base class*), *see* NXfit_function, **399**
fit_range (*field*), 794
fit_sum (*field*), 397, 847
fit_type (*field*), 794
fixed_energy (*field*), 781
fixed_revolutions (*field*), 624
flags (*field*), 470
flatfield (*field*), 325, 333, 734
flatfield_applied (*field*), 325, 333, 734
flatfield_error (*field*), 734
flatfield_errors (*field*), 325, 334, 734
flexible name, 143
flight_path (*field*), 206, 210, 217, 558, 568
flip_current (*field*), 400
flip_sign (*field*), 1117
flip_turns (*field*), 400
flipper (*base class*), *see* NXflipper, **400**
floating-point numbers, **48**
fluence (*field*), 228
fluo (*application definition*), *see* NXfluo, **682**
flux (*field*), 228, 411, 505, 655, 739
flux (*group attribute*), 738
flux_integrated (*field*), 740
flyback_time (*field*), 497
focal_length (*field*), 378, 450
focal_size (*field*), 240
focus_parameters (*field*), 401
focus_type (*field*), 534
folder, *see* group
font_size (*field*), 1039
form_factor (*field*), 796
form_factor_errors (*field*), 796
format unification, 13
formula (*field*), 993, 999–1001
formula_description (*field*), 398, 399, 850, 852
forward_speedN (*field*), 1146
four-circle diffractometer, 28, 38, 537, 546, 836
frame_average (*field*), 1167
frame_start_number (*field*), 325, 834
frame_sum (*field*), 1167
frame_time (*field*), 327, 736, 1173
frequency (*field attribute*), 321
frequency (*field*), 275, 505, 1081, 1095, 1104
frequency_bandwidth (*field*), 1076, 1135
frequency_cutoff (*field*), 1135
frequency_harmonic (*field*), 1134
frequency_modulation_bandwidth (*field*), 1014
frequency_shift (*field*), 1077, 1134
frequency_step (*field*), 1076
fresnel_zone_plate (*base class*), *see* NXfresnel_zone_plate, **401**
frog_delays (*field*), 228
frog_frequencies (*field*), 228
frog_trace (*field*), 228
function_type (*field*), 399, 848, 850, 852
fwhm (*field*), 795
fwhm_errors (*field*), 795
fwhm_left (*field*), 796
fwhm_left_errors (*field*), 796
fwhm_right (*field*), 796
fwhm_right_errors (*field*), 796

G

g2 (*field*), 1168
g2_derr (*field*), 1168
g2_err_from_two_time_corr_func (*field*), 1171
g2_err_from_two_time_corr_func_partials (*field*), 1171
g2_from_two_time_corr_func (*field*), 1170
g2_from_two_time_corr_func_partials (*field*), 1171
G2_unnormalized (*field*), 1168
G_voltage_constant (*field*), 1139
gain (*field*), 270, 877, 1164
gain_setting (*field*), 327, 736
gamma (*field*), 527, 659
gap (*field*), 423
gas (*field*), 535
gas_pressure (*field*), 323, 506, 535
GDA (*data acquisition software*), 1207
generating_help_mode (*field*), 1040
geometry, 27, 32, 33
geometry (*base class*), *see* NXgeometry, **403**
geometry (*field*), 903, 904, 985
git, 1199
glass_transition_temperature (*field*), 1140
goniometer_x (*field*), 1180
goniometer_y (*field*), 1180
goniometer_z (*field*), 1180
goodness_of_fit (*field*), 794
gradient_guide_magnitude (*field*), 931
gradient_guide_projection (*field*), 931
grain_diameter (*field*), 553
grain_diameter_errors (*field*), 553
grating (*base class*), *see* NXgrating, **404**
grid_resolution (*field*), 910

group, 4, **19**
    HDF, 59
group (*NXDL element*), 70, **150**
group attribute, 4, **20**
group_index (*field*), 335, 732
group_names (*field*), 335, 732
group_parent (*field*), 335, 733
group_type (*field*), 335
groupType (*NXDL data type*), **165**
guide (*base class*), *see* NXguide, **406**
guide_current (*field*), 401
guide_turns (*field*), 400
Gumtree (*data analysis software*), 1207

H

h (*field*), 469, 799
h5py
    code examples, **88**
h5py_version (*file attribute*), 483
h5wasm
    tools, 1208
h5web
    tools, 1208
hagb_enthalpy (*field*), 1051
hagb_pre_factor (*field*), 1051
half_angle_interval (*field*), 371
half_axes_radii (*field*), 969
half_edge_incident_face (*field*), 250
half_edge_next (*field*), 251
half_edge_prev (*field*), 251
half_edge_twin (*field*), 250
half_edge_vertex_origin (*field*), 250
halfwidth (*field*), 1046
halo_region (*field*), 1064
handedness (*field*), 1060
harmonic (*field*), 424
harmonic_orderN (*field*), 1015
has_cell_edge_detection (*field*), 960
has_cell_geometry (*field*), 960
has_cell_neighbors (*field*), 960
has_cell_volume (*field*), 960
has_closure (*field*), 954
has_current_to_next_links (*field*), 902
has_exterior_facets (*field*), 954
has_interior_tetrahedra (*field*), 954
has_next_to_current_links (*field*), 902
has_object (*field*), 914
has_object_edge_contact (*field*), 915
has_object_geometry (*field*), 914
has_object_ions (*field*), 915
has_object_obb (*field*), 914
has_object_properties (*field*), 914
has_object_proxigram (*field*), 916
has_object_proxigram_edge_contact (*field*), 916
has_proxy (*field*), 915
has_proxy_edge_contact (*field*), 916
has_proxy_geometry (*field*), 915
has_proxy_ions (*field*), 916
has_proxy_obb (*field*), 916
has_proxy_properties (*field*), 916
has_scalar_fields (*field*), 910
has_triangle_soup (*field*), 913
HDF
    file format, 59
    tools, 1208
HDF4, 1212
HDF5, 1212
HDF5_Version (*file attribute*), 483
HDF_version (*file attribute*), 483
HDFexplorer
    tools, 1208
HDFview
    tools, 1208
head_temperature (*field*), 874, 1117, 1152, 1156
heat_treatment_quenching_rate (*field*), 554
heat_treatment_quenching_rate_errors (*field*), 554
heat_treatment_temperature (*field*), 554
heat_treatment_temperature_errors (*field*), 554
heat_treatment_time (*field*), 554
heater_current (*field*), 656
heater_power (*field*), 657
heater_voltage (*field*), 656
height (*field*), 242, 253, 262, 392, 530, 795, 921, 969
height_errors (*field*), 795
hierarchy, 4, 18, 19, 34, 36, 66, 67, 74, 1205
high_passN (*field*), 1015
high_throughput_method (*field*), 891, 892
high_trip_value (*field*), 499
history (*base class*), *see* NXhistory, **409**
hit_multiplicity (*field*), 564
hit_positions (*field*), 563
hit_quality (*field*), 564
hit_quality_type (*field*), 564
hit_test (*field*), 1040
holder (*field*), 1098, 1107
hp_filter_orderN (*field*), 1015
hv_gainN (*field*), 1136

I

I (*field*), 592, 597, 607
I_axes (*group attribute*), 604
I_errors (*field*), 593, 597
I_t (*field*), 1141
ibeam_column (*base class*), *see* NXibeam_column, **409**
id (*field*), 221, 470
identifier (*field*), 696–698, 700–702, 706, 709, 754, 1098, 1107, 1162

identifier_analysis (*field*), 878, 882, 966, 968, 972, 975, 1006
identifier_calibration_method (*field*), 236
identifier_calibration_reference (*field*), 236
identifier_canonical_smiles (*field*), 1160
identifier_cas (*field*), 1160
identifier_chemical (*field*), 221
identifier_collection (*field*), 1086
identifier_environment (*field*), 1117, 1128
identifier_event (*field*), 369
identifier_experiment (*field*), 630, 745, 1086, 1116
identifier_inchi_key (*field*), 1160
identifier_inchi_str (*field*), 1160
identifier_iupac_name (*field*), 1160
identifier_parent (*field*), 555, 633
identifier_piezo_material (*field*), 1139
identifier_pub_chem (*field*), 1160
identifier_sample (*field*), 369, 633, 645
identifier_simulation (*field*), 1048, 1060
identifier_smiles (*field*), 1160
identifierNAME (*field*), 409, 443, 445, 553, 555, 631
Idev (*field*), 608
IDF_Version (*file attribute*), 383, 518
IDL (*data analysis software*), 1207
IGOR Pro (*data analysis software*), 1207
imag (*field*), 414–416, 418–420, 645–649
image (*base class*), *see* NXimage, **412**
image_compression (*field*), 217, 568
image_key (*field*), 326, 822
images, 48
imaging_mode (*field*), 371, 657
import_wizard (*field*), 1041
incidence_vector (*field*), 209
incident_angle (*field*), 406, 434, 979
incident_beam_divergence (*field*), 227, 987
incident_beam_energy (*field*), 987
incident_beam_size (*field*), 740
incident_energy (*field*), 226, 698, 699, 791, 1172, 1177
incident_energy_spread (*field*), 226, 698, 699, 987, 1172
incident_energy_weights (*field*), 226
incident_polarisation_stokes (*field*), 740
incident_polarization (*field*), 227, 698, 699, 748
incident_polarization_stokes (*field*), 227, 740
incident_polarization_type (*field*), 1172
incident_wavelength (*field*), 226, 613, 738, 748, 792
incident_wavelength_spread (*field*), 227, 614, 739, 748
incident_wavelength_weight (*field*), 739
incident_wavelength_weights (*field*), 227, 739
increment (*field*), 949, 969, 1158
incubation_time (*field*), 1050, 1051
independent_controllers (*field*), 1087
independent_scan_axes (*field*), 1143
index (*group attribute*), 584
index (*NXDL attribute*), 70
index_of_refraction (*field*), 450, 1071, 1074
index_of_refraction_coating (*field*), 451, 533, 981, 1075
index_of_refraction_substrate (*field*), 532, 980
index_offset (*field*), 262, 883, 894, 921, 925, 926, 929, 935, 957, 962, 968, 969, 1022–1025, 1027, 1061, 1092
index_offset_crystal (*field*), 1063
index_offset_edge (*field*), 245, 250, 936
index_offset_face (*field*), 246, 250, 936
index_offset_phase (*field*), 1064
index_offset_vertex (*field*), 245, 250, 936
indexing_rate (*field*), 360, 658
indication_time (*field*), 1082
indices (*field*), 222, 262, 898, 910, 918–920
indices_cluster (*field*), 1092
indices_crystal (*field*), 1023, 1024, 1026, 1027, 1036, 1063
indices_edge (*field*), 246
indices_face (*field*), 246, 936
indices_feature (*field*), 895, 903, 904, 932, 933
indices_feature_xdmf (*field*), 934
indices_group (*field*), 418–420, 512, 513, 648, 649
indices_image (*field*), 418–420, 648, 649
indices_interface (*field*), 1024, 1026, 1027
indices_offset_face (*field*), 957, 962
indices_offset_vertex (*field*), 957, 962
indices_patch (*field*), 898
indices_pattern (*field*), 1007, 1008
indices_phase (*field*), 1023, 1024, 1027, 1036, 1064
indices_point (*field*), 899
indices_polyline (*field*), 1026
indices_quadruple_junction (*field*), 1027
indices_spectrum (*field*), 512, 513, 651, 652
indices_triangle (*field*), 899
indices_triangle_cluster (*field*), 932
indices_triple_junction (*field*), 1024, 1025, 1027
indices_vertex (*field*), 246
indices_voxel (*field*), 884
indirecttof (*application definition*), *see* NXindirecttof, **684**
infection_direction (*field*), 1065
ingestion, 1205
initial (*field attribute*), 455
initial (*field*), 1055
initial_guess (*field*), 879
initial_radius (*field*), 556
initialization (*field*), 916
inner_plot_spacing (*field*), 1039
input (*field attribute*), 234
input (*field*), 324

