CHAPTER

TEN

NEXUS: INTERFACES REFERENCE DOCUMENTATION (EXPERIMENTAL)

Caution: EXPERIMENTAL This section is **experimental and is NOT an accepted part of the NeXus standard**. It shows how NeXus base classes can be improved by introducing NeXus interfaces.

10.1 Introduction

When defining NeXus base classes the NeXus community encountered two reoccurring problems:

1. Some fields, for example for positioning a component in the beam, are shared between many components.
2. Some base classes, for example NXdetector, describe a whole set of different pieces of equimpment and thus have become to complex.

The classical way to solve this problem would have been to use inheritance. But this would have complicated the NeXus hierarchy even further and causes problems with backwards compatability. Also there would have been cases of multiple inheritance where things turn really messy. Instead the NeXus community decided to solve the problem with interfaces. Interfaces detail the fields necessary to describe either a shared or special concept. A set of interfaces are defined which a base class can implement. The interfaces which a given base class can implements are stored as a group attribute with the name **implements** which becomes a komma separated list of interfaces. In a real file the **implements** attribute would hold the information which interfaces are actually implemented by this isntance of a group.

NeXus interfaces are up to now only used to improve the definition of base classes.

Information is stored in a NeXus data file by grouping together similar parts. For example, information about the sample could include a descriptive name, the temperature, and other items. NeXus specifies the contents of these groupings using *classes*. In some parts of this manual, these classes might be called *group type* or some similar term. In this section, the NeXus classes are described in detail. Each class is specified using *NXDL: The NeXus Definition Language*, described in a separate chapter.

10.2 WARNINGS

As this is experimental, some things are not as they should be:

- The base classes used here were derived 2-3 years ago. So, there may be things missing or in need of change.
- The application definitions are NOT given. They are not affected by interfaces. Though it may be an idea to extend the interfaces idea to application definitions to. Tobias features operate along these lines.

10.3 NeXus Class Specifications

10.3.1 Base Class Definitions

A description of each NeXus base class definition is given. NeXus base class definitions define the set of terms that *might* be used in an instance of that class. Consider the base classes as a set of *components* that are used to construct a data file.

NXaperture

Status:

base class, extends *NXobject*, version 1.0

Description:

Template of a beamline aperture.

Symbols:

No symbol table

Groups cited: *NXgeometry*, *NXnote*

Structure:

material: *NX_CHAR*

Absorbing material of the aperture

description: *NX_CHAR*

Description of aperture

(geometry): *NXgeometry*

location and shape of aperture

(geometry): *NXgeometry*

location and shape of each blade

(note): *NXnote*

describe an additional information in a note*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXaperture.nxdl.xml

NXattenuator

Status:

base class, extends *NXobject*, version 1.0

Description:

Description of 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: none

Structure:

@implements: *NX_CHAR*

NXattenuator can implement: NXIFbeamline_component, NXIFmetadata

thickness: *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of attenuator along beam direction

scattering_cross_section: *NX_FLOAT* {units=*NX_CROSS_SECTION*}

Scattering cross section (coherent+incoherent)

absorption_cross_section: *NX_FLOAT* {units=*NX_CROSS_SECTION*}

Absorption cross section

attenuator_transmission: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

The nominal amount of the beam that gets through (transmitted intensity)/(incident intensity)

status: *NX_CHAR*

In or out or moving of the beam

Any of these values: `in` | `out` | `moving`

@time: *NX_DATE_TIME*

time stamp for this observation

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXattenuator.nxdl.xml

NXbeam

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXdata*

Structure:

@implements: *NX_CHAR*

NXbeam can implement: NXIFbeamline_component in order to describe the position where the beam was characterized.

incident_energy[i]: *NX_FLOAT* {units=*NX_ENERGY*}

Energy on entering beamline component

final_energy[i]: *NX_FLOAT* {units=*NX_ENERGY*}

Energy on leaving beamline component

energy_transfer[i]: *NX_FLOAT* {units=*NX_ENERGY*}

Energy change caused by beamline component

incident_wavelength[i]: *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength on entering beamline component

incident_wavelength_spread[i]: *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength spread FWHM on entering component

incident_beam_divergence[2, j]: *NX_FLOAT* {units=*NX_ANGLE*}

Divergence of beam entering this component

final_wavelength[i]: *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength on leaving beamline component

incident_polarization[2, j]: *NX_FLOAT* {units=*NX_ANY*}

Polarization vector on entering beamline component

final_polarization[2, j]: *NX_FLOAT* {units=*NX_ANY*}

Polarization vector on leaving beamline component

final_wavelength_spread[i]: *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength spread FWHM of beam leaving this component

final_beam_divergence[2, j]: *NX_FLOAT* {units=*NX_ANGLE*}

Divergence FWHM of beam leaving this component

flux[i]: *NX_FLOAT* {units=*NX_FLUX*}

flux incident on beam plane area

(data): *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.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXbeam.nxdl.xml

NXbeam_stop

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXgeometry*

Structure:

@implements: *NX_CHAR*

NXbeam_stop can implement: NXIFbeamline_component, NXIFmetadata

size: *NX_FLOAT* {units=*NX_LENGTH*}

size of beamstop

status: *NX_CHAR*

Any of these values: `in | out`

(geometry): *NXgeometry*

engineering shape, orientation and position of the beam stop.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXbeam_stop.nxdl.xml

NXbending_magnet

Status:

base class, extends *NXobject*, version 1.0

Description:

description for a bending magnet

Symbols:

No symbol table

Groups cited: *NXdata*, *NXgeometry*

Structure:

@implements: *NX_CHAR*

NXbending_magnet can implement: NXIFbeamline_component, NXIFmetadata

critical_energy: *NX_FLOAT* {units=*NX_ENERGY*}

bending_radius: *NX_FLOAT* {units=*NX_LENGTH*}

magnetic_field: *NX_FLOAT* {units=*NX_CURRENT*}

strength of magnetic field of dipole magnets

accepted_photon_beam_divergence: *NX_FLOAT* {units=*NX_LENGTH*}

An array of four numbers giving X+, X-, Y+ and Y- half divergence

source_distance_x: *NX_FLOAT* {units=*NX_LENGTH*}

Distance of source point from particle beam waist in X (horizontal) direction.

source_distance_y: *NX_FLOAT* {units=*NX_LENGTH*}

Distance of source point from particle beam waist in Y (vertical) direction.

divergence_x_plus: *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: *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: *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: *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.

spectrum: *NXdata*

bending magnet spectrum

(geometry): *NXgeometry*

"Engineering" position of bending magnet

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXbending_magnet.nxdl.xml

NXcapillary

Status:

base class, extends *NXobject*, version 1.0

Description:

This is a dictionary of field names to use for describing a capillary as used in X-ray beamlines. Based on information provided by Gerd Wellenreuther.

Symbols:

No symbol table

Groups cited: *NXdata*

Structure:

@implements: *NX_CHAR*

NXcapillary can implement: NXIFbeamline_component

type: *NX_CHAR*

Type of the capillary

Any of these values:

- `single_bounce`

- `polycapillary`
- `conical_capillary`

manufacturer: *NX_CHAR*

The manufacturer of the capillary. This is actually important as it may have an impact on performance.

maximum_incident_angle: *NX_FLOAT* {units=*NX_ANGLE*}

accepting_aperture: *NX_FLOAT* {units=*NX_ANGLE*}

working_distance: *NX_FLOAT* {units=*NX_LENGTH*}

focal_size: *NX_FLOAT*

The focal size in FWHM

gain: *NXdata*

The gain of the capillary as a function of energy

transmission: *NXdata*

The transmission of the capillary as a function of energy

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXcapillary.nxdl.xml

NXcharacterization

Status:

base class, extends *NXobject*, version 1.0

Description:

note: This base class may be removed in future releases of NXDL. If you have a use for this base class, please provide a description of your intended use to the NIAC (nexus-committee@nexusformat.org).

Symbols:

No symbol table

Groups cited: none

Structure:

@source: *NX_CHAR*

If missing, the source file is the current file

@location: *NX_CHAR*

@mime_type: *NX_CHAR*

If missing, the source file is NAPI readable

definition: *NX_CHAR*

@version: *NX_CHAR*

@URL: *NX_CHAR*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXcharacterization.nxdl.xml

NXcite

Status:

base class, extends *NXobject*, version 1.0

Description:

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: *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: *NX_CHAR*

URL referencing the document or data.

doi: *NX_CHAR*

DOI referencing the document or data.

endnote: *NX_CHAR*

Bibliographic reference data in EndNote format.

bibtex: *NX_CHAR*

Bibliographic reference data in BibTeX format.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXcite.nxdl.xml

NXcollection

Status:

contributed definition, extends *NXobject*, version 1.0

Description:

Use `NXcollection` to gather together any set of terms. The original suggestion is to use this as a container class for the description of a beamline.

For NeXus validation, `NXcollection` will always generate a warning since it is always an optional group. Anything (groups, fields, or attributes) placed in an `NXcollection` group will not be validated.

Symbols:

No symbol table

Groups cited: none

Structure:

beamline: *NX_CHAR*

name of the beamline for this collection

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/contributed/NXcollection.nxdl.xml

NXcollimator

Status:

base class, extends *NXobject*, version 1.0

Description:

Template of a beamline collimator.

Symbols:

No symbol table

Groups cited: *NXlog*

Structure:

@implements: *NX_CHAR*

NXcollimator can implement: NXIFbeamline_component

type: *NX_CHAR*

Any of these values: `Soller`|`radial`|`oscillating`|`honeycomb`

soller_angle: *NX_FLOAT* {units=*NX_ANGLE*}

Angular divergence of Soller collimator

divergence_x: *NX_FLOAT* {units=*NX_ANGLE*}

divergence of collimator in local x direction

divergence_y: *NX_FLOAT* {units=*NX_ANGLE*}

divergence of collimator in local y direction

frequency: *NX_FLOAT* {units=*NX_FREQUENCY*}

Frequency of oscillating collimator

blade_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

blade thickness

blade_spacing: *NX_FLOAT* {units=*NX_LENGTH*}

blade spacing

absorbing_material: *NX_CHAR*

name of absorbing material

transmitting_material: *NX_CHAR*

name of transmitting material

frequency_log: *NXlog*

Log of frequency

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXcollimator.nxdl.xml

NXcrystal

Status:

base class, extends *NXobject*, version 1.0

Description:

Template of a crystal monochromator or analyzer. Permits double bent monochromator comprised of multiple segments with anisotropic Gaussian mosaic.

If curvatures are set to zero or are absent, array is considered to be flat.

Scattering vector is perpendicular to surface. Crystal is oriented parallel to beam incident on crystal before rotation, and lies in vertical plane.

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*, *NXlog*, *NXshape*

Structure:

@implements: *NX_CHAR*

NXcrystal can implement: NXIFbeamline_component, NXIFmonochromator, NXIFsingle-crystal

usage: *NX_CHAR*

How this crystal is used. Choices are in the list.