input_dependent (*field*), 397, 847
input_impedance (*field*), 270
input_independent (*field*), 397, 847
input_path (*field attribute*), 237, 238
inputs (*field*), 277
insertion_device (*base class*), *see* NXinsertion_device, **423**
inside_poly (*field*), 1040
inspection, 1205
installation, *see* NAPI installation
instrument (*base class*), *see* NXinstrument, **425**
instrument definitions, 9
int_prf (*field*), 475
int_prf_errors (*field*), 475
int_prf_var (*field*), 475
int_sum (*field*), 475
int_sum_errors (*field*), 475
int_sum_var (*field*), 475
integers, **48**
integral (*field*), 439, 691, 770, 773, 825, 835
integral_counts (*field*), 817
integration_time (*field*), 1180
integration_type (*field*), 794
intended_use (*field*), 752
intensity (*field*), 364, 414–416, 418, 419, 459, 510–513, 568, 569, 645–652, 659, 660, 848, 850, 1008, 1045, 1047, 1180, 1181
interior_angle (*field*), 257, 267, 930, 935
interior_atmosphere (*field*), 404, 406, 434
interpolation (*field*), 1032
interpretation (*field attribute*), 798, 799, 987
interpretation (*group attribute*), 592, 597
intersection_detection_method (*field*), 901
intersection_volume (*field*), 906
introduction, 3
ion_energy_profile (*field*), 411
ion_id (*field*), 934
ion_multiplicity (*field*), 935
ion_query_nuclide_source (*field*), 947
ion_query_nuclide_target (*field*), 948
ion_query_nuclide_vector (*field*), 891
ion_query_type_source (*field*), 947
ion_query_type_target (*field*), 948
ion_type_filter (*field*), 890
ion_volume (*field*), 217, 568
iontypes (*field*), 571, 944
iontypes_randomized (*field*), 951
iqproc (*application definition*), *see* NXiqproc, **685**
is_amorphous (*field*), 556
is_antipodal (*field*), 485
is_axis_aligned (*field*), 253, 255, 926
is_box (*field*), 253
is_center_of_mass (*field*), 262
is_centrosymmetric (*field*), 527
is_chiral (*field*), 527
is_closed (*field*), 262
is_core (*field*), 895
is_cylindrical (*field*), 288
is_deformed (*field*), 1064
is_geodesic_mesh (*field*), 263
is_mesh (*field*), 263
is_noise (*field*), 895
is_polycrystalline (*field*), 555
is_recrystallized (*field*), 1064
is_rectangle (*field*), 255
is_regularized (*field*), 241
is_simulation (*field*), 553, 555, 632, 882, 1006, 1048
is_specific (*field*), 1069
is_surface_mesh (*field*), 263
is_triangle_mesh (*field*), 263
is_watertight (*field*), 957
ISAW (*data analysis software*), 1207
ISO8601 (*data type*), **170**
isocontour (*base class*), *see* NXisocontour, **1010**
isovalue (*field*), 929, 1011
issue reporting, 1199
iteration (*field*), 1021, 1062, 1063
iteration_indices (*group attribute*), 1062
iupac_line_candidates (*field*), 365, 660
iupac_line_name (*field*), 365
iv_temp (*application definition*), *see* NXiv_temp, **1011**

J

jones_quality_factor (*field*), 758
jupyterlab-h5web
    tools, 1208

K

k (*field*), 424, 469, 799
k_alpha_one (*field*), 1179
k_alpha_two (*field*), 1179
K_d (*field*), 461
K_electromechanical_constant (*field*), 1139
K_ff (*field*), 461
K_i (*field*), 461
K_p (*field*), 461
k_parallel (*field*), 703, 713
k_perpendicular (*field*), 703, 713
kappa (*field*), 838
kbeta (*field*), 1179
kernel_halfwidth (*field*), 1043
kernel_mu (*field*), 926
kernel_name (*field*), 1044
kernel_sigma (*field*), 926
kernel_size (*field*), 910, 926
kernel_type (*field*), 926
kernel_variance (*field*), 910
kernel_width (*field*), 954

Kf_coefficient (*field*), 1077
kfactor (*field*), 217, 568
ki_over_kf_scaling (*field*), 781
kinetic_energy (*field*), 352, 380, 708
Klosowski, Przemysław, 12, 1212
Kn_coefficient (*field*), 1077
Ko_coefficient (*field*), 1078
Kp_coefficient (*field*), 1077
kth (*field*), 949, 1044
kx (*field*), 703, 712
ky (*field*), 703, 712
kz (*field*), 703, 713
Könnecke, Mark, 12, 1213

L

l (*field*), 470, 799
label (*field*), 396, 459, 570, 847, 850, 886
lagb_enthalpy (*field*), 1051
lagb_pre_factor (*field*), 1051
lambda (*field*), 616
LAMP (*data analysis software*), 1207
last_fill (*field*), 506
last_good (*field attribute*), 316
last_process (*field*), 236
lateral_focal_point_offset (*field*), 748
lateral_surface_area (*field*), 243
lattice (*field*), 799
lattice_type (*field*), 1068
laue_group (*field*), 527
lauetof (*application definition*), *see* NXlauetof, **688**
layer_structure (*field*), 756
layer_thickness (*field*), 405, 435
layout (*field*), 324
length (*field*), 253, 262, 424, 530, 611, 1023, 1025, 1026, 1070
lens_diameter (*field*), 449
lens_geometry (*field*), 534
lens_length (*field*), 534
lens_material (*field*), 535
lens_mode (*field*), 274, 588, 700
lens_thickness (*field*), 534
level (*field*), 708
lexicography, 14
LGPL, 1216
license, 1216
lifetime (*field*), 343
line_energy (*field*), 1026
linear_beam_sample_polarization (*field*), 748
link, 4, **21**, 54, 80, 175
    external file, 23
link (*NXDL element*), **150**
link target, 150
link target (*internal attribute*), **21**
link, target, attribute, 21
LINK_TO_FIELD (*field*), 1123
linkType (*NXDL data type*), **166**
local_name (*field attribute*), 323, 324
local_name (*field*), 323
location (*field*), 206, 372, 558, 640, 1025, 1027, 1044
lockin (*base class*), *see* NXlockin, **1013**
lockin_status (*field*), 1151
log (*base class*), *see* NXlog, **427**
logged_against (*field attribute*), 209
long_name (*field attribute*), 316, 317, 321, 322, 362, 364, 365, 414–421, 510–514, 561, 562, 569, 592, 593, 597, 598, 645–652, 659, 660, 884, 1008, 1036, 1053, 1054
low_passN (*field*), 1015
low_trip_value (*field*), 499
low-level file format, *see* file format
lower_cap_radii (*field*), 243
lower_cap_surface_area (*field*), 243
lower_cutoff_frequency (*field*), 877
lp (*field*), 475
lp_filter_orderN (*field*), 1015
LRMECS
    instrument examples, **122**

M

m_value (*field*), 394, 407, 434
Mac OS X, *see* NAPI installation
magnetic_field (*field*), 234, 488, 586
magnetic_kicker (*base class*), *see* NXmagnetic_kicker, **1016**
magnetic_wavelength (*field*), 424
magnification (*field*), 274, 282, 377, 450, 846
magnitude (*field*), 196, 646, 653–655
magnitude_errors (*field*), 196
magnitude_errors_model (*field*), 196
mailing lists, **1198**
manipulator (*base class*), *see* NXmanipulator, **430**
Mantid (*data analysis software*), 1207
manual source, 1215
manufacturer (*field*), 239
mapping_MAPPING (*field*), 237
marching_cubes (*field*), 929
mark (*field*), 991
marker (*field*), 1039
marker_edge_color (*field*), 1039
marker_face_color (*field*), 1040
marker_size (*field*), 1039
mask (*field*), 293, 565, 900, 925, 956, 963, 964, 969, 974, 975
mask (*group attribute*), 604
Mask_indices (*group attribute*), 604
mask_material (*field*), 402
mask_thickness (*field*), 402
mass (*field*), 202, 489, 495, 572

mass_calibration (*field*), 566
mass_resolution (*field*), 566
mass_resolution_fw (*field*), 567
mass_to_charge (*field*), 566, 878, 890, 968, 972
mass_to_charge_range (*field*), 203, 223, 571, 572
match (*field*), 898, 918, 920, 969, 991, 1018
match_filter (*base class*), *see* NXmatch_filter, **1018**
matching_phase (*field*), 360
matching_phase_descriptor (*field*), 360
material (*field*), 200, 452, 1070, 1071, 1074
material_other (*field*), 452
material_phase (*field*), 998
material_phase_comment (*field*), 998
MATLAB, 1207
    code examples, **108**
matrix_elements (*field*), 233
max (*field attribute*), 455
max (*field*), 949, 950, 969, 1158
max_delta_x (*field*), 1054
max_gpus (*field*), 297, 299, 976
max_iteration (*field*), 1054
max_mass_to_charge (*field*), 214, 569
max_physical_capacity (*field*), 293, 300
max_processes (*field*), 297, 299, 976
max_read_rate (*field*), 300
max_resident_memory_snapshot (*field*), 299
max_revolutions (*field*), 624
max_slew_rate (*field*), 1129
max_sothree_bandwidth (*field*), 1040
max_stwo_bandwidth (*field*), 1040
max_threads (*field*), 297, 299, 976
max_time (*field*), 1054
max_virtual_memory_snapshot (*field*), 299
max_write_rate (*field*), 300
max_x (*field*), 1054
maximum_incident_angle (*field*), 240
maximum_number_of_atoms_per_molecular_ion (*field*), 214, 571
maximum_value (*field*), 429, 1094, 1103
mcp_efficiency (*field*), 208
McStas, 32, 33
measured_data (*field*), 625, 1163
measured_data_errors (*field*), 625
measurement (*field*), 210, 430–432, 498, 561, 705, 706, 753
measurement_direction (*field*), 787
measurement_sensors (*field*), 1088
measurement_time (*field*), 1081, 1128
measurement_type (*field*), 1128, 1178
medium (*field*), 469
melting_temperature (*field*), 1049
member_count (*field*), 1092
memory (*field*), 1040
mesh_efficiency (*field*), 208
metadata, **25**, 66, 67, 73, 78, 146
method (*field*), 359, 693, 709, 722, 843, 897, 898, 909, 911, 917, 918, 920, 954, 960, 969, 991, 1018, 1179
methods_advise (*field*), 1040
metric (*field attribute*), 396, 847
Microsoft Windows, *see* NAPI installation
microstructure (*base class*), *see* NXmicrostructure, **1018**
microstructure_feature (*base class*), *see* NXmicrostructure_feature, **1030**
microstructure_ipf (*base class*), *see* NXmicrostructure_ipf, **1031**
microstructure_kanapy_results (*application definition*), *see* NXmicrostructure_kanapy_results, **1034**
microstructure_mtex_config (*base class*), *see* NXmicrostructure_mtex_config, **1038**
microstructure_odf (*base class*), *see* NXmicrostructure_odf, **1043**
microstructure_pf (*base class*), *see* NXmicrostructure_pf, **1046**
microstructure_score_config (*application definition*), *see* NXmicrostructure_score_config, **1047**
microstructure_score_results (*application definition*), *see* NXmicrostructure_score_results, **1059**
microstructure_slip_system (*base class*), *see* NXmicrostructure_slip_system, **1068**
miller (*field*), 361
miller_direction (*field*), 1069
miller_indices (*field*), 1046
miller_plane (*field*), 1068
mime_type (*group attribute*), 520
min (*field attribute*), 455
min (*field*), 949, 969, 1158
min_abundance (*field*), 203, 572
min_cluster_size (*field*), 892
min_half_life (*field*), 203, 572
min_mass_to_charge (*field*), 214, 569
min_pts (*field*), 891, 894
min_samples (*field*), 880, 886, 892
minimum_value (*field*), 429, 1094, 1103
mirror (*base class*), *see* NXmirror, **434**
misorientation_angle (*field*), 486
misorientation_axis (*field*), 486
misorientation_quaternion (*field*), 486
mobility (*field*), 1025, 1026, 1028
mobility_weight (*field*), 1064
mode (*field*), 347, 439, 506, 509, 683, 689, 691, 767, 770, 773, 777, 811, 814, 817, 820, 830, 835, 956, 1098, 1107
model (*field*), 282, 390, 498, 558–560, 640–645, 696–

698, 700–702, 706, 754, 1049, 1051, 1052, 1082, 1117, 1162
model (*file attribute*), 455
model_label (*field*), 894
model_name (*field*), 992, 993, 997, 999–1001
moderator (*base class*), *see* NXmoderator, **436**
modulation_frequency (*field*), 1014, 1157
modulation_signal (*field*), 1014, 1151
modulation_signal_type (*field*), 1152, 1153, 1155, 1156
modulation_status (*field*), 1014
module_offset (*field*), 336, 737
molecular_formula_hill (*field*), 1159
molecular_mass (*field*), 1159
monitor, 49
monitor (*base class*), *see* NXmonitor, **438**
monitor_applied (*field*), 591, 596
monochromator (*base class*), *see* NXmonochromator, **441**
monopd (*application definition*), *see* NXmonopd, **690**
mosaic_horizontal (*field*), 288
mosaic_vertical (*field*), 288
mosek (*field*), 1040
motivation, 3, *see* dictionary of terms, *see* exchange format, *see* format unification, *see* plotting, **12**, 25
mpes (*application definition*), *see* NXmpes, **692**
mpes_arpes (*application definition*), *see* NXmpes_arpes, **722**
mrp_flight_path_length (*field*), 570
mrp_mass_to_charge (*field*), 570
mrp_value (*field*), 570
mrp_voltage (*field*), 570
mtex (*field*), 1041
multi-dimensional data, **54**
multi-modal data, 175
multiple_outputs (*field*), 978
mx (*application definition*), *see* NXmx, **729**