Any of these values:

- `Bragg`: reflection geometry
- `Laue`: transmission geometry, requires "thickness" field to be defined

type: *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 | Ge | Si | Cu | Fe3Si | CoFe | Cu2MnAl | Multilayer | Diamond

Note that "PG" is Highly Oriented Pyrolytic Graphite. Also, "Cu2MnAl" had the comment "Heusler".

order_no: *NX_INT*

A number which describes if this is the first, second,.. n^{th} crystal in a multi crystal monochromator

cut_angle: *NX_FLOAT* {units=*NX_ANGLE*}

Cut angle of reflecting Bragg plane and plane of crystal surface

d_spacing: *NX_FLOAT* {units=*NX_LENGTH*}

spacing between crystal planes of the reflection

scattering_vector: *NX_FLOAT* {units=*NX_WAVENUMBER*}

Scattering vector, Q, of nominal reflection

reflection[3]: *NX_INT* {units=*NX_UNITLESS*}

Miller indices (hkl) values of nominal reflection

thickness: *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the crystal. (Required for Laue orientations - see "usage" field)

density: *NX_NUMBER* {units=*NX_MASS_DENSITY*}

mass density of the crystal

segment_width: *NX_FLOAT* {units=*NX_LENGTH*}

Horizontal width of individual segment

segment_height: *NX_FLOAT* {units=*NX_LENGTH*}

Vertical height of individual segment

segment_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of individual segment

segment_gap: *NX_FLOAT* {units=*NX_LENGTH*}

Typical gap between adjacent segments

segment_columns: *NX_FLOAT* {units=*NX_LENGTH*}

number of segment columns in horizontal direction

segment_rows: *NX_FLOAT* {units=*NX_LENGTH*}

number of segment rows in vertical direction

mosaic_horizontal: *NX_FLOAT* {units=*NX_ANGLE*}

horizontal mosaic Full Width Half Maximum

mosaic_vertical: *NX_FLOAT* {units=*NX_ANGLE*}

vertical mosaic Full Width Half Maximum

curvature_horizontal: *NX_FLOAT* {units=*NX_ANGLE*}

Horizontal curvature of focusing crystal

curvature_vertical: *NX_FLOAT* {units=*NX_ANGLE*}

Vertical curvature of focusing crystal

is_cylindrical: *NX_BOOLEAN*

Is this crystal bent cylindrically?

cylindrical_orientation_angle: *NX_NUMBER* {units=*NX_ANGLE*}

If cylindrical: cylinder orientation angle

bragg_angle[i]: *NX_FLOAT* {units=*NX_ANGLE*}

Bragg angle of nominal reflection

temperature: *NX_FLOAT* {units=*NX_TEMPERATURE*}

average/nominal crystal temperature

temperature_coefficient: *NX_FLOAT* {units=*NX_ANY*}

how lattice parameter changes with temperature

temperature_log: *NXlog*

log file of crystal temperature

reflectivity: *NXdata*

crystal reflectivity versus wavelength

transmission: *NXdata*

crystal transmission versus wavelength

shape: *NXshape*

A NXshape group describing the shape of the crystal arrangement

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXcrystal.nxdl.xml

NXdata

Status:

base class, extends *NXobject*, version 1.0

Description:

(**required**) `NXdata` is a template of plottable data and their dimension scales. It is mandatory that there is at least one `NXdata` group in each NXentry group. Note that the `variable` and `data` can be defined with different names. The `signal` and `axes` attribute of the `data` item define which items are plottable data and which are *dimension scales*.

- Each `NXdata` group will consist of only one data set containing plottable data and their standard deviations.
- This data set may be of arbitrary rank up to a maximum of `NX_MAXRANK=32`.
- The plottable data will be identified by the attribute: `signal=1`
- The plottable data will identify the *dimension scale* specification(s) in the `axes` attribute.

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

- For each data dimension, there should be a one-dimensional array of the same length.
- These one-dimensional arrays are the *dimension scales* of the data, *i.e.* the values of the independent variables at which the data is measured, such as scattering angle or energy transfer.

There are two methods of linking each data dimension to its respective dimension scale.

The preferred (and recommended) method uses the `axes` attribute to specify the names of each *dimension scale*.

The older method uses the `axis` attribute on each *dimension scale* to identify with an integer the axis whose value is the number of the dimension.

`NXdata` is used to implement one of the basic motivations in NeXus, to provide a default plot for the data of this `NXentry`. The actual data might be stored in another group and (hard) linked to the `NXdata` group.

When constructing a NXdata group two cases have to be considered:

1. The NXdata describes raw data. In that case it is often (but not always) possible to use one of the detector interfaces matching your detector and use case as a guideline. Not the whole of the respective interface is needed, just the data field plus the associated axes need to be linked her. In the case of a scan, the variables varied during the scan need to be linked here too. In order to make NXdata similar to the tabular representation people are used to for scans.
2. The NXdata describes processed data. Then the content should still allow to plot the result of the processing. Thus it needs to contain the resulting data from the processing plus the axes required to plot the data.

Symbols:

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

dataRank: rank of the `data` field

n: length of the `variable` field

nx: length of the `x` field

ny: length of the `y` field

nz: length of the `z` field

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXdata can implement: NXIFsingle_detector, NXIFsingle_detector_scanned, NXIFsingle_detector_tof, NXIFlinear_detector, NXIFlinear_detector_scanned, NXIFlinear_detector_tof, NXIFarea_detector, NXIFarea_detector_scanned, NXIFarea_detector_tof, NXIFgeneric_detector, NXIFgeneric_detector_scanned, NXIFgeneric_detector_tof,

variable[n]: *NX_NUMBER*

Dimension scale defining an axis of the data. Client is responsible for defining the dimensions of the data. The name of this field may be changed to fit the circumstances. Standard NeXus client tools will use the attributes to determine how to use this field.

@long_name: *NX_CHAR*

Axis label

@distribution: *NX_BOOLEAN*

`0|false`: single value, `1|true`: multiple values

@first_good: *NX_INT*

Index of first good value

@last_good: *NX_INT*

Index of last good value

@axis: *NX_POSINT*

Index (positive integer) identifying this specific set of numbers. 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.

variable_errors[n]: *NX_NUMBER*

Errors (uncertainties) associated with axis `variable` Client is responsible for defining the dimensions of the data. The name of this field may be changed to fit the circumstances but is matched with the *variable* field with `_errors` appended.

data[n]: *NX_NUMBER*

This field contains the data values to be used as the NeXus *plottable data*. Client is responsible for defining the dimensions of the data. The name of this field may be changed to fit the circumstances. Standard NeXus client tools will use the attributes to determine how to use this field.

@signal: *NX_POSINT*

Plottable (independent) axis, indicate index number. Only one field in a `NXdata` group may have the `signal=1` attribute. Do not use the `signal` attribute with the `axis` attribute.

@axes: *NX_CHAR*

Defines the names of the dimension scales (independent axes) for this data set as a colon-delimited array. NOTE: The `axes` attribute is the preferred method of designating a link. Do not use the `axes` attribute with the `axis` attribute.

@uncertainties: *NX_CHAR*

Specify the names of the errors (uncertainties) of the dependent axes as plottable data. NOTE: The `errors` attribute uses the same syntax as the `axes` attribute.

@long_name: *NX_CHAR*

data label

errors[n]: *NX_NUMBER*

Standard deviations of data values - the data array is identified by the attribute `signal=1`. The `errors` array must have the same dimensions as `data`. Client is responsible for defining the dimensions of the data.

scaling_factor: *NX_FLOAT*

The elements in data are usually float values really. For efficiency reasons these are usually stored as integers after scaling with a scale factor. This value is the scale factor. It is required to get the actual physical value, when necessary.

offset: *NX_FLOAT*

An optional offset to apply to the values in data.

x[nx]: *NX_FLOAT* {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[ny]: *NX_FLOAT* {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[nz]: *NX_FLOAT* {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.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXdata.nxdl.xml

NXdetector

Status:

base class, extends *NXobject*, version 1.1

Description:

Template of a detector, detector bank, or multidetector.

Symbols:

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

np: number of scan points (only present in scanning measurements)

i: number of detector pixels in the first (X, slowest) direction

j: number of detector pixels in the second (Y, faster) direction

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

tof: number of bins in the time-of-flight histogram

Groups cited: *NXcharacterization*, *NXdetector_module*, *NXnote*

Structure:

@implements: *NX_CHAR*

NXdetector will of course only implement those interfaces out of the selection below which match the detector used and the mode in which the instrument was operated.

NXdetector can implement: NXIFbeamline_component, NXIFmetadata, NXIFphoton_counter, NXIFgasdetector, NXIFcount_time, NXIFcount_time_scanned, NXIFsingle_detector, NXIFsingle_detector_scanned, NXIFsingle_detector_tof, NXIFlinear_detector, NXIFlinear_detector_scanned, NXIFlinear_detector_tof, NXIFarea_detector, NXIFarea_detector_scanned, NXIFarea_detector_tof, NXIFgeneric_detector, NXIFgeneric_detector_scanned, NXIFgeneric_detector_tof,

detector_number[i, j]: *NX_INT*

Identifier for detector

solid_angle[i, j]: *NX_FLOAT* {units=*NX_SOLID_ANGLE*}

Solid angle subtended by the detector at the sample

crate[i, j]: *NX_INT*

Crate number of detector

@local_name: *NX_CHAR*

Equivalent local term

slot[i, j]: *NX_INT*

Slot number of detector

@local_name: *NX_CHAR*

Equivalent local term

input[i, j]: *NX_INT*

Input number of detector

@local_name: *NX_CHAR*

Equivalent local term

wavelength[i, j, k]: *NX_FLOAT* {units=*NX_WAVELENGTH*}

TODO: need documentation

calibration_date: *NX_DATE_TIME*

date of last calibration (geometry and/or efficiency) measurements

layout: *NX_CHAR*

How the detector is represented

Any of these values: `point` | `linear` | `area`

diameter: *NX_FLOAT* {units=*NX_LENGTH*}

The diameter of a cylindrical detector

acquisition_mode: *NX_CHAR*

The acquisition mode of the detector.

Any of these values:

- `gated`
- `triggered`
- `summed`
- `event`
- `histogrammed`

calibration_method: *NXnote*

summary of conversion of array data to pixels (e.g. polynomial approximations) and location of details of the calibrations

data_file: *NXnote*

(characterization): *NXcharacterization*

modulexx: *NXdetector_module*

In some cases detectors are built up from smaller modules. This group holds information about the modules a larger detector may be built up from.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXdetector.nxdl.xml

NXdetector_group

Status:

base class, extends *NXobject*, version 1.0

Description:

This class is used to allow a logical grouping of detector elements (e.g. which tube, bank or group of banks) to be recorded in the file. As well as allowing you to e.g just select the "left" or "east" detectors, it may also be useful for determining which elements belong to the same PSD tube and hence have e.g. the same dead time.

For example, if we had “bank1” composed of “tube1”, “tube2” and “tube3” then group_names would be the string “bank1, bank1/tube1, bank1/tube2,bank1/tube3” group_index would be {1,2,3,4} group_parent would be {-1,1,1,1}

The mapping array is interpreted as group 1 is a top level group containing groups 2, 3 and 4

A `group_index` array in `NXdetector` gives the base group for a detector element.

Symbols:

No symbol table

Groups cited: none

Structure:

group_names: *NX_CHAR*

Comma separated list of name

group_index[i]: *NX_INT*

Unique ID for group. A group_index array in `NXdetector` gives the base group for a detector element.

group_parent[ref(group_index)]: *NX_INT*

Index of group parent in the hierarchy: -1 means no parent (i.e. a top level) group

group_type[ref(group_index)]: *NX_INT*

Code number for group type, e.g. bank=1, tube=2 etc.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXdetector_group.nxdl.xml

NXdetector_module

Status:

base class, extends *NXobject*, version 1.0

Description:

This is the description of a detector module. Many detectors consist of multiple smaller modules. Soemtimes it is important to know the exact position of such modules. This is the prupose of this group. It is a child group to NXdetector.

Symbols:

No symbol table

Groups cited: none

Structure:

data_origin: *NX_NUMBER*

A two value field which gives the index of the start of the modules data in the main area detector image in the underlying NXdetector module.

data_size: *NX_NUMBER*

Two values for the size of the module in pixels in each direction.

module_offset: *NX_LENGTH*

Offset of the module in regards to the center of the detector in an arbitrary direction.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

fast_pixel_direction: *NX_LENGTH*

Values along the direction of fastest varying pixel direction.The direction itself is given through the vector attribute

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

slow_pixel_direction: *NX_LENGTH*

Values along the direction of fastest varying pixel direction. The direction itself is given through the vector attribute

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXdetector_module.nxdl.xml

NXdisk_chopper

Status:

base class, extends *NXobject*, version 1.0

Description:

A disk chopper is a speedily rotating disk made from an aborbing material placed into the beam. The rotation is around the beam direction. Its purpose is to cut the incoming beam into bunches. To this purpose the disk contains slits through which particles can pass when illuminated by the beam.

Disk choppers are mostly used in time-of flight neutron scattering.

At a constant flux neutron source like a reactor it cuts the incoming beam into bunches. Then the time the neutron needs to travel between the chopper and detector is recorded and used to determine the spead/wavelength of the neutron. This bunch generating chopper is often called the master chopper. At a pulsed neutron source, a master chopper is not necessary as the source already defines the pulse.

More complex chopper systems are used to further refine the raw neutron bunch by preventing frame overlap. by cutting the bunch up even further etc. These choppers are often called pulse shaping choppers.

Consider a bunch of neutrons travelling from the master chopper to a pulse shaping chopper further down the beamline. The time the bunch needs to travel to the pulse shaping chopper is not neglibile. Thus if the pulse shaping chopper would open in sync with the master chopper, the bunch would be absorbed. In order to prevent this, the pulse shaping chopper opens at an angular offset in relation to the master chopper. This angular offset is the chopper phase.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXattenuator can implement: NXIFbeamline_component

type: *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: *NX_FLOAT* {units=*NX_FREQUENCY*}

chopper rotation speed

slits: *NX_INT*

Number of slits

slit_angle: *NX_FLOAT* {units=*NX_ANGLE*}

angular opening

pair_separation: *NX_FLOAT* {units=*NX_LENGTH*}

disc spacing in direction of beam

radius: *NX_FLOAT* {units=*NX_LENGTH*}

radius to centre of slit

slit_height: *NX_FLOAT* {units=*NX_LENGTH*}

total slit height

phase: *NX_FLOAT* {units=*NX_ANGLE*}

chopper phase angle

ratio: *NX_INT*

pulse reduction factor of this chopper in relation to other choppers/fastest pulse in the instrument

wavelength_range[2]: *NX_FLOAT* {units=*NX_WAVELENGTH*}

low and high values of wavelength range transmitted

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXdisk_chopper.nxdl.xml

NXentry

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXcharacterization*, *NXcollection*, *NXdata*, *NXinstrument*, *NXmonitor*, *NXnote*, *NXparameters*, *NXprocess*, *NXsample*, *NXsubentry*, *NXuser*

Structure:

@IDF_Version: *NX_CHAR*

ISIS Muon IDF_Version

title: *NX_CHAR*

Extended title for entry

experiment_identifier: *NX_CHAR*

Unique identifier for the experiment, defined by the facility, possibly linked to the proposals

experiment_description: *NX_CHAR*

Brief summary of the experiment, including key objectives.

collection_identifier: *NX_CHAR*

User or Data Acquisition defined group of NeXus files or NXentry

collection_description: *NX_CHAR*

Brief summary of the collection, including grouping criteria.

entry_identifier: *NX_CHAR*

unique identifier for the measurement, defined by the facility.

definition: *NX_CHAR*

(alternate use: see same field in *NXsubentry* for preferred)

Official NeXus NXDL schema to which this file conforms.

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: *NX_CHAR*

NXDL version number

@URL: *NX_CHAR*

URL of NXDL file

definition_local: *NX_CHAR*

(deprecated use: see same field in NXsubentry for preferred) Local NXDL schema extended from the file specified in the `definition` field. This contains any locally-defined, additional fields in the file.

@version: *NX_CHAR*

NXDL version number

@URL: *NX_CHAR*

URL of NXDL file

start_time: *NX_DATE_TIME*

Starting time of measurement

end_time: *NX_DATE_TIME*

Ending time of measurement

duration: *NX_INT* {units=*NX_TIME*}

Duration of measurement

collection_time: *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: *NX_CHAR*

Such as "2007-3". Some user facilities organize their beam time into run cycles.

program_name: *NX_CHAR*

Name of program used to generate this file

@version: *NX_CHAR*

Program version number

@configuration: *NX_CHAR*

configuration of the program

revision: *NX_CHAR*

Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument definition format, ...

@comment: *NX_CHAR*

pre_sample_flightpath: *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: *NXnote*

Description of the full experiment (document in pdf, latex, ...)

notes: *NXnote*

Notes describing entry

thumbnail: *NXnote*

A small image that is representative of the entry. An example of this is a 640x480 jpeg image automatically produced by a low resolution plot of the NXdata.

@mime_type: *NX_CHAR*

The value should be an `image/*`

Obligatory value: `image/*`

(characterization): *NXcharacterization*

(user): *NXuser*

(sample): *NXsample*

(instrument): *NXinstrument*

(collection): *NXcollection*

(monitor): *NXmonitor*

(data): *NXdata*

(parameters): *NXparameters*

(process): *NXprocess*

(subentry): *NXsubentry*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXentry.nxdl.xml

NXenvironment

Status:

base class, extends *NXobject*, version 1.0

Description:

This class describes an external condition applied to the sample

Symbols:

No symbol table

Groups cited: *NXgeometry*, *NXnote*, *NXsensor*

Structure:

name: *NX_CHAR*

Apparatus identification code/model number; e.g. OC100 011

short_name: *NX_CHAR*

Alternative short name, perhaps for dashboard display like a present Seblock name

type: *NX_CHAR*

Type of apparatus. This could be the SE codes in scheduling database; e.g. OC/100

description: *NX_CHAR*

Description of the apparatus; e.g. 100mm bore orange cryostat with Roots pump

program: *NX_CHAR*

Program controlling the apparatus; e.g. LabView VI name

position: *NXgeometry*

The position and orientation of the apparatus

(note): *NXnote*

Additional information, LabView logs, digital photographs, etc

(sensor): *NXsensor*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXenvironment.nxdl.xml

NXevent_data

Status:

base class, extends *NXobject*, version 1.0

Description:

Time-of-flight events

Symbols:

No symbol table

Groups cited: none

Structure:

time_of_flight[i]: *NX_INT* {units=*NX_TIME_OF_FLIGHT*}

A list of time of flight for each event as it comes in. This list is for all pulses with information to attach to a particular pulse located in events_per_pulse.

pixel_number[i]: *NX_INT* {units=*NX_DIMENSIONLESS*}

There will be extra information in the NXdetector to convert pixel_number to detector_number. This list is for all pulses with information to attach to a particular pulse located in events_per_pulse.

pulse_time[j]: *NX_INT* {units=*NX_TIME*}

The time that each pulse started with respect to the offset

@offset: *NX_DATE_TIME*

ISO8601

events_per_pulse[j]: *NX_INT* {units=*NX_DIMENSIONLESS*}

This connects the index "i" to the index "j". The jth element is the number of events in "i" that occurred during the jth pulse.

pulse_height[i, k]: *NX_FLOAT* {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.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXevent_data.nxdl.xml

NXfermi_chopper

Status:

base class, extends *NXobject*, version 1.0

Description:

Description of a Fermi chopper, possibly with curved slits. A Fermi chopper is similar to a disk chopper, but the rotation axis is 0,1,0.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXfermi_chopper can implement: NXIFbeamline_component

type: *NX_CHAR*

Fermi chopper type

rotation_speed: *NX_FLOAT* {units=*NX_FREQUENCY*}

chopper rotation speed

radius: *NX_FLOAT* {units=*NX_LENGTH*}

radius of chopper

slit: *NX_FLOAT* {units=*NX_LENGTH*}

width of an individual slit

r_slit: *NX_FLOAT* {units=*NX_LENGTH*}

radius of curvature of slits

number: *NX_INT* {units=*NX_UNITLESS*}

number of slits

height: *NX_FLOAT* {units=*NX_LENGTH*}

input beam height

width: *NX_FLOAT* {units=*NX_LENGTH*}

input beam width

distance: *NX_FLOAT* {units=*NX_LENGTH*}

distance

wavelength: *NX_FLOAT* {units=*NX_WAVELENGTH*}

Wavelength transmitted by chopper

energy: *NX_FLOAT* {units=*NX_ENERGY*}

energy selected

absorbing_material: *NX_CHAR*

absorbing material

transmitting_material: *NX_CHAR*

transmitting material

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXfermi_chopper.nxdl.xml

NXfilter

Status:

base class, extends *NXobject*, version 1.0

Description:

Template for specifying the state of band pass 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*, *NXlog*, *NXsensor*

Structure:

@implements: *NX_CHAR*

NXfilter can implement: NXIFbeamline_component, NXIFsinglecrystal

description: *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: *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: *NX_FLOAT* {units=*NX_TEMPERATURE*}

average/nominal filter temperature

thickness: *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the filter

density: *NX_NUMBER* {units=*NX_MASS_DENSITY*}

mass density of the filter

m_value: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

m value of supermirror filter

substrate_material: *NX_CHAR*

substrate material of supermirror filter

substrate_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

substrate thickness of supermirror filter

coating_material: *NX_CHAR*

coating material of supermirror filter

substrate_roughness: *NX_FLOAT* {units=*NX_LENGTH*}

substrate roughness (RMS) of supermirror filter

coating_roughness[nsurf]: *NX_FLOAT* {units=*NX_LENGTH*}

coating roughness (RMS) of supermirror filter

transmission: *NXdata*

Wavelength transmission profile of filter

temperature_log: *NXlog*

Linked temperature_log for the filter

sensor_type: *NXsensor*

Sensor(s)used to monitor the filter temperature

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXfilter.nxdl.xml

NXflipper

Status:

base class, extends *NXobject*, version 1.0

Description:

Template of a beamline spin flipper. A spin flipper is a device used for analysis with polarised neutrons. It orients the neutron spin with the help of a magnetic field.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXflipper can implement: NXIFbeamline_component

type: *NX_CHAR*

Any of these values: `coil` | `current-sheet`

flip_turns: *NX_FLOAT* {units=*NX_PER_LENGTH*}

Linear density of turns (such as number of turns/cm) in flipping field coils

comp_turns: *NX_FLOAT* {units=*NX_PER_LENGTH*}

Linear density of turns (such as number of turns/cm) in compensating field coils

guide_turns: *NX_FLOAT* {units=*NX_PER_LENGTH*}

Linear density of turns (such as number of turns/cm) in guide field coils

flip_current: *NX_FLOAT* {units=*NX_CURRENT*}

Flipping field coil current in "on" state"

comp_current: *NX_FLOAT* {units=*NX_CURRENT*}

Compensating field coil current in "on" state"

guide_current: *NX_FLOAT* {units=*NX_CURRENT*}

Guide field coil current in "on" state"

thickness: *NX_FLOAT* {units=*NX_LENGTH*}

thickness along path of neutron travel

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXflipper.nxdl.xml

NXfresnel_zone_plate

Status:

base class, extends *NXobject*, version 1.0

Description:

description for a fresnel zone plate

Symbols:

No symbol table

Groups cited: *NXtransformations*

Structure:

focus_parameters[]: *NX_FLOAT*

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: *NX_FLOAT* {units=*NX_LENGTH*}

outermost_zone_width: *NX_FLOAT* {units=*NX_LENGTH*}

central_stop_diameter: *NX_FLOAT* {units=*NX_LENGTH*}

fabrication: *NX_CHAR*

how the zone plate was manufactured

Any of these values: `etched|plated|zone doubled|other`

zone_height: *NX_FLOAT* {units=*NX_LENGTH*}

zone_material: *NX_CHAR*

Material of the zones themselves

zone_support_material: *NX_CHAR*

Material present between the zones. This is usually only present for the "zone doubled" fabrication process

central_stop_material: *NX_CHAR*

central_stop_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

mask_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

mask_material: *NX_CHAR*

If no mask is present, set mask_thickness to 0 and omit the mask_material field

support_membrane_material: *NX_CHAR*

support_membrane_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

(transformations): *NXtransformations*

"Engineering" position of the fresnel zone plate

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXfresnel_zone_plate.nxdl.xml

NXgeometry

Status:

base class, extends *NXobject*, version 1.0

Description:

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: *NX_CHAR*

Optional description/label. Probably only present if we are an additional reference point for components rather than the location of a real component

component_index: *NX_INT*

Position of the component along the beam path. The sample is at 0, components upstream have negative component_index, components downstream have positive component_index.