N

n_ic_cluster (*field*), 885
n_mass_to_charge (*field*), 214, 569
n_max_ic_cluster (*field*), 880
name (*field attribute*), 603
name (*field*), 197, 198, 221, 277, 291, 319, 346, 349, 372, 387, 410, 425, 430, 460, 465, 487, 494, 498, 504, 529, 553, 558, 567, 571, 585, 588, 589, 591, 592, 596, 611, 614, 615, 631, 633, 640–644, 659, 683, 686, 689, 691, 695–697, 705, 706, 709, 731, 732, 741, 750, 753–755, 767, 769, 772, 773, 775–777, 781, 783, 787, 792, 794, 806, 810, 813, 814, 816, 817, 819, 822, 824, 825, 827, 829, 830, 832, 834, 884, 976, 983, 986, 988, 995, 996, 999–1001, 1012, 1053, 1081, 1087, 1088, 1095, 1098, 1104, 1107, 1116, 1117, 1139, 1159, 1162, 1164
name (*group attribute*), 616
name (*NXDL attribute*), 69, 70
NAME_spectrum (*field*), 625
NAMEcurrent (*field*), 1120
NAMEvoltage (*field*), 1120
naming convention, 41
NAPI, 4, **14**, 14, 80, **1184**, 1205, 1212
  bypassing, 58
  c, 1186
  c++, 1186
  core, 1185
  f77, 1186
  f90, 1186
  IDL, 1192
  installation, **1200**
  installation; download location, 1201
  installation; Mac OS X, 1202
  installation; RPM, 1201
  installation; source distribution, 1202
  installation; Windows, 1202
  java, 1187
natural_abundance_product (*field*), 202, 572
nature (*field*), 817, 819, 1098, 1107
ndattribute, **129**
Nelson, Mitchell, 12
NeXpy, 7
NeXpy (*data analysis software*), 1207
next_set_feature_to_cluster (*field*), 907
next_to_current_link (*field*), 906
next_to_current_link_type (*field*), 906
NeXus Application Programming Interface, *see* NAPI
NeXus Constructor, 1206
NeXus Definition Language, *see* NXDL
NeXus International Advisory Committee, *see* NIAC
NeXus link, **21**, 21, 23
NeXus webpage, **1197**
NeXus_release (*file attribute*), 483
NeXus_repository (*file attribute*), 483
NeXus_version (*file attribute*), 482
NIAC, 10, **1197**, 1197, 1212
noise (*field*), 1092
noise_level (*field*), 270
nominal (*field*), 439
nonNegativeUnbounded (*NXDL data type*), **169**
norm (*field*), 593, 598
normalization (*field*), 268, 554, 794, 910, 926
normalization_applied (*field*), 591, 596
normals (*field*), 267, 930, 931
note (*base class*), *see* NXnote, **443**
note (*field*), 592, 597

notes (*field*), 1094, 1102
nuclide_hash (*field*), 202, 222, 567, 571, 572, 934, 937
nuclide_list (*field*), 223, 571, 934
nuclide_whitelist (*field*), 909, 911
nuclides (*field*), 202, 572
num (*field*), 530
num_of_channels (*field*), 876
number (*field*), 392
number_of_atoms (*field*), 937
number_of_boundaries (*field*), 248, 927, 1061
number_of_bunches (*field*), 506
number_of_categorical_labels (*field*), 1092
number_of_core_members (*field*), 895
number_of_crystals (*field*), 1022, 1036, 1063
number_of_cycles (*field*), 328
number_of_dld_wires (*field*), 562
number_of_domains (*field*), 1055
number_of_edges (*field*), 245, 250, 259
number_of_faces (*field*), 245, 259, 929, 936, 957, 962
number_of_feature_types (*field*), 902, 903
number_of_features (*field*), 895
number_of_interfaces (*field*), 1024
number_of_ion_types (*field*), 214, 571
number_of_ions (*field*), 925, 937, 956, 974, 975
number_of_iterations (*field*), 917
number_of_junctions (*field*), 1025, 1027
number_of_lenses (*field*), 534
number_of_members (*field*), 896
number_of_noise_members (*field*), 895
number_of_numeric_labels (*field*), 1092
number_of_objects (*field*), 293, 565, 900, 963, 964, 969
number_of_phases (*field*), 1022, 1036, 1063
number_of_planes (*field*), 361
number_of_poles (*field*), 347
number_of_scan_points (*field*), 360, 361, 658, 659
number_of_solutions (*field*), 894
number_of_sweeps (*field*), 1129
number_of_targets (*field*), 895
number_of_total_vertices (*field*), 257, 260
number_of_trajectory_points (*field*), 1147
number_of_triangle_sets (*field*), 897
number_of_triangles (*field*), 900
number_of_triple_junctions (*field*), 1024
number_of_unique_vertices (*field*), 260, 266
number_of_vertices (*field*), 245, 250, 260, 929, 936, 957, 962
number_sections (*field*), 407
numbers, *see* floating-point numbers, *see* integers
numerical_aperture (*field*), 450, 1070
numerical_label (*field*), 895, 1092
NX
    used as NX class prefix, 24, 41
NX_ANGLE (*units type*), **171**
NX_ANY (*units type*), **171**
NX_AREA (*units type*), **171**
NX_BINARY (*data type*), **170**
NX_BOOLEAN (*data type*), **170**
NX_CCOMPLEX (*data type*), **170**
NX_CHAR (*data type*), **170**
NX_CHAR_OR_NUMBER (*data type*), **170**
NX_CHARGE (*units type*), **171**
NX_COMPLEX (*data type*), **170**
NX_COUNT (*units type*), **171**
NX_CROSS_SECTION (*units type*), **172**
NX_CURRENT (*units type*), **172**
NX_DATE_TIME (*data type*), **170**
NX_DIMENSIONLESS (*units type*), **172**
NX_EMITTANCE (*units type*), **172**
NX_ENERGY (*units type*), **172**
NX_FLOAT (*data type*), **170**
NX_FLUX (*units type*), **172**
NX_FREQUENCY (*units type*), **172**
NX_INT (*data type*), **170**
NX_LENGTH (*units type*), **172**
NX_MASS (*units type*), **172**
NX_MASS_DENSITY (*units type*), **172**
NX_MOLECULAR_WEIGHT (*units type*), **172**
NX_NUMBER (*data type*), **170**
NX_PCOMPLEX (*data type*), **171**
NX_PER_AREA (*units type*), **172**
NX_PER_LENGTH (*units type*), **172**
NX_PERIOD (*units type*), **172**
NX_POSINT (*data type*), **171**
NX_POWER (*units type*), **173**
NX_PRESSURE (*units type*), **173**
NX_PULSES (*units type*), **173**
NX_QUATERNION (*data type*), **171**
NX_SCATTERING_LENGTH_DENSITY (*units type*), **173**
NX_SOLID_ANGLE (*units type*), **173**
NX_TEMPERATURE (*units type*), **173**
NX_TIME (*units type*), **173**
NX_TIME_OF_FLIGHT (*units type*), **173**
NX_TRANSFORMATION (*units type*), **173**
NX_UINT (*data type*), **171**
NX_UNITLESS (*units type*), **173**
NX_VOLTAGE (*units type*), **174**
NX_VOLUME (*units type*), **174**
NX_WAVELENGTH (*units type*), **174**
NX_WAVENUMBER (*units type*), **174**
NXaberration (*base class*), **196**
    used in application definition, 630
    used in base class, 282
NXactivity (*base class*), **197**
    used in application definition, 693
    used in base class, 409, 516
NXactuator (*base class*), **198**

used in application definition, 630, 693, 744
used in base class, 342, 372, 387, 410, 425, 430, 1141
NXafm (*application definition*), **872**
NXamplifier (*base class*), **876**
used in application definition, 1116
NXaperture (*base class*), **199**
used in application definition, 602, 630, 693, 722, 1093, 1102
used in base class, 273, 282, 342, 380, 410, 425, 441, 503
NXapm (*application definition*), **547**
NXapm_charge_state_analysis (*base class*), **201**
used in application definition, 550
NXapm_compositionspace_config (*application definition*), **878**
NXapm_compositionspace_results (*application definition*), **881**
NXapm_event_data (*base class*), **203**
used in application definition, 550
used in base class, 212
NXapm_instrument (*base class*), **205**
used in application definition, 550
used in base class, 204, 212
NXapm_measurement (*base class*), **212**
used in application definition, 550
NXapm_paraprobe_clusterer_config (*application definition*), **889**
NXapm_paraprobe_clusterer_results (*application definition*), **893**
NXapm_paraprobe_distancer_config (*application definition*), **896**
NXapm_paraprobe_distancer_results (*application definition*), **899**
NXapm_paraprobe_intersector_config (*application definition*), **901**
NXapm_paraprobe_intersector_results (*application definition*), **905**
NXapm_paraprobe_nanochem_config (*application definition*), **908**
NXapm_paraprobe_nanochem_results (*application definition*), **924**
NXapm_paraprobe_ranger_config (*application definition*), **943**
NXapm_paraprobe_ranger_results (*application definition*), **944**
NXapm_paraprobe_selector_config (*application definition*), **945**
NXapm_paraprobe_selector_results (*application definition*), **945**
NXapm_paraprobe_spatstat_config (*application definition*), **946**
NXapm_paraprobe_spatstat_results (*application definition*), **951**
NXapm_paraprobe_surfacer_config (*application definition*), **953**
NXapm_paraprobe_surfacer_results (*application definition*), **955**
NXapm_paraprobe_tessellator_config (*application definition*), **959**
NXapm_paraprobe_tessellator_results (*application definition*), **961**
NXapm_paraprobe_tool_common (*base class*), **965**
used in application definition, 968, 975
NXapm_paraprobe_tool_config (*application definition*), **967**
NXapm_paraprobe_tool_parameters (*base class*), **971**
used in application definition, 889, 897, 901, 909, 944, 945, 947, 953, 959, 968
NXapm_paraprobe_tool_process (*base class*), **973**
used in application definition, 894, 899, 906, 925, 944, 946, 951, 955, 961, 975
NXapm_paraprobe_tool_results (*application definition*), **974**
NXapm_ranging (*base class*), **213**
used in application definition, 550
NXapm_reconstruction (*base class*), **215**
used in application definition, 550
NXapm_simulation (*base class*), **220**
used in application definition, 550
NXarchive (*application definition*), **583**
NXarpes (*application definition*), **587**
NXatom (*base class*), **220**
used in application definition, 550, 630, 882, 925
used in base class, 213, 268, 364, 410, 460, 526
NXattenuator (*base class*), **223**
used in application definition, 730, 857, 1093, 1102, 1161
used in base class, 425
NXazint1d (*application definition*), **590**
NXazint1d (*applications*), 591
NXazint2d (*application definition*), **595**
NXazint2d (*applications*), 595
NXbeam (*base class*), **225**
used in application definition, 630, 693, 730, 744, 764, 786, 843, 986, 1167, 1177
used in base class, 206, 342, 410, 425, 487, 982
NXbeam_splitter (*base class*), **977**
NXbeam_stop (*base class*), **231**
used in base class, 425
NXbeam_transfer_matrix_table (*base class*), **232**
used in application definition, 744
NXbending_magnet (*base class*), **234**

used in base class, 425
nxbrowse, 16
nxbrowse (*utility*), 1205
NXcalibration (*base class*), **236**
used in application definition, 693, 744, 843, 1116
used in base class, 269, 480, 1076, 1132, 1136, 1149
NXcanSAS (*application definition*), **600**
NXcanSAS (*applications*)
dQl, 610
dQw, 609
I, 607
Idev, 608
Q, 604
Qdev, 609
resolutions, 606
SASaperture, 611
SAScollimation, 611
SASdata, 603
SASdetector, 611
SASentry, 602
SASinstrument, 611
SASnote, 615
SASprocess, 615
SASprocessnote, 615
SASsample, 614
SASsource, 613
SAStransmission_spectrum, 616
Tdev, 617
NXcapillary (*base class*), **239**
used in base class, 425
NXcg_alpha_complex (*base class*), **240**
used in application definition, 955
NXcg_cylinder (*base class*), **242**
used in application definition, 909, 968
used in base class, 265, 1113
NXcg_ellipsoid (*base class*), **243**
used in application definition, 968
used in base class, 265, 1113
NXcg_face_list_data_structure (*base class*), **244**
used in application definition, 925, 955, 961, 968
used in base class, 252, 255, 257, 259, 265, 266
NXcg_grid (*base class*), **247**
used in application definition, 882, 925, 1035, 1048, 1060
used in base class, 991, 1011, 1021, 1031
NXcg_half_edge_data_structure (*base class*), **249**
used in base class, 252, 257, 259, 265
NXcg_hexahedron (*base class*), **251**
used in application definition, 925, 961, 968, 1060
used in base class, 265, 1113
NXcg_parallelogram (*base class*), **254**
used in base class, 265
NXcg_point (*base class*), **255**
used in base class, 241, 1021
NXcg_polygon (*base class*), **256**
used in base class, 1021
NXcg_polyhedron (*base class*), **258**
used in application definition, 925, 961, 968
used in base class, 248, 265, 1021, 1113
NXcg_polyline (*base class*), **259**
used in base class, 1021
NXcg_primitive (*base class*), **261**
NXcg_roi (*base class*), **264**
used in application definition, 925
NXcg_tetrahedron (*base class*), **265**
used in application definition, 955
used in base class, 241
NXcg_triangle (*base class*), **266**
used in application definition, 925, 955
used in base class, 241, 1021
NXcg_unit_normal (*base class*), **267**
used in application definition, 925
used in base class, 252, 261
NXchemical_composition (*base class*), **268**
used in application definition, 550, 925
used in base class, 1021, 1031
NXcircuit (*base class*), **269**
used in base class, 293, 295, 300, 497, 1128
NXcite (*base class*), **271**
used in application definition, 550, 630, 998, 1006
NXclass (*attribute*), 41
NXcollection (*base class*), **272**
used in application definition, 550, 602, 630, 730, 780, 872, 882, 986, 1006, 1035, 1048, 1060, 1093, 1102, 1116, 1151, 1154
used in base class, 216, 320, 355, 383, 425, 445, 518, 1038
NXcollectioncolumn (*base class*), **273**
used in application definition, 693, 722, 843
used in base class, 349
NXcollimator (*base class*), **275**
used in application definition, 602, 771, 775
used in base class, 425
NXcomponent (*base class*), **276**
used in application definition, 550, 630, 744
used in base class, 206, 342, 372, 410, 1076
NXcontainer (*base class*), **982**
nxconvert (*utility*), 1205

NXcoordinate_system (*base class*), **278**
used in application definition, 550, 630, 693, 722, 744, 843, 975, 1006, 1060
used in base class, 355, 966, 1128
NXcorrector_cs (*base class*), **281**
used in application definition, 630
used in base class, 342
NXcrystal (*base class*), **285**
used in application definition, 690, 810, 1093, 1102
used in base class, 425, 441
NXcs_computer (*base class*), **291**
used in base class, 297
NXcs_filter_boolean_mask (*base class*), **292**
used in application definition, 550, 899, 925, 955, 961, 968, 975
used in base class, 974, 1113
NXcs_memory (*base class*), **293**
used in base class, 291
NXcs_prng (*base class*), **294**
used in application definition, 947
NXcs_processor (*base class*), **295**
used in base class, 291
NXcs_profiling (*base class*), **296**
used in application definition, 550, 630, 882, 968, 975, 1006, 1035, 1048, 1060
used in base class, 966
NXcs_profiling_event (*base class*), **298**
used in base class, 297
NXcs_storage (*base class*), **300**
used in base class, 291
NXcsg (*base class*), **984**
used in base class, 984, 1112
NXcxi_ptycho (*application definition*), **985**
NXcylindrical_geometry (*base class*), **300**
used in base class, 231, 320
NXdata (*base class*), 5, 57, **301**
used in application definition, 550, 588, 591, 595, 602, 622, 630, 682, 686, 688, 690, 693, 722, 730, 744, 766, 769, 771, 775, 779, 780, 782, 786, 806, 810, 813, 816, 819, 821, 824, 827, 829, 832, 833, 837, 838, 840, 842, 843, 857, 882, 925, 986, 998, 1006, 1012, 1035, 1048, 1060, 1086, 1093, 1102, 1116, 1161, 1167, 1177, 1178
used in base class, 213, 216, 220, 226, 234, 236, 239, 286, 320, 349, 355, 364, 376, 379, 383, 393, 396, 404, 406, 413, 423, 434, 437, 441, 445, 459, 466, 480, 487, 494, 503, 509, 518, 532, 978, 1031, 1043, 1046, 1072, 1083, 1128, 1146, 1149
NXdeflector (*base class*), **319**
used in application definition, 630
used in base class, 273, 282, 342, 349, 380, 410, 497, 503, 516
NXdelocalization (*base class*), **990**
used in application definition, 925
NXdetector (*base class*), **320**
used in application definition, 550, 588, 602, 630, 682, 688, 690, 730, 744, 766, 769, 771, 775, 779, 786, 806, 810, 813, 816, 819, 821, 824, 829, 833, 837, 838, 841, 842, 857, 872, 986, 1093, 1102, 1161, 1167, 1177, 1178
used in base class, 206, 349, 372, 425
NXdetector_channel (*base class*), **332**
used in application definition, 730
used in base class, 320
NXdetector_group (*base class*), **334**
used in application definition, 730
used in base class, 425
NXdetector_module (*base class*), **336**
used in application definition, 730
used in base class, 320
nxdir (*utility*), 1205
NXdirecttof (*application definition*), **620**
NXdisk_chopper (*base class*), **338**
used in application definition, 620, 769, 1093, 1102
used in base class, 425
NXdispersion (*base class*), **992**
used in application definition, 998
NXdispersion_function (*base class*), **993**
used in application definition, 998
used in base class, 992
NXdispersion_repeated_parameter (*base class*), **995**
used in application definition, 998
used in base class, 993
NXdispersion_single_parameter (*base class*), **995**
used in application definition, 998
used in base class, 993
NXdispersion_table (*base class*), **996**
used in application definition, 998
used in base class, 992
NXdispersive_material (*application definition*), **997**
NXdistortion (*base class*), **341**
used in application definition, 693
NXDL, 24, 79, 141, **145**, 145
NXDL elements, **146**
NXDL template file, **69**
nxdl_to_hdf5.py, 1206
NXebeam_column (*base class*), **342**
used in application definition, 630
used in base class, 372
NXelectromagnetic_lens (*base class*), **346**
used in application definition, 550, 630
used in base class, 273, 282, 342, 349, 380, 410, 503, 516

NXelectron_detector (*base class*), **348**
used in application definition, 693
NXelectronanalyzer (*base class*), **349**
used in application definition, 693, 722, 843
NXelectrostatic_kicker (*base class*), **1004**
NXellipsometry (*application definition*), **621**
NXem (*application definition*), **627**
NXem_calorimetry (*application definition*), **1005**
NXem_ebsd (*base class*), **353**
used in application definition, 630
NXem_eds (*base class*), **364**
used in application definition, 630
NXem_eels (*base class*), **366**
used in application definition, 630
NXem_event_data (*base class*), **367**
used in application definition, 630
used in base class, 376
NXem_img (*base class*), **370**
used in application definition, 630
NXem_instrument (*base class*), **371**
used in application definition, 630
used in base class, 369, 376
NXem_interaction_volume (*base class*), **375**
used in application definition, 630
NXem_measurement (*base class*), **376**
used in application definition, 630
NXem_optical_system (*base class*), **377**
used in application definition, 630
used in base class, 372
NXem_simulation (*base class*), **379**
used in application definition, 630
NXenergydispersion (*base class*), **380**
used in application definition, 693, 722, 843
used in base class, 349
NXentry (*base class*), 5, **382**
used in application definition, 550, 584, 588, 591, 595, 602, 620, 622, 630, 682, 685, 686, 688, 690, 693, 722, 730, 744, 764, 766, 769, 771, 775, 779, 780, 782, 786, 806, 810, 813, 816, 819, 821, 824, 827, 829, 832, 833, 837, 838, 840–843, 857, 872, 878, 882, 889, 894, 897, 899, 901, 906, 909, 925, 944–947, 951, 953, 955, 959, 961, 968, 975, 986, 998, 1006, 1012, 1035, 1048, 1060, 1086, 1093, 1102, 1116, 1151, 1154, 1161, 1167, 1177, 1178
used in base class, 482
NXenvironment (*base class*), **387**
used in application definition, 693, 744, 872, 1012, 1086, 1116, 1151, 1154
used in base class, 487
NXevent_data (*base class*), **388**
used in application definition, 1093
used in base class, 425
NXfabrication (*base class*), **390**
used in application definition, 550, 622, 630, 693, 744, 1116, 1161
used in base class, 206, 269, 277, 342, 349, 372, 380, 425, 503, 876, 1014, 1076, 1081
NXfermi_chopper (*base class*), **391**
used in application definition, 620, 780, 1102
used in base class, 425
NXfilter (*base class*), **393**
used in base class, 425
NXfit (*base class*), **396**
used in application definition, 693, 843
NXfit_function (*base class*), **399**
used in application definition, 843
used in base class, 396, 459
NXflipper (*base class*), **400**
used in base class, 425
NXfluo (*application definition*), **682**
NXfresnel_zone_plate (*base class*), **401**
NXgeometry (*base class*), **403**
used in application definition, 771, 775, 1093, 1102
used in base class, 200, 231, 234, 275, 286, 320, 339, 387, 391, 393, 406, 423, 434, 437, 439, 441, 454, 487, 498, 503, 525, 530
NXgrating (*base class*), **404**
used in application definition, 1161
used in base class, 441
NXguide (*base class*), **406**
used in base class, 425
NXhistory (*base class*), **409**
used in application definition, 693, 744, 1086, 1116
used in base class, 425, 487, 494, 516
NXibeam_column (*base class*), **409**
used in application definition, 630
used in base class, 372
NXimage (*base class*), **412**
used in application definition, 550, 630
used in base class, 282, 364, 369, 371
NXindirecttof (*application definition*), **684**
nxingest (*utility*), 1205
NXinsertion_device (*base class*), **423**
used in application definition, 693
used in base class, 425
NXinstrument (*base class*), 6, **425**
used in application definition, 584, 588, 591, 595, 602, 620, 622, 682, 685, 686, 688, 690, 693, 722, 730, 744, 764, 766, 769, 771, 775, 779, 780, 782, 786, 806, 810, 813, 816, 819, 821, 824, 827, 829, 833, 837, 838, 840–

843, 857, 872, 986, 1012, 1086, 1093, 1102, 1116, 1151, 1154, 1161, 1167, 1177, 1178
used in base class, 383, 518
NXiqproc (*application definition*), **685**
NXisocontour (*base class*), **1010**
used in application definition, 925
NXiv_temp (*application definition*), **1011**
NXlauetof (*application definition*), **688**
NXlockin (*base class*), **1013**
used in application definition, 1116, 1151, 1154
NXlog (*base class*), **427**
used in application definition, 693, 1093, 1102
used in base class, 275, 286, 393, 430, 437, 439, 445, 461, 487, 498, 1004, 1017, 1080, 1090, 1111, 1114
NXmagnetic_kicker (*base class*), **1016**
NXmanipulator (*base class*), **430**
used in application definition, 550, 630, 693, 744
used in base class, 206, 372
NXmatch_filter (*base class*), **1018**
used in application definition, 897, 909, 968
used in base class, 972, 991
NXmicrostructure (*base class*), **1018**
used in application definition, 1035, 1048, 1060
NXmicrostructure_feature (*base class*), **1030**
used in application definition, 1035, 1060
used in base class, 1021
NXmicrostructure_ipf (*base class*), **1031**
NXmicrostructure_kanapy_results (*application definition*), **1034**
NXmicrostructure_mtex_config (*base class*), **1038**
NXmicrostructure_odf (*base class*), **1043**
NXmicrostructure_pf (*base class*), **1046**
NXmicrostructure_score_config (*application definition*), **1047**
NXmicrostructure_score_results (*application definition*), **1059**
NXmicrostructure_slip_system (*base class*), **1068**
NXmirror (*base class*), **434**
used in base class, 425
NXmoderator (*base class*), **436**
used in application definition, 1093, 1102
used in base class, 425
NXmonitor (*base class*), **438**
used in application definition, 591, 595, 682, 688, 690, 766, 769, 771, 775, 779, 806, 810, 813, 816, 819, 821, 824, 829, 833, 986, 1093, 1102
used in base class, 383, 518
NXmonochromator (*base class*), **441**
used in application definition, 588, 591, 595, 630, 682, 685, 693, 744, 766, 771, 806, 829, 833, 1161
used in base class, 342, 410, 425
NXmonopd (*application definition*), **690**
NXmpes (*application definition*), **692**
NXmpes_arpes (*application definition*), **722**
NXmx (*application definition*), **729**
NXnote (*base class*), **443**
used in application definition, 550, 602, 630, 786, 878, 882, 889, 897, 901, 909, 947, 959, 968, 975, 1006, 1048, 1060, 1086, 1093, 1102, 1167
used in base class, 197, 200, 213, 216, 236, 320, 355, 383, 387, 409, 413, 445, 466, 480, 503, 509, 518, 534, 966, 972, 1159
NXobject (*base class*), **444**
used in application definition, 1178
NXoff_geometry (*base class*), **447**
used in base class, 200, 224, 231, 234, 275, 286, 320, 339, 391, 393, 404, 406, 423, 434, 437, 439, 441, 487, 498, 503, 530, 534, 1112
NXoptical_fiber (*base class*), **1069**
NXoptical_lens (*base class*), **449**
used in application definition, 622, 744
used in base class, 282
NXoptical_polarizer (*base class*), **1072**
NXoptical_spectroscopy (*application definition*), **744**
NXoptical_window (*base class*), **451**
used in application definition, 744
NXorientation (*base class*), **453**
used in application definition, 1093, 1102
used in base class, 403, 498
NXparameters (*base class*), **454**
used in application definition, 550, 591, 595, 630, 686, 782, 786, 827, 832, 843, 878, 901, 953, 1048, 1116
used in base class, 202, 206, 216, 220, 236, 379, 383, 399, 445, 466, 480, 518, 1043, 1046, 1136, 1149
NXpdb (*base class*), **456**
NXpeak (*base class*), **458**
used in application definition, 550, 843
used in base class, 213, 364, 367, 396
NXphase (*base class*), **459**
used in application definition, 630
used in base class, 355, 1021
NXphase_lock_loop (*base class*), **1075**
used in application definition, 872
NXpid_controller (*base class*), **460**
used in application definition, 693, 744, 1086, 1116, 1154
used in base class, 198, 430, 1128, 1141