(shape): *NXshape*

shape/size information of component

(translation): *NXtranslation*

translation of component

(orientation): *NXorientation*

orientation of component

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXgeometry.nxdl.xml

NXgrating

Status:

base class, extends *NXobject*, version 1.0

Description:

Description for a diffraction grating, as could be used in a soft X-ray monochromator

Symbols:

No symbol table

Groups cited: *NXdata*, *NXshape*, *NXtransformations*

Structure:

angles[2]: *NX_FLOAT* {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[]: *NX_FLOAT* {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: *NX_FLOAT* {units=*NX_UNITLESS*}

depth: *NX_FLOAT* {units=*NX_LENGTH*}

diffraction_order: *NX_INT* {units=*NX_UNITLESS*}

deflection_angle: *NX_FLOAT* {units=*NX_ANGLE*}

Angle between the incident beam and the utilised outgoing beam.

interior_atmosphere: *NX_CHAR*

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

substrate_material: *NX_CHAR*

substrate_density: *NX_FLOAT* {units=*NX_MASS_DENSITY*}

substrate_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

coating_material: *NX_CHAR*

substrate_roughness: *NX_FLOAT* {units=*NX_LENGTH*}

coating_roughness: *NX_FLOAT* {units=*NX_LENGTH*}

layer_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

An array describing the thickness of each layer

shape: *NXshape*

A NXshape group describing the shape of the mirror

figure_data: *NXdata*

Numerical description of the surface figure of the mirror.

(transformations): *NXtransformations*

“Engineering” position of the grating

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXgrating.nxdl.xml

NXguide

Status:

base class, extends *NXobject*, version 1.0

Description:

`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 `NXmirrors` each having their own `NXgeometries` 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*

Structure:

@implements: *NX_CHAR*

NXguide can implement: NXIFbeamline_component

description: *NX_CHAR*

A description of this particular instance of `NXguide`.

incident_angle: *NX_FLOAT* {units=*NX_ANGLE*}

TODO: documentation needed

bend_angle_x: *NX_FLOAT* {units=*NX_ANGLE*}

TODO: documentation needed

bend_angle_y: *NX_FLOAT* {units=*NX_ANGLE*}

TODO: documentation needed

interior_atmosphere: *NX_CHAR*

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

external_material: *NX_CHAR*

external material outside substrate

m_value[nsurf]: *NX_FLOAT*

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

substrate_material[nsurf]: *NX_FLOAT*

TODO: documentation needed

substrate_thickness[nsurf]: *NX_FLOAT* {units=*NX_LENGTH*}

TODO: documentation needed

coating_material[nsurf]: *NX_FLOAT*

TODO: documentation needed

substrate_roughness[nsurf]: *NX_FLOAT* {units=*NX_LENGTH*}

TODO: documentation needed

coating_roughness[nsurf]: *NX_FLOAT* {units=*NX_LENGTH*}

TODO: documentation needed

number_sections: *NX_INT* {units=*NX_UNITLESS*}

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

(geometry): *NXgeometry*

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

reflectivity: *NXdata*

Reflectivity as function of reflecting surface and wavelength

data[nsurf, nwl]: *NX_NUMBER*

reflectivity of each surface as a function of wavelength

@signal: *NX_POSINT*

Use `signal=1` to indicate that this is the plottable data for NeXus.

Obligatory value: `1`

@axes: *NX_CHAR*

Use `axes="surface:wavelength"` to indicate the dimension scales to be used when plotting this data.

Obligatory value: `surface:wavelength`

surface[nsurf]: *NX_NUMBER* {units=*NX_ANY*}

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

wavelength[nwl]: *NX_NUMBER* {units=*NX_WAVELENGTH*}

wavelengths at which reflectivity was measured

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXguide.nxdl.xml

NXinsertion_device

Status:

base class, extends *NXobject*, version 1.0

Description:

Description of an insertion device, as in a synchrotron.

Symbols:

No symbol table

Groups cited: *NXdata*

Structure:

@implements: *NX_CHAR*

NXinsertion_device can implement: NXIFbeamline_component, NXIFmonochromator

type: *NX_CHAR*

Any of these values: `undulator|wiggler`

gap: *NX_FLOAT* {units=*NX_LENGTH*}

separation between opposing pairs of magnetic poles

taper: *NX_FLOAT* {units=*NX_ANGLE*}

angular of gap difference between upstream and downstream ends of the insertion device

phase: *NX_FLOAT* {units=*NX_ANGLE*}

poles: *NX_INT* {units=*NX_UNITLESS*}

number of poles

magnetic_wavelength: *NX_FLOAT* {units=*NX_WAVELENGTH*}

k: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

beam displacement parameter

length: *NX_FLOAT* {units=*NX_LENGTH*}

length of insertion device

power: *NX_FLOAT* {units=*NX_POWER*}

total power delivered by insertion device

energy: *NX_FLOAT* {units=*NX_ENERGY*}

energy of peak intensity in output spectrum

bandwidth: *NX_FLOAT* {units=*NX_ENERGY*}

bandwidth of peak energy

harmonic: *NX_INT* {units=*NX_UNITLESS*}

harmonic number of peak

spectrum: *NXdata*

spectrum of insertion device

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXinsertion_device.nxdl.xml

NXinstrument

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXaperture*, *NXattenuator*, *NXbeam_stop*, *NXbeam*, *NXbending_magnet*, *NXcapillary*, *NXcollection*, *NXcollimator*, *NXcrystal*, *NXdetector_group*, *NXdetector*, *NXdisk_chopper*, *NXevent_data*, *NXfermi_chopper*, *NXfilter*, *NXflipper*, *NXguide*, *NXinsertion_device*, *NXmirror*, *NXmoderator*, *NXmonochromator*, *NXpolarizer*, *NXpositioner*, *NXsource*, *NXvelocity_selector*, *NXxraylens*

Structure:

name: *NX_CHAR*

Name of instrument

@short_name: *NX_CHAR*

short name for instrument, perhaps the acronym

(aperture): *NXaperture*

(attenuator): *NXattenuator*

(beam): *NXbeam*

(beam_stop): *NXbeam_stop*

(bending_magnet): *NXbending_magnet*

(collimator): *NXcollimator*

(collection): *NXcollection*

(capillary): *NXcapillary*

(crystal): *NXcrystal*

(detector): *NXdetector*

(detector_group): *NXdetector_group*

(disk_chopper): *NXdisk_chopper*

(event_data): *NXevent_data*

(fermi_chopper): *NXfermi_chopper*

(filter): *NXfilter*

(flipper): *NXflipper*

(guide): *NXguide*

(insertion_device): *NXinsertion_device*

(mirror): *NXmirror*

(moderator): *NXmoderator*

(monochromator): *NXmonochromator*

(polarizer): *NXpolarizer*

(positioner): *NXpositioner*

(source): *NXsource*

(velocity_selector): *NXvelocity_selector*

(xraylens): *NXxraylens*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXinstrument.nxdl.xml

NXlog

Status:

base class, extends *NXobject*, version 1.0

Description:

Definition of information that is recorded against time, such as information monitored during the run. It contains the logged values and the times at which they were measured as elapsed time since a starting time recorded in ISO8601 format. This method of storing logged data helps to distinguish instances in which a variable is a dimension scale of the data, in which case it is stored in an `NXdata` group, and instances in which it is logged during the run, when it should be stored in an `NXlog` group. Note: When using multiple `NXlog` groups, it is suggested to place them inside a `NXcollection` group. In such cases, when `NXlog` is used in another class, `NXcollection/NXlog` is then constructed.

Symbols:

No symbol table

Groups cited: none

Structure:

time: *NX_FLOAT* {units=*NX_TIME*}

Time of logged entry. The times are relative to the "start" attribute and in the units specified in the "units" attribute.

@start: *NX_DATE_TIME*

value: *NX_NUMBER* {units=*NX_ANY*}

Array of logged value, such as temperature

raw_value: *NX_NUMBER* {units=*NX_ANY*}

Array of raw information, such as thermocouple voltage

description: *NX_CHAR*

Description of logged value

average_value: *NX_FLOAT* {units=*NX_ANY*}

average_value_error: *NX_FLOAT* {units=*NX_ANY*}

estimated uncertainty (often used: standard deviation) of average_value

minimum_value: *NX_FLOAT* {units=*NX_ANY*}

maximum_value: *NX_FLOAT* {units=*NX_ANY*}

duration: *NX_FLOAT* {units=*NX_ANY*}

Total time log was taken

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXlog.nxdl.xml

NXmirror

Status:

base class, extends *NXobject*, version 1.0

Description:

Template of a beamline mirror or supermirror.