NXpinhole (*base class*), **463**
NXplot (*utility*), 1206
NXpolarizer (*base class*), **463**
used in application definition, 1093, 1102
used in base class, 425
NXpositioner (*base class*), **464**
used in application definition, 872, 1093, 1102, 1167
used in base class, 200, 425, 430, 487, 1132
NXprocess, 74
NXprocess (*base class*), **466**
used in application definition, 550, 591, 595, 602, 630, 686, 744, 782, 786, 827, 832, 878, 882, 889, 894, 906, 909, 925, 947, 951, 1006, 1060, 1086, 1116, 1167, 1177
used in base class, 213, 216, 220, 282, 355, 364, 367, 376, 379, 383, 413, 452, 482, 509, 518, 1021, 1043, 1091
NXprogram (*base class*), **467**
used in application definition, 550, 622, 630, 744, 882, 968, 975, 1006, 1035, 1048, 1060
used in base class, 213, 216, 220, 277, 294, 364, 367, 379, 413, 509, 966, 1021
NXpump (*base class*), **468**
used in application definition, 550, 630
used in base class, 206, 372
NXquadric (*base class*), **1078**
used in base class, 1112
NXquadrupole_magnet (*base class*), **1080**
NXraman (*application definition*), **763**
NXrcs (*base class*), **1081**
used in application definition, 1116
NXreflections (*base class*), **469**
used in application definition, 786
NXrefscan (*application definition*), **766**
NXreftof (*application definition*), **768**
NXregion (*base class*), **1082**
NXregistration (*base class*), **479**
used in application definition, 693
NXresolution (*base class*), **479**
used in application definition, 693, 722, 744, 1161
used in base class, 349
NXroi_process (*base class*), **481**
used in application definition, 550, 630, 909
NXroot (*base class*), **482**
attributes, **21**
NXrotations (*base class*), **484**
used in base class, 355, 1021
NXsample (*base class*), 6, **487**
used in application definition, 550, 584, 588, 602, 622, 630, 682, 686, 688, 690, 693, 722, 730, 744, 766, 769, 771, 775, 779, 780, 782, 786, 806, 810, 813, 816, 819, 821, 824, 827, 829, 832, 833, 837, 838, 842, 843, 857, 882, 986, 998, 1006, 1012, 1048, 1086, 1093, 1102, 1116, 1161, 1167, 1178
used in base class, 383, 449, 518, 532, 978, 1069, 1072
NXsample_component (*base class*), **494**
used in base class, 487
NXsas, 1211
NXsas (*application definition*), **771**
NXsastof (*application definition*), **775**
NXscan (*application definition*), **778**
NXscan_controller (*base class*), **496**
used in application definition, 630
used in base class, 342, 410
NXsensor (*base class*), **498**
used in application definition, 550, 630, 693, 744, 872, 1012, 1086, 1116, 1151, 1154
used in base class, 206, 342, 372, 387, 393, 410, 425, 430, 461
NXsensor_scan (*application definition*), **1085**
NXseparator (*base class*), **1090**
NXshape (*base class*), **501**
used in application definition, 771, 775, 1093, 1102
used in base class, 286, 403, 404, 434, 978, 982, 1072
NXsimilarity_grouping (*base class*), **1091**
used in application definition, 894
NXslit (*base class*), **502**
used in application definition, 1161
NXsnsevent (*application definition*), **1093**
NXsnshisto (*application definition*), **1102**
NXsolenoid_magnet (*base class*), **1111**
NXsolid_geometry (*base class*), **1112**
NXsource (*base class*), **503**
used in application definition, 550, 584, 588, 591, 595, 602, 630, 682, 686, 690, 693, 730, 744, 766, 771, 775, 782, 786, 806, 810, 821, 824, 827, 829, 833, 840, 843, 986, 1093, 1102, 1161, 1178
used in base class, 206, 342, 410, 425
NXspatial_filter (*base class*), **1113**
used in application definition, 968
used in base class, 972
NXspe (*application definition*), **780**
NXspectrum (*base class*), **509**
used in application definition, 630
used in base class, 367, 369
NXspin_rotator (*base class*), **1114**
NXspindispersion (*base class*), **516**
used in base class, 349
NXspm (*application definition*), **1115**
NXspm_bias_spectroscopy (*base class*), **1127**

used in application definition, 1116, 1154
NXspm_cantilever (*base class*), **1131**
used in application definition, 872
NXspm_cantilever_config (*base class*), **1132**
used in base class, 1132
NXspm_cantilever_oscillator (*base class*), **1134**
used in application definition, 872
used in base class, 1132
NXspm_piezo_config (*base class*), **1135**
used in base class, 1138
NXspm_piezo_sensor (*base class*), **1137**
used in application definition, 872, 1116
NXspm_piezoelectric_material (*base class*), **1139**
used in base class, 1136, 1138
NXspm_positioner (*base class*), **1140**
used in application definition, 1116, 1154
used in base class, 1128, 1138
NXspm_scan_control (*base class*), **1143**
used in application definition, 1116, 1151, 1154
used in base class, 1128
NXspm_scan_pattern (*base class*), **1145**
used in application definition, 1116
used in base class, 1128, 1143
NXspm_scan_region (*base class*), **1148**
used in application definition, 1116
used in base class, 1128, 1143
NXspm_temperature_sensor (*base class*), **1149**
used in application definition, 1116
NXsqom (*application definition*), **782**
NXstm (*application definition*), **1151**
NXstress (*application definition*), **784**
NXsts (*application definition*), **1154**
NXstxm (*application definition*), **806**
NXsubentry (*base class*), **517**
used in base class, 383
NXsubsampling_filter (*base class*), **1158**
used in application definition, 968
used in base class, 972
NXsubstance (*base class*), **1159**
nxsummary, 1205
NXtas (*application definition*), **809**
NXtofnpd (*application definition*), **813**
NXtofraw (*application definition*), **815**
NXtofsingle (*application definition*), **818**
NXtomo (*application definition*), **821**
NXtomophase (*application definition*), **824**
NXtomoproc (*application definition*), **827**
NXtransformations (*base class*), **521**
used in application definition, 722, 730, 744, 786, 843, 986
used in base class, 206, 226, 277, 280, 387, 404, 425, 479, 982
nxtranslate (*utility*), 1206
NXtranslation (*base class*), **525**
used in application definition, 1093, 1102
used in base class, 403
NXtransmission (*application definition*), **1161**
NXunit_cell (*base class*), **526**
used in application definition, 630
used in base class, 460
NXuser (*base class*), **528**
used in application definition, 550, 584, 630, 693, 744, 786, 813, 816, 819, 882, 975, 1006, 1035, 1048, 1086, 1093, 1102, 1161
used in base class, 369, 383, 518, 966
nxvalidate, 78
NXvelocity_selector (*base class*), **529**
used in base class, 425, 441
NXwaveplate (*base class*), **531**
used in application definition, 622, 744
NXxas (*application definition*), **829**
NXxasproc (*application definition*), **831**
NXxbase (*application definition*), **833**
NXxeuler (*application definition*), **836**
NXxkappa (*application definition*), **838**
NXxlaue (*application definition*), **839**
NXxlaueplate (*application definition*), **840**
NXxnb (*application definition*), **841**
NXxpcs (*application definition*), **1166**
NXxps (*application definition*), **843**
NXxraylens (*base class*), **534**
used in base class, 425
NXxrd (*application definition*), **1176**
NXxrd_pan (*application definition*), **1178**
NXxrot (*application definition*), **856**

O

o (*field*), 797
o_errors (*field*), 797
object (*base class*), *see* NXobject, **444**
observed_frame (*field*), 472
observed_frame_errors (*field*), 472
observed_frame_var (*field*), 472
observed_phi (*field*), 473
observed_phi_errors (*field*), 473
observed_phi_var (*field*), 473
observed_px_x (*field*), 472
observed_px_x_errors (*field*), 473
observed_px_x_var (*field*), 472
observed_px_y (*field*), 473
observed_px_y_errors (*field*), 473
observed_px_y_var (*field*), 473
observed_x (*field*), 473
observed_x_errors (*field*), 474
observed_x_var (*field*), 474
observed_y (*field*), 474
observed_y_errors (*field*), 474

observed_y_var (*field*), 474
occupancy (*field*), 222
odd_layer_density (*field*), 435
odd_layer_material (*field*), 435
odf (*field*), 1041
off_geometry (*base class*), *see* NXoff_geometry, **447**
offset (*field attribute*), **28**, 229, 337, 389, 524, 738, 828
offset (*field*), 238, 241, 270, 318, 708, 956
offset_azimuth (*field*), 725
offset_polar (*field*), 726
offset_tilt (*field*), 725
offset_units (*field attribute*), 337, 524
offset_value (*field*), 1120
offset_values (*field*), 954
offset_x (*field*), 320
offset_y (*field*), 320
omega (*field*), 747, 1181
open_gl_bug (*field*), 1040
OpenGENIE (*data analysis software*), 1207
openloop_amplification (*field*), 876
operating_frequency (*field*), 270
operating_system (*field*), 291
operation (*field*), 985
operation_mode (*field*), 342, 410, 552, 652, 655, 656
optical_fiber (*base class*), *see* NXoptical_fiber, **1069**
optical_lens (*base class*), *see* NXoptical_lens, **449**
optical_loss (*field*), 979
optical_polarizer (*base class*), *see* NXoptical_polarizer, **1072**
optical_spectroscopy (*application definition*), *see* NXoptical_spectroscopy, **744**
optical_window (*base class*), *see* NXoptical_window, **451**
ORCID (*field*), 529
orcid (*field*), 1087
order (*field*), 1077
order (*transformation*), *see* depends on (field attribute)
order_no (*field*), 287
orientation (*base class*), *see* NXorientation, **453**
orientation (*field*), 263, 268, 930, 931, 937, 969, 1051
orientation_angle (*field*), 453
orientation_euler (*field*), 486
orientation_matrix (*field*), 287, 394, 489, 495, 689, 811, 834
orientation_quaternion (*field*), 486
orientation_spread (*field*), 1023
origin (*field*), 248, 280, 355, 558, 635–637, 639, 657, 844, 883, 925, 1035, 1060, 1061
original_axis (*field*), 237
original_centre (*field*), 341
original_points (*field*), 341
Osborn, Raymond, 1213
oscillating (*field*), 1147
oscillation_frequency (*field*), 1147
outer_diameter (*field*), 402
outer_plot_spacing (*field*), 1039
outermost_zone_width (*field*), 402
output (*field attribute*), 234
output_channels (*field*), 270
output_frequency_range (*field*), 1077
output_heater_power (*field*), 431, 705
output_impedance (*field*), 270
output_signal (*field*), 270
output_slew_rate (*field*), 271
outputs (*field*), 277
outputVALUE (*field*), 199
overlaps (*field*), 476

P

P_pyroelectric_constant (*field*), 1139
packing_fraction (*field*), 983
pair_separation (*field*), 339
PARAMETER (*field*), 238, 400, 455
parameter_reliability (*field*), 748
parameter_units (*field*), 994, 995
parameters (*base class*), *see* NXparameters, **454**
parameters (*field*), 1079
parent (*field*), 1083
parent_mask (*field*), 1084
partiality (*field*), 471
pass_energy (*field*), 380, 589, 701
path_length (*field*), 490
path_length_window (*field*), 490
pdb (*base class*), *see* NXpdb, **456**
peak (*base class*), *see* NXpeak, **458**
peak (*field*), 366
peak_power (*field*), 506
period (*field*), 404, 505
pf (*field*), 1041
pf_anno_fun_hdl (*field*), 1039
pf_extensions (*field*), 1041
phase (*base class*), *see* NXphase, **459**
phase (*field*), 340, 424
phase_bandwidth (*field*), 1076
phase_difference (*field*), 1014
phase_error (*field*), 1077
phase_id (*field*), 359, 460
phase_lock_loop (*base class*), *see* NXphase_lock_loop, **1075**
phase_modulation_bandwidth (*field*), 1014
phase_name (*field*), 799
phase_shift (*field*), 1134
phases_per_scan_point (*field*), 359
phi (*field*), 747, 837, 838, 913, 1181
photon_energy (*field*), 703, 712
physical file format, *see* file format
physical_form (*field*), 491, 632, 709, 756

physical_quantity (*field*), 199, 236, 350, 351, 480, 656, 695, 700, 707, 708, 723, 754, 755
pid_controller (*base class*), *see* NXpid_controller, **460**
pinhole (*base class*), *see* NXpinhole, **463**
pinhole_position (*field*), 209
pitch (*field*), 612, 614
pixel_id (*field*), 1096, 1105
pixel_mask (*field*), 325, 334, 734
pixel_mask_applied (*field*), 325, 334, 734
pixel_shape (*field*), 360
pixel_x (*field*), 702
pixel_y (*field*), 702
plotting, 4, 6, 13, 15, 20, 24, **25**, 36, 58, 80, 236, 312, 314, 316, 317, 383, 445, 483, 518, 591, 596, 602, 1086, 1206
    how to find data, 49
point_group (*field*), 490, 496, 527
point_normal_form (*field*), 935
poison_depth (*field*), 437
poison_material (*field*), 437
polar (*field*), 751, 781
polar_angle (*field*), 289, 323, 476, 685, 688, 691, 767, 769, 773, 776, 810, 811, 814, 816, 819, 837, 838, 842, 857, 1096, 1105, 1177
polar_width (*field*), 781
polarization_applied (*field*), 591, 596
polarizer (*base class*), *see* NXpolarizer, **463**
polarizer (*field*), 1162
polarizer_angle (*field*), 1073
polarizing (*field*), 978
poles (*field*), 424
polylines (*field*), 260
populated_elements (*field attribute*), 1170
porto_notation_vectors (*field attribute*), 765
position (*field*), 222, 248, 250, 256, 374, 459, 571, 656, 848, 850, 878, 883, 890, 968, 972, 1007, 1026, 1027
positioner (*base class*), *see* NXpositioner, **464**
power (*field*), 209, 424, 505, 561, 845
power_consumption (*field*), 270
power_loss (*field*), 1070
power_setting (*field*), 347, 652, 656
power_source (*field*), 270
pre_factor (*field*), 1052
pre_sample_flightpath (*field*), 384, 520, 813, 816, 819
precompiled executable, *see* NAPI installation
predicted_frame (*field*), 471
predicted_phi (*field*), 472
predicted_px_x (*field*), 472
predicted_px_y (*field*), 472
predicted_x (*field*), 471
predicted_y (*field*), 472
preparation_date (*field*), 490, 555, 586, 633, 756
preset (*field*), 439, 683, 689, 691, 767, 770, 773, 777, 811, 814, 817, 820, 830, 835
pressure (*field*), 488, 586
previous_source (*field*), 506
prf_cc (*field*), 475
primary (*field attribute*), 321, 322
primary_element (*field*), 217, 568
primary_horizontal_distance (*field*), 791
primary_horizontal_evaluation (*field*), 791
primary_horizontal_sample_width (*field*), 791
primary_horizontal_source_width (*field*), 791
primary_horizontal_type (*field*), 791
primary_vertical_distance (*field*), 791
primary_vertical_evaluation (*field*), 791
primary_vertical_sample_width (*field*), 789
primary_vertical_source_width (*field*), 789
primary_vertical_type (*field*), 789
probability_mass (*field*), 952
probe (*field*), 343, 377, 504, 585, 588, 592, 596, 641, 683, 686, 691, 696, 697, 767, 772, 776, 783, 788, 806, 810, 822, 824, 827, 829, 834, 986, 1095, 1104
probe_current (*field*), 378
procedure (*field*), 453
process (*base class*), *see* NXprocess, **466**
process_id (*field*), 962
Processed Data, 74
processing_type (*field*), 786
profile (*field*), 740
program (*base class*), *see* NXprogram, **467**
program (*field*), 387, 466, 467, 552, 553, 563–571, 585, 592, 596, 626, 631, 640, 657, 686, 754, 758, 783, 794, 827, 832, 882, 969, 975, 1006, 1035, 1049, 1060, 1086
program_name (*field*), 384, 519, 780
program_url (*field attribute*), 1087
programs, 1205
projection (*field*), 274, 700
projection_direction (*field*), 1032
protection_features (*field*), 270
protocol_name (*field*), 217, 568
proton_charge (*field*), 1094, 1103
psi (*field*), 781
pub_chem_link (*field attribute*), 1160
pulse_delay (*field*), 228
pulse_duration (*field*), 228
pulse_energy (*field*), 209, 228, 506, 561
pulse_fraction (*field*), 208, 560
pulse_frequency (*field*), 208, 560
pulse_height (*field*), 389
pulse_id (*field*), 205
pulse_id_offset (*field*), 204
pulse_mode (*field*), 208, 560

pulse_number (*field*), 208, 560
pulse_time (*field*), 1096
pulse_voltage (*field*), 208, 560
pulse_width (*field*), 506
pump (*base class*), *see* NXpump, **468**
punx (*utility*), 1206
PyMCA (*data analysis software*), 1208

Q

Q (*field*), 604
q (*field*), 1181
Q_indices (*group attribute*), 604
q_norm (*field*), 1181
q_parallel (*field*), 1181
q_perpendicular (*field*), 1181
Qdev (*field*), 609
qh (*field*), 810
qk (*field*), 810
ql (*field*), 810
Qmean (*field*), 610
quadric (*base class*), *see* NXquadric, **1078**
quadrupole_magnet (*base class*), *see* NXquadrupole_magnet, **1080**
quality (*field*), 213, 216, 558
qx (*field*), 687, 783, 799
qy (*field*), 687, 783, 800
qz (*field*), 783, 800

R

r_slit (*field*), 391
radial_axis (*field*), 593, 597
radial_axis_edges (*field*), 593, 598
radiation (*field*), 613
radii (*field*), 242, 244, 921, 969
radius (*field*), 242, 244, 339, 381, 391, 530, 846, 1053
radius_indices (*group attribute*), 1053
raman (*application definition*), *see* NXraman, **763**
raman_experiment_type (*field*), 765
Raman-BC, 189
randomize_iontypes (*field*), 947
range (*field*), 439
rangeN (*field*), 1136
ranging_definitions (*field*), 879, 968, 972
rank, 15, 54, 58, 70
rank (*NXDL attribute*), 70
ratio (*field*), 340
ratio_k_alphatwo_k_alphaone (*field*), 1179
raw (*field*), 702, 749
raw_data_file (*field*), 793
raw_file (*field*), 828, 832
raw_frames (*field*), 1094, 1103
raw_time_of_flight (*field*), 321
raw_tof (*field*), 565
raw_value (*field*), 428, 465
rcs (*base class*), *see* NXrcs, **1081**
rdeform_field (*field*), 342
read file, 16
real (*field*), 414–416, 418, 419, 561, 562, 645–648
real_time (*field*), 324
realloc (*field*), 1055
reconstructed_positions (*field*), 216, 567
recover_evaporation_id (*field*), 889
recrystallized_grain_id (*field*), 1065
rediscretization (*field*), 1055
ref_offset_phaseN (*field*), 1015
reference (*field*), 592, 597, 1036
reference_amplitude (*field*), 873, 1014, 1134
reference_beam (*field attribute*), 228
reference_data_link (*field*), 453, 626
reference_frame (*field*), 485, 526
reference_frequency (*field*), 873, 1014, 1077, 1134, 1152, 1153, 1156
reference_peak (*field*), 708
reference_phase (*field*), 874, 1014, 1077, 1134
reference_plane (*field*), 229
reflectance (*field*), 450, 532, 979
reflection (*field*), 288, 464, 1073
reflection_id (*field*), 470
reflections (*base class*), *see* NXreflections, **469**
reflectivity (*field*), 758
refractive_index (*field*), 997, 999, 1000
refscan (*application definition*), *see* NXrefscan, **766**
reftof (*application definition*), *see* NXreftof, **768**
region (*base class*), *see* NXregion, **1082**
region_origin (*field*), 589
region_size (*field*), 589
region_type (*file attribute*), 1083
registration (*base class*), *see* NXregistration, **479**
regular expression, 41
regularization (*field*), 241
relative_area (*field*), 397
relative_atomic_concentration (*field*), 848
relative_intensity (*field*), 352, 361, 709
relative_molecular_mass (*field*), 489, 495, 983
relative_permittivity (*field*), 1139
relative_resolution (*field*), 351, 480, 696
relative_resolution_errors (*field*), 480
relative_sensitivity_factor (*field*), 397
release
    NeXus definitions, **1202**
    notes, 1202
    process, 1202
    tags, 141, 1203
    versioning, 141, 1203
release_date (*field*), 585
repository, 1199, *see* NAPI installation
representation (*field*), 994, 999–1001, 1022, 1023, 1025, 1026, 1063

reserved prefixes, **43**
BLUESKY_, 43
DECTRIS_, 43
identifier, 43
IDF_, 43
NDAttr, 43
NX, 43
NX_, 43
PDBX_, 43
SAS_, 43
SILX_, 43
reserved suffixes, **44**
end, 44
errors, 44
increment_set, 44
indices, 44
mask, 44
offset, 44
scaling_factor, 44
set, 44
weights, 44
reset_bias (*field*), 1130
residual (*field*), 397, 847
residuals (*field*), 799
resolution (*base class*), *see* NXresolution, **479**
resolution (*field*), 351, 480, 695, 700, 723, 755, 1044, 1047, 1163, 1164
resolution_errors (*field*), 351, 480
resolution_formula_description (*field*), 480
resolutions (*field attribute*), 606
resolutions_description (*field attribute*), 606
resonance_amplitude (*field*), 1133
resonance_frequency (*field*), 1133
response_time (*field*), 1164
results (*field*), 912
retardance (*field*), 532
revision (*field*), 384, 519, 584
revision history, **1216**
revolutions (*field*), 624
Riedel, Richard, 1212
roi_cylinder_height (*field*), 919
roi_cylinder_radius (*field*), 919
roi_id (*field*), 937
roi_orientation (*field*), 919
roi_process (*base class*), *see* NXroi_process, **481**
role (*field*), 529, 585, 632, 787, 794, 1098, 1107
roll (*field*), 612, 614
root (*base class*), *see* NXroot, **482**
rotating_element_type (*field*), 624
rotation, 28
rotation (*field*), 374, 378, 656
rotation_angle (*field*), 491, 691, 767, 770, 773, 776, 779, 781, 807, 810, 811, 822, 825, 837, 838, 842, 857
rotation_angle_step (*field*), 857
rotation_convention (*field*), 556, 634
rotation_euler (*field*), 485
rotation_handedness (*field*), 556, 634
rotation_quaternion (*field*), 485
rotation_speed (*field*), 339, 391, 530, 620
rotations (*base class*), *see* NXrotations, **484**
RPM, *see* NAPI installation
rules, 3
HDF, 19, 42, 143
naming, 24, 33, 41, 143
NeXus, 33
NX prefix, 24
XML, 143
run (*field*), 603
run_control (*field*), 499, 1088
run_cycle (*field*), 384, 519, 584
run_number (*field*), 551, 816, 1094, 1103

S

s (*field*), 1008
sacrifice_isotopic_uniqueness (*field*), 203, 572
sample (*base class*), *see* NXsample, **487**
sample_azimuth (*field*), 725
sample_component (*base class*), *see* NXsample_component, **494**
sample_component (*field*), 490
sample_id (*field*), 585, 756, 1182
sample_medium (*field*), 757
sample_medium_refractive_indices (*field*), 758
sample_mode (*field*), 1182
sample_name (*field*), 1182
sample_normal_polar_angle_of_tilt (*field*), 853
sample_normal_tilt_azimuth_angle (*field*), 853
sample_orientation (*field*), 489, 495, 756
sample_polar (*field*), 725
sample_rotation_angle (*field*), 852
sample_symmetry (*field*), 485
sample_tilt (*field*), 725
sample_x (*field*), 808
sample_y (*field*), 808
sampled_fraction (*field*), 440
sas (*application definition*), *see* NXsas, **771**
sastof (*application definition*), *see* NXsastof, **775**
saturation_value (*field*), 328, 334, 736
save_to_file (*field*), 1040
scale (*field*), 1084
scaling (*field attribute*), 828
scaling_factor (*field attribute*), 428, 429, 608
scaling_factor (*field*), 238, 317
scan (*application definition*), *see* NXscan, **778**
scan_angle_x (*field*), 1119
scan_angle_y (*field*), 1119
scan_angleN (*field*), 1149