Symbols:

No symbol table

Groups cited: *NXdata*, *NXshape*

Structure:

@implements: *NX_CHAR*

NXmirror can implement: NXIFbeamline_component

type: *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: *NX_CHAR*

description of this mirror

incident_angle: *NX_FLOAT* {units=*NX_ANGLE*}

bend_angle_x: *NX_FLOAT* {units=*NX_ANGLE*}

bend_angle_y: *NX_FLOAT* {units=*NX_ANGLE*}

interior_atmosphere: *NX_CHAR*

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

external_material: *NX_CHAR*

external material outside substrate

m_value: *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: *NX_CHAR*

substrate_density: *NX_FLOAT* {units=*NX_MASS_DENSITY*}

substrate_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

coating_material: *NX_CHAR*

substrate_roughness: *NX_FLOAT* {units=*NX_LENGTH*}

coating_roughness: *NX_FLOAT* {units=*NX_LENGTH*}

even_layer_material: *NX_CHAR*

even_layer_density: *NX_FLOAT* {units=*NX_MASS_DENSITY*}

odd_layer_material: *NX_CHAR*

odd_layer_density: *NX_FLOAT* {units=*NX_MASS_DENSITY*}

layer_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

An array describing the thickness of each layer

reflectivity: *NXdata*

Reflectivity as function of wavelength

shape: *NXshape*

A NXshape group describing the shape of the mirror

figure_data: *NXdata*

Numerical description of the surface figure of the mirror.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXmirror.nxdl.xml

NXmoderator

Status:

base class, extends *NXobject*, version 1.0

Description:

This is the description for a general moderator

Symbols:

No symbol table

Groups cited: *NXdata*, *NXlog*

Structure:

@implements: *NX_CHAR*

NXmoderator can implement: NXIFbeamline_component

type: *NX_CHAR*

Any of these values:

- `H20`
- `D20`
- `Liquid H2`
- `Liquid CH4`
- `Liquid D2`
- `Solid D2`

- `C`
- `Solid CH4`
- `Solid H2`

poison_depth: *NX_FLOAT* {units=*NX_LENGTH*}

coupled: *NX_BOOLEAN*

whether the moderator is coupled

coupling_material: *NX_CHAR*

The material used for coupling. Usually Cd.

poison_material: *NX_CHAR*

Any of these values: `Gd` | `Cd`

temperature: *NX_FLOAT* {units=*NX_TEMPERATURE*}

average/nominal moderator temperature

temperature_log: *NXlog*

log file of moderator temperature

pulse_shape: *NXdata*

moderator pulse shape

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXmoderator.nxdl.xml

NXmonitor

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXlog*

Structure:

@implements: *NX_CHAR*

NXmonitor can implement: NXIFbeamline_component, NXIF_single_detector, NXIF_single_detector_tof, NXIF_single_detector_scanned, NXIF_count_time, NXIF_count_time_scanned

mode: *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: *NX_DATE_TIME*

Starting time of measurement

end_time: *NX_DATE_TIME*

Ending time of measurement

preset: *NX_NUMBER* {units=*NX_ANY*}

preset value for time or monitor

distance: *NX_FLOAT* {units=*NX_LENGTH*}

Distance of monitor from sample

range[2]: *NX_FLOAT* {units=*NX_ANY*}

Range (X-axis, Time-of-flight, etc.) over which the integral was calculated

integral: *NX_NUMBER* {units=*NX_ANY*}

Total integral monitor counts

type: *NX_CHAR*

Any of these values: `Fission Chamber|Scintillator`

efficiency[ref(i)]: *NX_NUMBER* {units=*NX_DIMENSIONLESS*}

Monitor efficiency

sampled_fraction: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Proportion of incident beam sampled by the monitor (0<x<1)

count_time: *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

integral_log: *NXlog*

Time variation of monitor counts

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXmonitor.nxdl.xml

NXmonochromator

Status:

base class, extends *NXobject*, version 1.0

Description:

This is a base class for everything which selects a wavelength or energy, be it a monochromator crystal, a velocity selector, an undulator or whatever.

The expected units are:

- wavelength: angstrom
- energy: eV

Symbols:

No symbol table

Groups cited: *NXcrystal*, *NXdata*, *NXgeometry*, *NXvelocity_selector*

Structure:

wavelength: *NX_FLOAT* {units=*NX_WAVELENGTH*}

wavelength selected

wavelength_error: *NX_FLOAT* {units=*NX_WAVELENGTH*}

wavelength standard deviation

energy: *NX_FLOAT* {units=*NX_ENERGY*}

energy selected

energy_error: *NX_FLOAT* {units=*NX_ENERGY*}

energy standard deviation

distribution: *NXdata*

geometry: *NXgeometry*

(crystal): *NXcrystal*

Use as many crystals as necessary to describe

(velocity_selector): *NXvelocity_selector*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXmonochromator.nxdl.xml

NXnote

Status:

base class, extends *NXobject*, version 1.0

Description:

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: *NX_CHAR*

Author or creator of note

date: *NX_DATE_TIME*

Date note created/added

type: *NX_CHAR*

Mime content type of note data field e.g. image/jpeg, text/plain, text/html

file_name: *NX_CHAR*

Name of original file name if note was read from an external source

description: *NX_CHAR*

Title of an image or other details of the note

data: *NX_BINARY*

Binary note data - if text, line terminator is [CR][LF].

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXnote.nxdl.xml

NXobject

Status:

base class, extends none, version 1.0

Description:

This is the base object of NeXus

Symbols:

No symbol table

Groups cited: none

Structure:

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXobject.nxdl.xml

NXorientation

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXgeometry*

Structure:

value[numobj, 6]: *NX_FLOAT* {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): *NXgeometry*

Link to another object if we are using relative positioning, else absent

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXorientation.nxdl.xml

NXparameters

Status:

base class, extends *NXobject*, version 1.0

Description:

Container for parameters, usually used in processing or analysis.

Symbols:

No symbol table

Groups cited: none

Structure:

term: *NX_CHAR*

A parameter (also known as a term) that is used in or results from processing.

@units: *NX_CHAR*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXparameters.nxdl.xml

NXpinhole

Status:

base class, extends *NXobject*, version 1.0

Description:

Reduce the size of an input beam by filtering out a circular section from it.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXattenuator can implement: NXIFbeamline_component, NXIFmetadata

diameter: *NX_FLOAT* {units=*NX_LENGTH*}

Diameter of the pinhole

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXpinhole.nxdl.xml

NXpolarizer

Status:

base class, extends *NXobject*, version 1.0

Description:

Template of a beamline spin polarizer. This is a draft and is subject to revision.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXpolarizer can implement: NXIFbeamline_component

type: *NX_CHAR*

one of these values: “crystal”, “supermirror”, “3He”

composition: *NX_CHAR*

description of the composition of the polarizing material

reflection[3]: *NX_INT* {units=*NX_UNITLESS*}

[hkl] values of nominal reflection

efficiency: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

polarizing efficiency

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXpolarizer.nxdl.xml

NXpositioner

Status:

base class, extends *NXobject*, version 1.0

Description:

This group describes a generic positioner such as a motor or piezo-electric transducer. It is used to document the current information of a piece of beam line equipment. Note: When using multiple `NXpositioner` groups, it is suggested to place them inside a `NXcollection` group. In such cases, when `NXpositioner` is used in another class, `NXcollection/NXpositioner` is then constructed.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXpositioner can implement: NXIFbeamline_component, NXIFmetadata

value[n]: *NX_NUMBER* {units=*NX_ANY*}

best known value of positioner - need [n] as may be scanned

raw_value[n]: *NX_NUMBER* {units=*NX_ANY*}

raw value of positioner - need [n] as may be scanned

target_value[n]: *NX_NUMBER* {units=*NX_ANY*}

targeted (commanded) value of positioner - need [n] as may be scanned

tolerance[n]: *NX_NUMBER* {units=*NX_ANY*}

maximum allowable difference between target_value and value

soft_limit_min: *NX_NUMBER* {units=*NX_ANY*}

minimum allowed limit to set value

soft_limit_max: *NX_NUMBER* {units=*NX_ANY*}

maximum allowed limit to set value

velocity: *NX_NUMBER* {units=*NX_ANY*}

velocity of the positioner (distance moved per unit time)

acceleration_time: *NX_NUMBER* {units=*NX_ANY*}

time to ramp the velocity up to full speed

controller_record: *NX_CHAR*

Hardware device record, e.g. EPICS process variable, taco/tango ...

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXpositioner.nxdl.xml

NXprocess

Status:

base class, extends *NXobject*, version 1.0

Description:

Document an event of data processing, reconstruction, or analysis for this data.

Symbols:

No symbol table

Groups cited: *NXnote*

Structure:

program: *NX_CHAR*

Name of the program used

version: *NX_CHAR*

Version of the program used

date: *NX_DATE_TIME*

Date and time of processing.

(note): *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.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXprocess.nxdl.xml

NXroot

Status:

base class, extends *NXobject*, version 1.0

Description:

Definition of the root NeXus group.

Symbols:

No symbol table

Groups cited: *NXentry*

Structure:

@NX_class: *NX_CHAR*

The root of any NeXus data file is an NXroot class (no other choice is allowed for a valid NeXus data file). This attribute cements that definition.

Obligatory value: `NXroot`

@file_time: *NX_CHAR*

Date and time file was originally created

@file_name: *NX_CHAR*

File name of original NeXus file

@file_update_time: *NX_CHAR*

Date and time of last file change at close

@NeXus_version: *NX_CHAR*

Version of NeXus API used in writing the file

@HDF_version: *NX_CHAR*

Version of NeXus API used in writing the file

@HDF5_Version: *NX_CHAR*

Version of NeXus API used in writing the file. Note this attribute is spelled with uppercase "V", different than other version attributes.

@XML_version: *NX_CHAR*

Version of NeXus API used in writing the file

@creator: *NX_CHAR*

facility or program where file originated

(entry): *NXentry*

entries

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXroot.nxdl.xml

NXsample

Status:

base class, extends *NXobject*, version 1.0

Description:

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

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*, *NXlog*, *NXpositioner*

Structure:

@implements: *NX_CHAR*

NXsample can implement: NXIFbeamline_component, NXIFmetadata, NXIFsinglecrystal

temperature[n_Temp]: *NX_FLOAT* {units=*NX_TEMPERATURE*}

Sample temperature. This could be a scanned variable

electric_field[n_eField]: *NX_FLOAT* {units=*NX_VOLTAGE*}

Applied electric field

@direction: *NX_CHAR*

Any of these values: `x` | `y` | `z`

magnetic_field[n_mField]: *NX_FLOAT* {units=*NX_ANY*}

Applied magnetic field

@direction: *NX_CHAR*

Any of these values: `x` | `y` | `z`

stress_field[n_sField]: *NX_FLOAT* {units=*NX_ANY*}

Applied external stress field

@direction: *NX_CHAR*

Any of these values: `x` | `y` | `z`

pressure[n_pField]: *NX_FLOAT* {units=*NX_PRESSURE*}

Applied pressure

changer_position: *NX_INT* {units=*NX_UNITLESS*}

Sample changer position

sample_orientation[3]: *NX_FLOAT* {units=*NX_ANGLE*}

This will follow the Busing and Levy convention from Acta.Crysta v22, p457 (1967)

mass[n_comp]: *NX_FLOAT* {units=*NX_MASS*}

Mass of sample

density[n_comp]: *NX_FLOAT* {units=*NX_MASS_DENSITY*}

Density of sample

relative_molecular_mass[n_comp]: *NX_FLOAT* {units=*NX_MASS*}

Relative Molecular Mass of sample

type: *NX_CHAR*

Any of these values:

- `sample`
- `sample+can`
- `can`
- `calibration sample`
- `normalisation sample`
- `simulated data`
- `none`
- `sample environment`

situation: *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`

preparation_date: *NX_DATE_TIME*

Date of preparation of the sample

component[n_comp]: *NX_CHAR*

Details of the component of the sample and/or can

sample_component[n_comp]: *NX_CHAR*

What type of component we are: “sample | can | atmosphere | kit”

Any of these values: `sample` | `can` | `atmosphere` | `kit`

concentration[n_comp]: *NX_FLOAT* {units=*NX_MASS_DENSITY*}

Concentration of each component

volume_fraction[n_comp]: *NX_FLOAT*

Volume fraction of each component

scattering_length_density[n_comp]: *NX_FLOAT* {units=*NX_SCATTERING_LENGTH_DENSITY*}

Scattering length density of each component

path_length: *NX_FLOAT* {units=*NX_LENGTH*}

Path length through sample/can for simple case when it does not vary with scattering direction

path_length_window: *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of a beam entry/exit window on the can (mm) - assumed same for entry and exit

thickness: *NX_FLOAT* {units=*NX_LENGTH*}

sample thickness

external_DAC: *NX_FLOAT* {units=*NX_ANY*}

value sent to user's sample setup

short_title: *NX_CHAR*

20 character fixed length sample description for legends

(beam): *NXbeam*

Details of beam incident on sample - used to calculate sample/beam interaction point

transmission: *NXdata*

As a function of Wavelength

temperature_log: *NXlog*

temperature_log.value is a link to e.g. temperature_env.sensor1.value_log.value

temperature_env: *NXenvironment*

Additional sample temperature environment information

magnetic_field_log: *NXlog*

magnetic_field_log.value is a link to e.g. magnetic_field_env.sensor1.value_log.value

magnetic_field_env: *NXenvironment*

Additional sample magnetic environment information

external_ADC: *NXlog*

logged value (or logic state) read from user's setup

(positioner): *NXpositioner*

Any positioner (motor, PZT, ...) used to locate the sample

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXsample.nxdl.xml

NXsensor

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXgeometry*, *NXlog*, *NXorientation*

Structure:

model: *NX_CHAR*

Sensor identification code/model number

name: *NX_CHAR*

Name for the sensor

short_name: *NX_CHAR*

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

attached_to: *NX_CHAR*

where sensor is attached to ("sample" | "can")

measurement: *NX_CHAR*

name for measured signal

Any of these values:

- `temperature`
- `pH`
- `magnetic_field`
- `electric_field`
- `conductivity`
- `resistance`
- `voltage`
- `pressure`
- `flow`
- `stress`
- `strain`
- `shear`
- `surface_pressure`

type: *NX_CHAR*

The type of hardware used for the measurement. Examples (suggestions but not restrictions):

Temperature J | K | T | E | R | S | Pt100 | Rh/Fe

pH Hg/Hg2Cl2 | Ag/AgCl | ISFET

Ion selective electrode specify species; e.g. Ca2+

Magnetic field Hall

Surface pressure wilhelmy plate

run_control: *NX_BOOLEAN*

Is data collection controlled or synchronised to this quantity: 1=no, 0=to "value", 1=to "value_deriv1", etc.

high_trip_value: *NX_FLOAT* {units=*NX_ANY*}

Upper control bound of sensor reading if using run_control

low_trip_value: *NX_FLOAT* {units=*NX_ANY*}

Lower control bound of sensor reading if using run_control

value[n]: *NX_FLOAT* {units=*NX_ANY*}

nominal setpoint or average value - need [n] as may be a vector

value_deriv1[ref(value)]: *NX_FLOAT* {units=*NX_ANY*}

Nominal/average first derivative of value e.g. strain rate - same dimensions as "value" (may be a vector)

value_deriv2[ref(value)]: *NX_FLOAT* {units=*NX_ANY*}

Nominal/average second derivative of value - same dimensions as "value" (may be a vector)

external_field_brief: *NX_CHAR*

Any of these values:

- `along beam`
- `across beam`
- `transverse`
- `solenoidal`
- `flow shear gradient`
- `flow vorticity`

geometry: *NXgeometry*

Defines the axes for logged vector quantities if they are not the global instrument axes

value_log: *NXlog*

Time history of sensor readings

value_deriv1_log: *NXlog*

Time history of first derivative of sensor readings

value_deriv2_log: *NXlog*

Time history of second derivative of sensor readings

external_field_full: *NXorientation*

For complex external fields not satisfied by External_field_brief

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXsensor.nxdl.xml

NXshape

Status:

base class, extends *NXobject*, version 1.0

Description:

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: *NX_CHAR*

general shape of a component

Any of these values:

- `nxflat`
- `nxcylinder`
- `nxbox`
- `nxsphere`
- `nxcone`
- `nxelliptical`
- `nxtoroidal`
- `nxparabolic`
- `nxpolynomial`

size[numobj, nshapepar]: *NX_FLOAT* {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: *NX_CHAR*

Any of these values: `concave` | `convex`

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXshape.nxdl.xml

NXslit

Status:

base class, extends *NXobject*, version 1.0

Description:

A slit is a device which filters out a rectangular section of an input beam. In order to reduce the beam size. two slits following each other at some distance also provide for collimation. In front of detectors slits reduce backgrounds.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXslit can implement: NXIFbeamline_component, NXIFmetadata

x_gap: *NX_FLOAT* {units=*NX_LENGTH*}

Width of the slit in the x direction

y_gap: *NX_FLOAT* {units=*NX_LENGTH*}

Width of the slit in the y direction

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXslit.nxdl.xml

NXsource

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXdata*, *NXgeometry*, *NXnote*

Structure:

@implements: *NX_CHAR*

NXsource can implement: NXIFbeamline_component, NXIFmetadata

type: *NX_CHAR*

type of radiation source (pick one from the enumerated list and spell exactly)

Any of these values:

- `Spallation Neutron Source`
- `Pulsed Reactor Neutron Source`
- `Reactor Neutron Source`
- `Synchrotron X-ray Source`
- `Pulsed Muon Source`
- `Rotating Anode X-ray`
- `Fixed Tube X-ray`
- `UV Laser`
- `Free-Electron Laser`
- `Optical Laser`
- `Ion Source`
- `UV Plasma Source`

probe: *NX_CHAR*

type of radiation probe (pick one from the enumerated list and spell exactly)

Any of these values:

- `neutron`
- `x-ray`
- `muon`
- `electron`
- `ultraviolet`
- `visible light`
- `positron`
- `proton`

power: *NX_FLOAT* {units=*NX_POWER*}

Source power

emittance_x: *NX_FLOAT* {units=*NX_EMITTANCE*}

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

emittance_y: *NX_FLOAT* {units=*NX_EMITTANCE*}

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

sigma_x: *NX_FLOAT* {units=*NX_LENGTH*}

particle beam size in x

sigma_y: *NX_FLOAT* {units=*NX_LENGTH*}

particle beam size in y

flux: *NX_FLOAT* {units=*NX_FLUX*}

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

energy: *NX_FLOAT* {units=*NX_ENERGY*}

Source energy. For storage rings, this would be the particle beam energy. For X-ray tubes, this would be the excitation voltage.

current: *NX_FLOAT* {units=*NX_CURRENT*}

Accelerator, X-ray tube, or storage ring current

voltage: *NX_FLOAT* {units=*NX_VOLTAGE*}

Accelerator voltage

frequency: *NX_FLOAT* {units=*NX_FREQUENCY*}

Frequency of pulsed source

period: *NX_FLOAT* {units=*NX_PERIOD*}

Period of pulsed source

target_material: *NX_CHAR*

Pulsed source target material

Any of these values:

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

number_of_bunches: *NX_INT*

For storage rings, the number of bunches in use.

bunch_length: *NX_FLOAT* {units=*NX_TIME*}

For storage rings, temporal length of the bunch

bunch_distance: *NX_FLOAT* {units=*NX_TIME*}

For storage rings, time between bunches

pulse_width: *NX_FLOAT* {units=*NX_TIME*}

temporal width of source pulse

mode: *NX_CHAR*

source operating mode

Any of these values:

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

top_up: *NX_BOOLEAN*

Is the synchrotron operating in top_up mode?

last_fill: *NX_NUMBER* {units=*NX_CURRENT*}

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

@time: *NX_DATE_TIME*

date and time of the most recent injection.

notes: *NXnote*

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

bunch_pattern: *NXdata*

For storage rings, description of the bunch pattern. This is useful to describe irregular bunch patterns.

title: *NX_CHAR*

name of the bunch pattern

pulse_shape: *NXdata*

source pulse shape

geometry: *NXgeometry*

"Engineering" location of source

distribution: *NXdata*

The wavelength or energy distribution of the source

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXsource.nxdl.xml

NXsubentry

Status:

base class, extends *NXobject*, version 1.0

Description:

`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-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: *NXcharacterization*, *NXcollection*, *NXdata*, *NXinstrument*, *NXmonitor*, *NXnote*, *NXparameters*, *NXprocess*, *NXsample*, *NXuser*

Structure:

@IDF_Version: *NX_CHAR*

ISIS Muon IDF_Version

title: *NX_CHAR*

Extended title for entry

experiment_identifier: *NX_CHAR*

Unique identifier for the experiment, defined by the facility, possibly linked to the proposals

experiment_description: *NX_CHAR*

Brief summary of the experiment, including key objectives.

collection_identifier: *NX_CHAR*

User or Data Acquisition defined group of NeXus files or NXentry

collection_description: *NX_CHAR*

Brief summary of the collection, including grouping criteria.

entry_identifier: *NX_CHAR*

unique identifier for the measurement, defined by the facility.

definition: *NX_CHAR*

Official NeXus NXDL schema to which this subentry conforms

@version: *NX_CHAR*

NXDL version number

@URL: *NX_CHAR*

URL of NXDL file

definition_local: *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: *NX_CHAR*

NXDL version number

@URL: *NX_CHAR*

URL of NXDL file

start_time: *NX_DATE_TIME*

Starting time of measurement

end_time: *NX_DATE_TIME*

Ending time of measurement

duration: *NX_INT* {units=*NX_TIME*}

Duration of measurement

collection_time: *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: *NX_CHAR*

Such as "2007-3". Some user facilities organize their beam time into run cycles.

program_name: *NX_CHAR*

Name of program used to generate this file

@version: *NX_CHAR*

Program version number

@configuration: *NX_CHAR*

configuration of the program

revision: *NX_CHAR*

Revision id of the file due to re-calibration, reprocessing, new analysis, new instrument definition format, ...

@comment: *NX_CHAR*

pre_sample_flightpath: *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: *NXnote*

Description of the full experiment (document in pdf, latex, ...)

notes: *NXnote*

Notes describing entry

thumbnail: *NXnote*

A small image that is representative of the entry. An example of this is a 640x480 jpeg image automatically produced by a low resolution plot of the NXdata.

@mime_type: *NX_CHAR*

The value should be an `image/*`

Obligatory value: `image/*`

(characterization): *NXcharacterization*

(user): *NXuser*

(sample): *NXsample*

(instrument): *NXinstrument*

(collection): *NXcollection*

(monitor): *NXmonitor*

(data): *NXdata*

(parameters): *NXparameters*

(process): *NXprocess*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXsubentry.nxdl.xml

NXtransformations

Status:

contributed definition, extends *NXobject*, version 1.0

Description:

Use `NXtransformations` to gather together any set of movable or fixed elements positioning the device described by the class that contains this.

Symbols:

No symbol table

Groups cited: none

Structure:

anonymous__NEEDS_XSD_CHANGE__: *NX_NUMBER*

Units need to be appropriate for translation or rotation

@transformation_type: *NX_CHAR*

Any of these values: `translation` | `rotation`

@vector: *NX_NUMBER*

Three values that define the axis for this transformation

@offset: *NX_NUMBER*

A fixed offset applied before the transformation (three vector components).

@offset_units: *NX_CHAR*

Units of the offset.

@depends_on: *NX_CHAR*

Points to the path of the next element in the geometry chain.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/contributed/NXtransformations.nxdl.xm

NXtranslation

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: *NXgeometry*

Structure:

distances[numobj, 3]: *NX_FLOAT* {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: *NXgeometry*

Link to other object if we are relative , else absent

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXtranslation.nxdl.xml

NXuser

Status:

base class, extends *NXobject*, version 1.0

Description:

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

Symbols:

No symbol table

Groups cited: none

Structure:

name: *NX_CHAR*

Name of user responsible for this entry

role: *NX_CHAR*

Role of user responsible for this entry. Suggested roles are "local_contact", "principal_investigator", and "proposer"

affiliation: *NX_CHAR*

Affiliation of user

address: *NX_CHAR*

Address of user

telephone_number: *NX_CHAR*

Telephone number of user

fax_number: *NX_CHAR*

Fax number of user

email: *NX_CHAR*

Email of user

facility_user_id: *NX_CHAR*

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

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXuser.nxdl.xml

NXvelocity_selector

Status:

base class, extends *NXobject*, version 1.0

Description:

This is the description for a (typically neutron) velocity selector. A neutron velocity selector looks like a turbine but with curved blades. Only neutrons which match the speed of the rotating blades make it through. Thus a velocity selector is a monochromator. It can be adjusted by varying the speed of the turbine.