scan_axis (*field*), 1179
scan_control_type (*field*), 1144
scan_controller (*base class*), *see* NXscan_controller, **496**
scan_end_bias (*field*), 1122, 1130
scan_end_x (*field*), 1119
scan_end_y (*field*), 1119
scan_endN (*field*), 1149
scan_mode (*field*), 872, 1116, 1151, 1154, 1180
scan_number (*field*), 1167
scan_offset_bias (*field*), 1122, 1129
scan_offset_value_x (*field*), 1119
scan_offset_value_y (*field*), 1119
scan_offset_valueN (*field*), 1148
scan_point_positions (*field*), 360
scan_points_bias (*field*), 1122, 1130
scan_points_x (*field*), 1119
scan_points_y (*field*), 1119
scan_pointsN (*field*), 1144, 1146
scan_range_bias (*field*), 1122, 1129
scan_range_x (*field*), 1118
scan_range_y (*field*), 1118
scan_rangeN (*field*), 1148
scan_resolutionN (*field*), 1144
scan_schema (*field*), 497, 655, 656
scan_speedN (*field*), 1146
scan_start_bias (*field*), 1122, 1130
scan_start_x (*field*), 1119
scan_start_y (*field*), 1119
scan_startN (*field*), 1149
scan_time (*field*), 1130
scan_time_end (*field*), 1143
scan_time_start (*field*), 1143
scan_type (*field*), 1116, 1129, 1144
scanTAG (*field*), 1118
scattering_angle (*field*), 517
scattering_configuration (*field*), 765
scattering_cross_section (*field*), 224
scattering_energy (*field*), 517
scattering_length_density (*field*), 490, 495
scattering_target (*field*), 517
scattering_vector (*field*), 288
scheme (*field*), 273, 380, 700, 701, 724
Scientific Data Sets, *see* field
SDD (*field*), 611
SDS (*Scientific Data Sets*), *see* field
seblock (*field*), 781
second_order_correctionN (*field*), 1137
secondary_horizontal_detector_width (*field*), 791
secondary_horizontal_distance (*field*), 791
secondary_horizontal_evaluation (*field*), 791
secondary_horizontal_sample_width (*field*), 791
secondary_horizontal_type (*field*), 791
seed (*field*), 294, 949
segment_columns (*field*), 288
segment_gap (*field*), 288
segment_height (*field*), 288
segment_rows (*field*), 288
segment_thickness (*field*), 288
segment_width (*field*), 288
semi_axes_value (*field*), 244
semi_axes_values (*field*), 244
semi_convergence_angle (*field*), 377
sensitivity (*field*), 1133
sensitivity_factor (*field*), 1015
sensor (*base class*), *see* NXsensor, **498**
sensor_material (*field*), 328, 736
sensor_scan (*application definition*), *see* NXsensor_scan, **1085**
sensor_size (*field*), 589
sensor_thickness (*field*), 328, 736
separator (*base class*), *see* NXseparator, **1090**
sequence_index (*field*), 444, 466, 562, 563, 565–567, 569–571, 883–886, 1007, 1008
sequence_number (*field*), 324, 825
serial_number (*field*), 323, 390, 558–560, 640–645
set_Bfield_current (*field*), 1090, 1114
set_current (*field*), 1004, 1017, 1080, 1111
set_Efield_voltage (*field*), 1090, 1114
set_identifier (*field*), 902, 903
set_point (*field*), 1141
set_voltage (*field*), 1004, 1017
setpoint (*field*), 430–432, 461, 705, 706, 754
setting (*field*), 652, 656
settling_time (*field*), 1129
sgl (*field*), 811
sgu (*field*), 811
ShadowFactor (*field*), 610
shank_angle (*field*), 217, 556, 568
shape (*base class*), *see* NXshape, **501**
shape (*field*), 200, 252, 257, 262, 501, 611, 724, 772, 776, 979, 1073, 1096, 1097, 1105, 1106
shear_modulus (*field*), 1049
shermann_function (*field*), 517
short_name (*field attribute*), 349, 425, 504, 732, 741, 787
short_name (*field*), 198, 387, 498
short_title (*field*), 491
shortest_half_life (*field*), 202, 572
show_coordinates (*field*), 1039
show_micron_bar (*field*), 1039
sigma_x (*field*), 505
sigma_y (*field*), 505
sign_convention (*field*), 557, 635
signal (*field attribute*), 316, 688, 834
signal (*file attribute*), 314
signal (*group attribute*), 329, 352, 407, 561, 568, 569, 592, 597, 603, 616, 645–652, 659, 660, 702,

708, 712, 726, 749, 758, 773, 798, 884, 885, 987, 988, 1008, 1035, 1053, 1062
signal attribute value, **50**, 314
signal data, 20, 50
signal_amplitude (*field*), 207
signal_over_noise (*field*), 876
signal_type (*field*), 270
signed_distance (*field*), 937
silx (*data analysis software*), 1207
similarity_grouping (*base class*), *see* NXsimilarity_grouping, **1091**
situation (*field*), 489, 586, 709, 724
size (*field*), 201, 231, 502, 772, 776, 933, 979, 1073, 1096, 1097, 1105, 1106
slit (*base class*), *see* NXslit, **502**
slit (*field*), 391
slit_angle (*field*), 339
slit_edges (*field*), 339
slit_height (*field*), 339
slit_length (*field*), 612
slits (*field*), 339
slot (*field*), 324
slow_axes (*field*), 350, 700
slow_pixel_direction (*field*), 337, 738
SNSbanking_file_name (*field*), 1095, 1104
SNSdetector_calibration_id (*field*), 1095, 1104
snsevent (*application definition*), *see* NXsnsevent, **1093**
SNSgeometry_file_name (*field*), 1095, 1104
snshisto (*application definition*), *see* NXsnshisto, **1102**
SNSmapping_file_name (*field*), 1095, 1104
SNStranslation_service (*field*), 1095, 1104
soft_limit_max (*field*), 465
soft_limit_min (*field*), 465
software, 1205, 1206
solenoid_magnet (*base class*), *see* NXsolenoid_magnet, **1111**
solid_angle (*field*), 323
solid_angle_applied (*field*), 591, 596
solid_geometry (*base class*), *see* NXsolid_geometry, **1112**
soller_angle (*field*), 275
source (*base class*), *see* NXsource, **503**
source distribution, *see* NAPI installation
source_distance_x (*field*), 235
source_distance_y (*field*), 235
source_peak_wavelength (*field*), 1179
space_group (*field*), 287, 490, 495, 527, 659, 799
spatial0 (*field*), 704, 713
spatial1 (*field*), 704, 713
spatial_acceptance (*field*), 274, 700
spatial_distribution (*field*), 1050
spatial_filter (*base class*), *see* NXspatial_filter, **1113**
spe (*application definition*), *see* NXspe, **780**
spec2nexus, 1208
special_enthalpy (*field*), 1052
special_pre_factor (*field*), 1052
specific_polarization_filter_type (*field*), 752
specimen_symmetry_point_group (*field*), 1043, 1046
spectral_range (*field*), 1070
spectrum (*base class*), *see* NXspectrum, **509**
spectrum (*field*), 1164
Sphinx (*documentation generator*), 1215
spin_rotator (*base class*), *see* NXspin_rotator, **1114**
spindispersion (*base class*), *see* NXspindispersion, **516**
spiral_radiusN (*field*), 1144, 1147
splitting_ratio (*field*), 978
spm (*application definition*), *see* NXspm, **1115**
spm_bias_spectroscopy (*base class*), *see* NXspm_bias_spectroscopy, **1127**
spm_cantilever (*base class*), *see* NXspm_cantilever, **1131**
spm_cantilever_config (*base class*), *see* NXspm_cantilever_config, **1132**
spm_cantilever_oscillator (*base class*), *see* NXspm_cantilever_oscillator, **1134**
spm_piezo_config (*base class*), *see* NXspm_piezo_config, **1135**
spm_piezo_sensor (*base class*), *see* NXspm_piezo_sensor, **1137**
spm_piezoelectric_material (*base class*), *see* NXspm_piezoelectric_material, **1139**
spm_positioner (*base class*), *see* NXspm_positioner, **1140**
spm_scan_control (*base class*), *see* NXspm_scan_control, **1143**
spm_scan_pattern (*base class*), *see* NXspm_scan_pattern, **1145**
spm_scan_region (*base class*), *see* NXspm_scan_region, **1148**
spm_temperature_sensor (*base class*), *see* NXspm_temperature_sensor, **1149**
spot_position (*field*), 209
spring_constant (*field*), 1133
spwidth (*field*), 530
sqom (*application definition*), *see* NXsqom, **782**
stage_type (*field*), 753
standard (*field*), 750
standing_voltage (*field*), 208, 560
start (*field attribute*), 324, 339, 389, 428, 429
start (*field*), 1084, 1180
start_time (*field*), 197, 204, 297, 298, 324, 369, 384, 439, 519, 551, 560, 584, 588, 620, 630, 645, 683, 685, 691, 693, 709, 730, 745, 767, 769, 772, 775, 779, 786, 806, 810, 813, 816, 819, 821, 824, 829, 833, 882, 966, 970, 976, 986, 1006, 1007, 1035, 1048, 1060, 1086, 1094, 1103, 1162, 1167

state (*field*), 935
static_q_list (*field*), 1174
static_roi_map (*field*), 1174
status (*field*), 212, 224, 231, 359, 393, 558, 966, 969, 975
status_indicators (*field*), 270
step (*field*), 1180
step_size_bias (*field*), 1122, 1130
step_size_x (*field*), 1119
step_size_y (*field*), 1119
step_sizeN (*field*), 1147
steppingN (*field*), 1146
stepsize (*field*), 1050
stm (*application definition*), *see* NXstm, **1151**
stop_on_symmetry_mismatch (*field*), 1040
stop_time (*field*), 324
storage_mode (*field attribute*), 1168–1171
stored_energy (*field*), 1050
strategies, 74
    simplest case(s), 74
stress (*application definition*), *see* NXstress, **784**
stress_field (*field*), 488, 586, 792
stride (*field*), 1084
strings, **48**
    arrays, 48
    fixed-length, 48
    variable-length, 48
sts (*application definition*), *see* NXsts, **1154**
stxm (*application definition*), *see* NXstxm, **806**
stxm_scan_type (*field*), 807
subentry
    NXsubentry, 175
subentry (*base class*), *see* NXsubentry, **517**
subsampling_filter (*base class*), *see* NXsubsampling_filter, **1158**
substance (*base class*), *see* NXsubstance, **1159**
substrate (*field*), 757
substrate_density (*field*), 404, 435
substrate_material (*field*), 394, 404, 407, 435, 450, 532, 980, 1074
substrate_roughness (*field*), 395, 405, 407, 435
substrate_thickness (*field*), 395, 404, 407, 435, 450, 532, 980, 1074
support_membrane_material (*field*), 402
support_membrane_thickness (*field*), 402
surface (*field*), 408
surface_energy (*field*), 1025, 1052
surface_indices (*group attribute*), 407
surface_reconstruction (*field*), 248
surface_resistivity (*field*), 1140
surface_type (*field*), 1079
switch_off_delay (*field*), 1142
sx (*field*), 801
sy (*field*), 801
symbol (*field attribute*), 237
symbols (*NXDL element*), **150**
symbolsType (*NXDL data type*), **167**
symmetric (*field*), 534
symmetry (*field*), 248, 341, 883, 925, 1035, 1061
sz (*field*), 801

T

T (*field*), 616
T_axes (*group attribute*), 616
table (*field*), 530
tags, 141, 1203
taper (*field*), 423
target, attribute, **21**
target_amplitude (*field*), 1135
target_dcom_radius (*field*), 917
target_detection_rate (*field*), 210, 561
target_edge_length (*field*), 917
target_frequency (*field*), 1135
target_material (*field*), 505
target_smoothing_step (*field*), 917
target_temperature (*field*), 1150
target_value (*field*), 465
targets (*field*), 894
tas (*application definition*), *see* NXtas, **809**
Tdev (*field*), 617
telephone_number (*field*), 529, 632, 1087
temp_offset_value (*field*), 1150
temperature (*field*), 289, 393, 437, 488, 586, 589, 614, 704, 714, 731, 781, 792, 835, 1012, 1054, 1062, 1063, 1097, 1106, 1123, 1150, 1174
temperature_coefficient (*field*), 289
temperature_indices (*group attribute*), 1053, 1062
temperature_nominal (*field*), 757
temperature_range (*field*), 270, 1140
temperature_set (*field*), 1174
TEMPERATUREchannel (*field*), 1150
template, *see* NXDL template file
tensor (*field*), 1041, 1062, 1063
term (*field*), 615
text (*field*), 998
text_interpreter (*field*), 1040
texture_index (*field*), 1044
theta (*field*), 1044, 1053
thickness (*field*), 224, 288, 393, 401, 452, 491, 614, 634, 756, 1074
thickness_determination (*field*), 756
thread_id (*field*), 962, 1063, 1065
threshold_distance (*field*), 897
threshold_energy (*field*), 328, 333, 736
threshold_proximity (*field*), 902
tiled
    tools, 1208
tilt1 (*field*), 373, 656