Symbols:

No symbol table

Groups cited: none

Structure:

@implements: *NX_CHAR*

NXvelocity_selector can implement: NXIFbeamline_component, NXIFmetadata, NXIF-monochromator

rotation_speed: *NX_FLOAT* {units=*NX_FREQUENCY*}

velocity selector rotation speed

radius: *NX_FLOAT* {units=*NX_LENGTH*}

radius at beam centre

spwidth: *NX_FLOAT* {units=*NX_LENGTH*}

spoke width at beam centre

length: *NX_FLOAT* {units=*NX_LENGTH*}

rotor length

num: *NX_INT* {units=*NX_UNITLESS*}

number of spokes/lamella

twist: *NX_FLOAT* {units=*NX_ANGLE*}

twist angle along axis

table: *NX_FLOAT* {units=*NX_ANGLE*}

offset vertical angle

height: *NX_FLOAT* {units=*NX_LENGTH*}

input beam height

width: *NX_FLOAT* {units=*NX_LENGTH*}

input beam width

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXvelocity_selector.nxdl.xml

NXxraylens

Status:

base class, extends *NXobject*, version 1.0

Description:

This is a dictionary of field names to use for describing a X-ray lens as used at synchrotron beam lines. Based on information provided by Gerd Wellenreuther.

Symbols:

No symbol table

Groups cited: *NXnote*

Structure:

@implements: *NX_CHAR*

NXxraylens can implement: NXIFbeamline_component, NXIFmetadata

lens_geometry: *NX_CHAR*

Geometry of the lens

Any of these values:

- `paraboloid`
- `spherical`
- `elliptical`
- `hyperbolical`

symmetric: *NX_BOOLEAN*

Is the device symmetric?

cylindrical: *NX_BOOLEAN*

Is the device cylindrical?

focus_type: *NX_CHAR*

The type of focus of the lens

Any of these values: `line`|`point`

lens_thickness: *NX_FLOAT* {units=*NX_LENGTH*}

Thickness of the lens

lens_length: *NX_FLOAT* {units=*NX_LENGTH*}

Length of the lens

curvature: *NX_FLOAT* {units=*NX_LENGTH*}

Radius of the curvature as measured in the middle of the lens

aperture: *NX_FLOAT* {units=*NX_LENGTH*}

Diameter or radius of the lens.

number_of_lenses: *NX_INT*

Number of lenses that make up the compound lens.

lens_material: *NX_CHAR*

Material used to make the lens.

gas: *NX_CHAR*

Gas used to fill the lens

gas_pressure: *NX_FLOAT* {units=*NX_PRESSURE*}

Gas pressure in the lens

cylinder_orientation: *NXnote*

Orientation of the cylinder axis.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXxraylens.nxdl.xml

10.3.2 Interfaces

A description of the NeXus Interfaces are given. NeXus interfaces are groups of common fields and structures which can be implemented by one or more base classes. NeXus interfaces also serve to structure the NeXus base classes better.

NXIFbeamline_component

Status:

base class, extends *NXobject*, version 1.0

Description:

This is the interface for a beamline component. It contains those fields which are necessary to describe the position of the component in the beamline. All fields are axes. Each field has an offset attribute which generally is 0,0,0. But if there is a translational offset to the previous operation, then offset is the place to store it. Each field also has a depends_on attribute. This attribute serves to reconstruct the sequence of operations applied to position the component.

Symbols:

No symbol table

Groups cited: none

Structure:

distance: *NX_LENGTH*

Distance along the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

height: *NX_LENGTH*

Height in relation to the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_translation: *NX_LENGTH*

Translation peperndicular to the beam in the scattering plane

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

rotation_angle: *NX_ANGLE*

Rotation of the component around y

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

azimuthal_angle: *NX_ANGLE*

Rotation of the component around z

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

meridional_angle: *NX_ANGLE*

Rotation of the component around x

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

depends_on: *NX_CHAR*

This fields value is the axes on top of the dependency chain. Or in other words: it names the last element in the chain of operations required to position this component. By starting at this named element and working down the depends_on chain the transformation matrix for positioning the component can be reconstructed.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFbeamline_component.nxdl.xr

NXIFgeneric_detector_tof

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a generic detector operated in time of flight mode. This is the complicated case when there is no regluarity in the detector and the position and orientation of each pixel has to be described explicitly. ISIS is well known to have such detectors.

The translation and rotation fields described here shadow those of NXbeamline_component.

Symbols:

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

ndet: The number of detectors in the detector

tof: number of bins in the time-of-flight histogram

Groups cited: none

Structure:

data[ndet, tof]: *NX_NUMBER*

The counts collected in each detector

@signal: *NX_POSINT*

Obligatory value: `1`

efficiency[ndet]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[ndet, tof]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

dead_time[ndet]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

distance[ndet]: *NX_LENGTH*

Distance along the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

height[ndet]: *NX_LENGTH*

Height in relation to the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_translation[ndet]: *NX_LENGTH*

Translation peperndicular to the beam in the scattering plane

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

rotation_angle[ndet]: *NX_ANGLE*

Rotation of the component around y

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

azimuthal_angle[ndet]: *NX_ANGLE*

Rotation of the component around z

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

meridional_angle[ndet]: *NX_ANGLE*

Rotation of the component around x

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

time_of_flight[tof+1]: *NX_FLOAT* {units=*NX_TIME_OF_FLIGHT*}

Total time of flight

@axis: *NX_POSINT*

Obligatory value: `3`

raw_time_of_flight[tof+1]: *NX_INT* {units=*NX_PULSES*}

In DAQ clock pulses

@frequency: *NX_NUMBER*

Clock frequency in Hz

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFgeneric_detector_tof.nxdl.xm

NXIFsingle_detector_tof

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a single counter in time-of-flight mode

Symbols:

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

tof: number of bins in the time-of-flight histogram

Groups cited: none

Structure:

data[tof]: *NX_NUMBER*

The counts collected in the single detector versus neutron time-of-flight

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `time_of_flight`

efficiency: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[tof]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

time_of_flight[tof+1]: *NX_FLOAT* {units=*NX_TIME_OF_FLIGHT*}

Total time of flight

dead_time: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

raw_time_of_flight[tof+1]: *NX_INT* {units=*NX_PULSES*}

In DAQ clock pulses

@frequency: *NX_NUMBER*

Clock frequency in Hz

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFsingle_detector_tof.nxdl.xml

NXIFarea_detector

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a regular 2D area detector

Symbols:

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

i: number of detector pixels in the first (X, slowest) direction

j: number of detector pixels in the second (Y, faster) direction

Groups cited: none

Structure:

data[i, j]: *NX_NUMBER*

The counts collected in the area detector

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `x_pixel_offset,y_pixel_offset`

efficiency[i, j]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[i, j]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

x_pixel_offset[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

@axis: *NX_POSINT*

Obligatory value: `1`

@primary: *NX_POSINT*

Obligatory value: `1`

y_pixel_offset[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in the y-direction. Can be multidimensional when different values are required for each pixel.

@axis: *NX_POSINT*

Obligatory value: `2`

@primary: *NX_POSINT*

Obligatory value: `1`

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

y_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

beam_center_x: *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: *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.

dead_time[i, j]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFarea_detector.nxdl.xml

NXIFarea_detector_scanned

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a regular 2D area detector whch has been scanned against a variable.

Symbols:

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

i: number of detector pixels in the first (X, slowest) direction

j: number of detector pixels in the second (Y, faster) direction

np: number of scan points (only present in scanning measurements)

scan_axis: The name of the variable which was varied during the scan. May be in another group.

Groups cited: none

Structure:

data[np, i, j]: *NX_NUMBER*

The counts collected in the area detector

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `scan_axis,x_pixel_offset,y_pixel_offset`

efficiency[i, j]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[np, i, j]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

x_pixel_offset[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

@axis: *NX_POSINT*

Obligatory value: `2`

@primary: *NX_POSINT*

Obligatory value: `1`

y_pixel_offset[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in the y-direction. Can be multidimensional when different values are required for each pixel.

@axis: *NX_POSINT*

Obligatory value: `3`

@primary: *NX_POSINT*

Obligatory value: `1`

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

y_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

beam_center_x: *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: *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.

dead_time[i, j]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

sequence_number[np]: *NX_CHAR*

In order to properly sort the order of the images taken in (for example) a tomography experiment, a sequence number is stored with each image.

frame_start_number: *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.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFarea_detector_scanned.nxdl.x

NXIFarea_detector_tof

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a regular 2D area detector operated in time of flight mode

Symbols:

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

i: number of detector pixels in the first (X, slowest) direction

j: number of detector pixels in the second (Y, faster) direction

tof: number of bins in the time-of-flight histogram

Groups cited: none

Structure:

data[i, j, tof]: *NX_NUMBER*

The counts collected in the area detector

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `x_pixel_offset,y_pixel_offset,time_of_flight`

efficiency[i, j]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[i, j, tof]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

x_pixel_offset[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

@axis: *NX_POSINT*

Obligatory value: `1`

@primary: *NX_POSINT*

Obligatory value: `1`

y_pixel_offset[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in the y-direction. Can be multidimensional when different values are required for each pixel.

@axis: *NX_POSINT*

Obligatory value: `2`

@primary: *NX_POSINT*

Obligatory value: `1`

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

y_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

beam_center_x: *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: *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.

dead_time[i, j]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

time_of_flight[tof+1]: *NX_FLOAT* {units=*NX_TIME_OF_FLIGHT*}

Total time of flight

@axis: *NX_POSINT*

Obligatory value: `3`

raw_time_of_flight[tof+1]: *NX_INT* {units=*NX_PULSES*}

In DAQ clock pulses

@frequency: *NX_NUMBER*

Clock frequency in Hz

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFarea_detector_tof.nxdl.xml

NXIFcount_time

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface defines how to store count time data.

Symbols:

No symbol table

Groups cited: none

Structure:

start_time: *NX_FLOAT* {units=*NX_TIME*}

start time for each frame, with the `start` attribute as absolute reference

stop_time: *NX_FLOAT* {units=*NX_TIME*}

stop time for each frame, with the `start` attribute as absolute reference

real_time: *NX_NUMBER* {units=*NX_TIME*}

real-time of the exposure (use this if exposure time varies for each array element, otherwise use `count_time` field)

count_time: *NX_NUMBER* {units=*NX_TIME*}

Elapsed actual counting time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFcount_time.nxdl.xml

NXIFcount_time_scanned

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface defines how to store count time data.

Symbols:

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

np: number of scan points (only present in scanning measurements)

Groups cited: none

Structure:

start_time[np]: *NX_FLOAT* {units=*NX_TIME*}

start time for each frame, with the `start` attribute as absolute reference

@start: *NX_DATE_TIME*

stop_time[np]: *NX_FLOAT* {units=*NX_TIME*}

stop time for each frame, with the `start` attribute as absolute reference

@start: *NX_DATE_TIME*

real_time[np]: *NX_NUMBER* {units=*NX_TIME*}

real-time of the exposure (use this if exposure time varies for each array element, otherwise use `count_time` field)

count_time[np]: *NX_NUMBER* {units=*NX_TIME*}

Elapsed actual counting time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFcount_time_scanned.nxdl.xm

NXIFgasdetector

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface defines a few fields which are special to gas detectors.