tilt2 (*field*), 374, 656
tilt_angle (*field*), 282, 842
tilt_correction (*field*), 378
tiltN (*field*), 1136
time (*field attribute*), 224, 506
time (*field*), 256, 357, 428, 1008, 1021, 1053, 1062, 1063, 1094, 1095, 1103
time_indices (*group attribute*), 1053, 1062
time_of_flight (*field*), 321, 439, 689, 769, 770, 776, 777, 814, 816, 817, 819, 820, 1098, 1105, 1107
time_offset (*field*), 256
time_origin_location (*field attribute*), 1170
time_per_channel (*field*), 589, 733
time_points (*field*), 1163
time_zone (*field*), 732
timestamp (*field*), 256
timestamp (*group attribute*), 604, 616
timing (*field*), 1004, 1017
tip_lift (*field*), 1141
tip_radius (*field*), 217
tip_radius_zero (*field*), 218
Tischler, Jonathan, 12, 1213
title (*field*), 318, 362, 383, 507, 518, 568, 569, 584, 588, 603, 620, 622, 645–652, 659, 660, 683, 685, 686, 691, 693, 730, 745, 765, 767, 769, 772, 775, 779, 782, 786, 794, 796, 806, 810, 813, 816, 819, 821, 824, 827, 829, 832, 833, 884, 885, 986, 1008, 1036, 1053, 1062, 1094, 1103
tof_distance (*field*), 381
tof_zero_estimate (*field*), 565
tofnpd (*application definition*), *see* NXtofnpd, **813**
tofraw (*application definition*), *see* NXtofraw, **815**
tofsingle (*application definition*), *see* NXtofsingle, **818**
tolerance (*field*), 465
tomo (*application definition*), *see* NXtomo, **821**
tomophase (*application definition*), *see* NXtomophase, **824**
tomoproc (*application definition*), *see* NXtomoproc, **827**
top_dead_center (*field*), 339
top_up (*field*), 506
total (*field*), 268, 934, 1092
total_area (*field*), 397, 459, 847
total_counts (*field*), 1094, 1096, 1103, 1105
total_elapsed_time (*field*), 297, 882, 966, 970, 976, 1006
total_event_golden (*field*), 563
total_event_incomplete (*field*), 563
total_event_multiple (*field*), 563
total_event_partials (*field*), 564
total_event_record (*field*), 564
total_flux (*field*), 740
total_flux_integrated (*field*), 740
total_spectroscopy_time (*field*), 1129
total_surface_area (*field*), 243
total_uncounted_counts (*field*), 1094, 1103
trajectory_points (*field*), 1147
TRANSFER_MATRIX (*field*), 233
transfer_rate (*field*), 1070
transform (*field attribute*), 828
transformation matrices, 27
transformation type (*field attribute*), 28
transformation_type (*field attribute*), 336, 337, 523, 723–726, 738, 751, 845–847, 852, 853
transformations (*base class*), *see* NXtransformations, **521**
transitions (*field*), 694, 844
translation, 28
translation (*base class*), *see* NXtranslation, **525**
translation (*field*), 987
transmission (*application definition*), *see* NXtransmission, **1161**
transmission (*field*), 450, 614, 979, 1073
transmittance (*field*), 758
transmitting_material (*field*), 275, 392
tree structure, *see* hierarchy
trigger_dead_time (*field*), 327
trigger_delay_time (*field*), 327
trigger_delay_time_set (*field*), 327
trigger_internal_delay_time (*field*), 327
trunc_species (*field*), 879
tutorial
    WONI, 62
twist (*field*), 530
two_theta (*field*), 1180
two_time_corr_func (*field*), 1169
twotheta (*field*), 747
type, *see* data type
type (*field attribute*), 445, 553, 631, 633, 702, 712, 1160
type (*field*), 199, 206, 222, 224, 239, 241, 275, 280, 286, 293–295, 300, 319, 324, 339, 344, 345, 347, 369, 372, 387, 391, 400, 423, 430, 434, 437, 439, 443, 449, 464, 480, 489, 499, 504, 517, 530, 532, 553, 558, 563–569, 585, 588, 592, 596, 623, 631, 635–637, 639, 640, 642, 644, 645, 649, 657, 658, 683, 686, 691, 695–697, 700, 705, 706, 723, 737, 749, 751, 754, 755, 767, 772, 776, 783, 788, 806, 822, 824, 827, 829, 834, 949, 956, 976, 978, 986, 1060, 1062, 1063, 1069, 1072, 1074, 1076, 1081, 1096, 1097, 1104, 1106, 1139, 1164
type (*group attribute*), 385
type (*NXDL attribute*), 69, 70

U

ub_matrix (*field*), 489
UDunits, 49
unassigned (*field*), 1092
uncertainties (*field attribute*), 605, 608, 616

underload_value (*field*), 328, 334, 736
Unidata UDunits, 49
unit category, **171**
unit_cell (*base class*), *see* NXunit_cell, **526**
unit_cell (*field*), 287, 488, 689, 811, 834
unit_cell_a (*field*), 287, 394
unit_cell_abc (*field*), 488, 495
unit_cell_alpha (*field*), 287, 394
unit_cell_alphabetagamma (*field*), 488, 495
unit_cell_b (*field*), 287, 394
unit_cell_beta (*field*), 287, 394
unit_cell_c (*field*), 287, 394
unit_cell_class (*field*), 490, 495
unit_cell_gamma (*field*), 287, 394
unit_cell_volume (*field*), 287, 394, 489, 495
units, 16, 20, **49**, 144
units (*field attribute*), 316, 455, 592, 593, 597, 598, 605, 607–610, 625, 795–798, 800, 987, 1070, 1177, 1179
units (*NXDL attribute*), 70
upper_cap_radii (*field*), 243
upper_cap_surface_area (*field*), 243
upper_cutoff_frequency (*field*), 877
URL (*field attribute*), 384, 519, 622, 745, 754, 764, 998, 1162
url (*field attribute*), 468, 1035
url (*field*), 272
URL (*group attribute*), 626
url (*group attribute*), 1162
usage (*field*), 286
use (*field*), 879
use of, 175
use_these (*field attribute*), 1024–1028
user (*base class*), *see* NXuser, **528**
UTF-8, **48**, 170
utilities, 1205
uuid (*field*), 291

V

validation, **78**, 1206
validItemName (*NXDL data type*), **168**
validNXClassName (*NXDL data type*), **169**
validTargetName (*NXDL data type*), **169**
value (*field*), 209, 210, 387, 428, 430–432, 454, 462, 465, 499, 561, 705, 706, 710, 711, 754, 996, 999–1001, 1005, 1017, 1080, 1088, 1090, 1094–1097, 1103, 1105, 1106, 1111, 1112, 1114, 1115
value (*NXDL attribute*), 70
value (*transformation matrix*), **28**
value_deriv1 (*field*), 499
value_deriv2 (*field*), 499
value_timestamp (*field*), 1088
value_x (*field*), 344, 655
value_y (*field*), 344, 655
values (*field*), 994, 995, 999–1001
variable (*field*), 687
varied_parameter_link (*field*), 626
varied_variable (*field attribute*), 687
varphi_one (*field*), 1045
varphi_two (*field*), 1045
vary (*field attribute*), 455
vector (*field attribute*), 28, 229, 336, 337, 523, 723–726, 738, 751, 845–847, 852, 853
vector (*field*), 988
vector (*group attribute*), 988
velocity (*field*), 465
velocity_selector (*base class*), *see* NXvelocity_selector, **529**
vendor (*field*), 390, 558–560, 640–645, 696–698, 700–702, 706, 754, 1117
verification, **78**
version (*field attribute*), 291, 383, 384, 390, 467, 519, 551–553, 563–571, 585, 622, 631, 640, 657, 693, 722, 744, 754, 759, 764, 780, 878, 882, 968, 969, 975, 976, 998, 1006, 1035, 1049, 1060, 1086, 1087, 1162
version (*field*), 466, 592, 597, 626, 686, 783, 827, 832, 1095, 1104
version (*group attribute*), 602, 730
vertex_incident_half_edge (*field*), 250
vertex_index_offset (*field*), 926, 929
vertex_normal (*field*), 936
vertex_normal_orientation (*field*), 936
vertex_normals (*field*), 898, 920
vertices (*field*), 246, 260, 301, 448, 898, 910, 918–920, 927, 929, 933, 934, 936, 957, 963, 968
vertices_are_unique (*field*), 246, 260
virtual_pixel_interpolation_applied (*field*), 327, 735
vocabulary, **175**
voltage (*field*), 207, 319, 343, 344, 347, 411, 505, 560, 652, 653, 655, 1012, 1120
voltage_filter_final (*field*), 216, 568
voltage_filter_initial (*field*), 216, 568
voltage_gain (*field*), 1121
voltage_range (*field*), 350
voltage_zero (*field*), 218
volume (*field*), 216, 222, 253, 262, 527, 567, 933, 957, 962, 1023, 1026, 1036, 1064
volume_fraction (*field*), 490, 495, 1044
voxel (*field*), 886
vscode-h5web
    tools, 1208

W

warmup (*field*), 294, 949

wavelength (*field*), 209, 288, 329, 392, 408, 441, 506, 530, 561, 591, 596, 683, 691, 766, 767, 772, 789, 834, 840, 997, 999, 1000, 1097, 1106, 1163
wavelength_dispersion (*field*), 442
wavelength_error (*field*), 441
wavelength_errors (*field*), 441
wavelength_identifier (*field*), 993, 999–1001
wavelength_indices (*group attribute*), 329, 408
wavelength_max (*field*), 614, 994
wavelength_min (*field*), 613, 994
wavelength_range (*field*), 340, 978, 1073, 1164
wavelength_range_coating (*field*), 533, 980
wavelength_spread (*field*), 530, 772
wavelength_unit (*field*), 994, 999–1001
wavelengths (*field*), 532
waveplate (*base class*), *see* NXwaveplate, **531**
webpage, 1197
wedge_angle (*field*), 980, 1074
weight (*field*), 366, 883, 884, 895, 992, 1044
weighting_method (*field*), 991
weinberg_vector (*field*), 251
why NeXus?, *see* motivation, 12
width (*field*), 253, 262, 392, 530, 850
winding_order (*field*), 246, 448
window_effects_corrected (*field*), 452
window_effects_type (*field*), 452
windowing_method (*field*), 968, 1113
Windows, *see* NAPI installation
WONI, 62
work_function (*field*), 349, 700, 846
working_distance (*field*), 240, 274, 377
write file, 14

X

x (*field*), 231, 280, 318, 558, 635, 636, 638, 639, 657, 828, 844, 976, 1036, 1062, 1063, 1121, 1129, 1138
x_alias (*field*), 280, 976, 1061
x_axis_direction (*field*), 1038
x_boundary_convention (*field*), 357, 658
x_direction (*field*), 280, 355, 558, 635, 636, 638, 639, 658, 1061
x_gap (*field*), 503, 611
x_indices (*field*), 988
x_indices (*group attribute*), 1062
x_normalization_direction (*field*), 357, 658
x_pixel_offset (*field*), 322, 1096, 1097, 1105, 1106
x_pixel_size (*field*), 323, 612, 689, 769, 773, 776, 822, 825, 834, 987, 1173
x_position (*field*), 612, 614
x_rotation_axis_pixel_position (*field*), 822
x_set (*field*), 1054, 1063
x_translation (*field*), 491, 822, 825, 835
xas (*application definition*), *see* NXxas, **829**
xasproc (*application definition*), *see* NXxasproc, **831**
XAXIS (*field*), 798
xbase (*application definition*), *see* NXxbase, **833**
xdmf_cell_id (*field*), 962
xdmf_gradient (*field*), 928
xdmf_indices_feature (*field*), 934
xdmf_intensity (*field*), 928
xdmf_topology (*field*), 927, 928, 930, 933, 934, 936, 957, 962
xdmf_xyz (*field*), 928
xeuler (*application definition*), *see* NXxeuler, **836**
xkappa (*application definition*), *see* NXxkappa, **838**
xlaue (*application definition*), *see* NXxlaue, **839**
xlaueplate (*application definition*), *see* NXxlaueplate, **840**
xmax (*field*), 218
xmin (*field*), 218
XML, 1212
XML_version (*file attribute*), 483
xmlns (*NXDL attribute*), 69
xnb (*application definition*), *see* NXxnb, **841**
xpcs (*application definition*), *see* NXxpcs, **1166**
xps (*application definition*), *see* NXxps, **843**
xray_tube_current (*field*), 1179
xray_tube_material (*field*), 1179
xray_tube_voltage (*field*), 1179
xraylens (*base class*), *see* NXxraylens, **534**
xrd (*application definition*), *see* NXxrd, **1176**
xrd_pan (*application definition*), *see* NXxrd_pan, **1178**
xrot (*application definition*), *see* NXxrot, **856**
xsi:schemaLocation (*NXDL attribute*), 69

Y

y (*field*), 231, 280, 318, 558, 635, 636, 638, 640, 658, 828, 844, 976, 1036, 1121, 1129, 1138
y_alias (*field*), 280, 976, 1061
y_boundary_convention (*field*), 357, 658
y_direction (*field*), 280, 356, 558, 635, 636, 638, 640, 658, 1061
y_gap (*field*), 503, 611, 1163
y_indices (*field*), 988
y_normalization_direction (*field*), 357, 658
y_pixel_offset (*field*), 322, 1096, 1105
y_pixel_size (*field*), 323, 612, 689, 769, 773, 776, 822, 825, 834, 987, 1173
y_position (*field*), 612, 614
y_pred (*field*), 885
y_rotation_axis_pixel_position (*field*), 822
y_translation (*field*), 822, 825, 835
yaw (*field*), 612, 615
ymax (*field*), 218
ymin (*field*), 218
young_modulus (*field*), 1140

Z

z (*field*), 281, 318, 558, 635, 637, 638, 640, 658, 828, 844, 976, 1036, 1121, 1129, 1138, 1141
z_alias (*field*), 280, 976, 1061
z_average_time (*field*), 1142
z_axis_direction (*field*), 1038
z_controller_hold (*field*), 1142
z_controller_time (*field*), 1142
z_direction (*field*), 280, 356, 558, 635, 637, 638, 640, 658, 844, 1061
z_offset_value (*field*), 1142
z_pixel_offset (*field*), 322
z_translation (*field*), 822, 825
zmax (*field*), 219
zmin (*field*), 219
zone_height (*field*), 402
zone_material (*field*), 402
zone_support_material (*field*), 402