Symbols:

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

i: number of detector pixels in the first (X, slowest) direction

j: number of detector pixels in the second (Y, faster) direction

Groups cited: none

Structure:

gas_pressure[i, j]: *NX_FLOAT* {units=*NX_PRESSURE*}

Detector gas pressure

detection_gas_path: *NX_FLOAT* {units=*NX_LENGTH*}

maximum drift space dimension

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFgasdetector.nxdl.xml

NXIFgeneric_detector

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a generic detector. This is the complicated case when there is no regluarity in the detector and the position and orientation of each pixel has to be described explicitly. ISIS is well known to have such detectors.

The translation and rotation fields described here shadow those of NXbeamline_component.

Symbols:

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

ndet: The number of detectors in the detector

Groups cited: none

Structure:

data[ndet]: *NX_NUMBER*

The counts collected in each detector

@signal: *NX_POSINT*

Obligatory value: `1`

efficiency[ndet]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[ndet]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

dead_time[ndet]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

distance[ndet]: *NX_LENGTH*

Distance along the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

height[ndet]: *NX_LENGTH*

Height in relation to the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_translation[ndet]: *NX_LENGTH*

Translation peperndicular to the beam in the scattering plane

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

rotation_angle[ndet]: *NX_ANGLE*

Rotation of the component around y

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

azimuthal_angle[ndet]: *NX_ANGLE*

Rotation of the component around z

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

meridional_angle[ndet]: *NX_ANGLE*

Rotation of the component around x

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFgeneric_detector.nxdl.xml

NXIFgeneric_detector_scanned

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a generic detector which has been scanned against some variable. This is the complicated case when there is no regluarity in the detector and the position and orientation of each pixel has to be described explicitly. ISIS is well known to have such detectors.

The translation and rotation fields described here shadow those of NXbeamline_component.

Symbols:

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

ndet: The number of detectors in the detector

np: number of scan points (only present in scanning measurements)

scan_axis: The name of the variable which was varied during the scan. May be in another group.

Groups cited: none

Structure:

data[np, ndet]: *NX_NUMBER*

The counts collected in each detector

@signal: *NX_POSINT*

Obligatory value: `1`

efficiency[ndet]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[np, ndet]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

dead_time[ndet]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

distance[ndet]: *NX_LENGTH*

Distance along the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

height[ndet]: *NX_LENGTH*

Height in relation to the beam

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

x_translation[ndet]: *NX_LENGTH*

Translation peperndicular to the beam in the scattering plane

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

rotation_angle[ndet]: *NX_ANGLE*

Rotation of the component around y

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,1,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

azimuthal_angle[ndet]: *NX_ANGLE*

Rotation of the component around z

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `0,0,1`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

meridional_angle[ndet]: *NX_ANGLE*

Rotation of the component around x

@transformation_type: *NX_CHAR*

Obligatory value: `rotation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFgeneric_detector_scanned.nx

NXIFlinear_detector

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a linear 1D detector

Symbols:

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

i: number of detector pixels

Groups cited: none

Structure:

data[i]: *NX_NUMBER*

The counts collected in the area detector

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `x_pixel_offset`

efficiency[i]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[i]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

x_pixel_offset[i]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

@axis: *NX_POSINT*

Obligatory value: `1`

@primary: *NX_POSINT*

Obligatory value: `1`

x_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

dead_time[i]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFlinear_detector.nxdl.xml

NXIFlinear_detector_scanned

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a linear 1D detector being used in a scan

Symbols:

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

i: number of detector pixels

np: number of scan points (only present in scanning measurements)

scan_axis: The name of the variable which was varied during the scan. May be in another group.

Groups cited: none

Structure:

data[np, i]: *NX_NUMBER*

The counts collected in the area detector

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `scan_axis,x_pixel_offset`

efficiency[i]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[np, i]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

x_pixel_offset[i]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

@axis: *NX_POSINT*

Obligatory value: `1`

@primary: *NX_POSINT*

Obligatory value: `1`

x_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

dead_time[i]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFlinear_detector_scanned.nxdl

NXIFlinear_detector_tof

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a linear 1D detector in time of flight mode

Symbols:

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

i: number of detector pixels

tof: number of bins in the time-of-flight histogram

Groups cited: none

Structure:

data[i, tof]: *NX_NUMBER*

The counts collected in the area detector

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `x_pixel_offset,time_of_flight`

efficiency[i]: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error[i, tof]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

x_pixel_offset[i]: *NX_FLOAT* {units=*NX_LENGTH*}

Offset from the detector center in x-direction. Can be multidimensional when needed.

@transformation_type: *NX_CHAR*

Obligatory value: `translation`

@vector: *NX_CHAR*

Obligatory value: `1,0,0`

@offset: *NX_CHAR*

@depends_on: *NX_CHAR*

@axis: *NX_POSINT*

Obligatory value: `1`

@primary: *NX_POSINT*

Obligatory value: `1`

x_pixel_size[i, j]: *NX_FLOAT* {units=*NX_LENGTH*}

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

dead_time[i]: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

time_of_flight[tof+1]: *NX_FLOAT* {units=*NX_TIME_OF_FLIGHT*}

Total time of flight

@axis: *NX_POSINT*

Obligatory value: `3`

raw_time_of_flight[tof+1]: *NX_INT* {units=*NX_PULSES*}

In DAQ clock pulses

@frequency: *NX_NUMBER*

Clock frequency in Hz

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFlinear_detector_tof.nxdl.xml

NXIFmetadata

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface determines general matadata which may occur in almost any component.

Symbols:

No symbol table

Groups cited: none

Structure:

type: *NX_CHAR*

A descriptive string describing the type of the component.

description: *NX_CHAR*

A more detailed description of the component.

name: *NX_CHAR*

A longer name of the component

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFmetadata.nxdl.xml

NXIFmonochromator

Status:

base class, extends *NXobject*, version 1.0

Description:

This is the interface implemented by any monochromating device.

Symbols:

No symbol table

Groups cited: none

Structure:

wavelength: *NX_WAVELENGTH*

The nominal wavelength of the monochromator

wavelength_spread: *NX_CHAR*

A measure for the wavelength distribution of the monochromator

energy: *NX_ENERGY*

The wavelength expressed as energy.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFmonochromator.nxdl.xml

NXIFphoton_counter

Status:

base class, extends *NXobject*, version 1.0

Description:

This is an interface defining the additional fields necessary to descrive single photon counting detector like the DECTRIS detectors. Please note that this interface is additional to the appropriate area_detector interfaces and has to be used together with one of them.

Symbols:

No symbol table

Groups cited: none

Structure:

angular_calibration_applied: *NX_BOOLEAN*

True when the angular calibration has been applied in the electronics, false otherwise.

angular_calibration[i, j]: *NX_FLOAT*

Angular calibration data.

flatfield_applied: *NX_BOOLEAN*

True when the flat field correction has been applied in the electronics, false otherwise.

flatfield[i, j]: *NX_FLOAT*

Flat field correction data.

flatfield_error[i, j]: *NX_FLOAT*

Errors of the flat field correction data.

pixel_mask_applied: *NX_BOOLEAN*

True when the pixel mask correction has been applied in the electronics, false otherwise.

pixel_mask[i, j]: *NX_INT*

The 32-bit pixel mask for the detector. 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)

The normal data analysis software would not take pixels into account when a bit in (mask & 0x00FF) 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 as well of course).

countrate_correction__applied: *NX_BOOLEAN*

True when a count-rate correction has already been applied in the electronics, false otherwise.

bit_depth_readout: *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: *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: *NX_FLOAT* {units=*NX_TIME*}

Time it takes to start exposure after a trigger signal has been received. This is important to know for time resolved experiments.

trigger_dead_time: *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[NP]: *NX_FLOAT* {units=*NX_TIME*}

This is time for each frame. This is exposure_time + readout time.

gain_setting: *NX_CHAR*

The gain setting of the detector. This influences background etc.

Any of these values: `high` | `standard` | `fast` | `auto`

saturation_value: *NX_INT*

The value at which the detector goes into saturation. Especially common to CCD detectors, the data is known to be invalid above this value.

number_of_cycles: *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: *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: *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: *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.

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFphoton_counter.nxdl.xml

NXIFsingle_detector

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a single counter

Symbols:

No symbol table

Groups cited: none

Structure:

data: *NX_NUMBER*

The counts collected in the single detector

@signal: *NX_POSINT*

Obligatory value: `1`

efficiency: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

data_error: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

dead_time: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFsingle_detector.nxdl.xml

NXIFsingle_detector_scanned

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface is for a single counter which was scanned against some variable

Symbols:

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

np: number of scan points (only present in scanning measurements)

scan_axis: The name of the variable which was varied during the scan. May be in another group. May only have meaning within the NXdata group.

Groups cited: none

Structure:

data[np]: *NX_NUMBER*

The counts collected in the single detector versus some scanned parameter

@signal: *NX_POSINT*

Obligatory value: `1`

@axes: *NX_CHAR*

Obligatory value: `scan_axis`

data_error[np]: *NX_NUMBER* {units=*NX_ANY*}

The best estimate of the uncertainty in the data value. Where possible, this should be the standard deviation, which has the same units as the data.

efficiency: *NX_FLOAT* {units=*NX_DIMENSIONLESS*}

Detection efficiency of the detector

dead_time: *NX_FLOAT* {units=*NX_TIME*}

Detector dead time

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFsingle_detector_scanned.nxd

NXIFsinglecrystal

Status:

base class, extends *NXobject*, version 1.0

Description:

This interface describes parameters for a single crystal.

Symbols:

No symbol table

Groups cited: none

Structure:

chemical_formula: *NX_CHAR*

The chemical formula specified using CIF conventions. Abbreviated version of CIF standard:

- Only recognized element symbols may be used.
- Each element symbol is followed by a 'count' number. A count of '1' may be omitted.
- A space or parenthesis must separate each cluster of (element symbol + count).
- Where a group of elements is enclosed in parentheses, the multiplier for the group must follow the closing parentheses. That is, all element and group multipliers are assumed to be printed as subscripted numbers.
- Unless the elements are ordered in a manner that corresponds to their chemical structure, the order of the elements within any group or moiety depends on whether or not carbon is present.
- If carbon is present, the order should be:
  - C, then H, then the other elements in alphabetical order of their symbol.
  - If carbon is not present, the elements are listed purely in alphabetic order of their symbol.
- This is the *Hill* system used by Chemical Abstracts.

unit_cell_a: *NX_FLOAT* {units=*NX_LENGTH*}

Unit cell lattice parameter: length of side a

unit_cell_b: *NX_FLOAT* {units=*NX_LENGTH*}

Unit cell lattice parameter: length of side b

unit_cell_c: *NX_FLOAT* {units=*NX_LENGTH*}

Unit cell lattice parameter: length of side c

unit_cell_alpha: *NX_FLOAT* {units=*NX_ANGLE*}

Unit cell lattice parameter: angle alpha

unit_cell_beta: *NX_FLOAT* {units=*NX_ANGLE*}

Unit cell lattice parameter: angle beta

unit_cell_gamma: *NX_FLOAT* {units=*NX_ANGLE*}

Unit cell lattice parameter: angle gamma

unit_cell_volume[n_comp]: *NX_FLOAT* {units=*NX_VOLUME*}

Unit cell

orientation_matrix[n_comp, 3, 3]: *NX_FLOAT*

Orientation matrix of single crystal filter using Busing-Levy convention

unit_cell_class[n_comp]: *NX_CHAR*

In case it is all we know and we want to record/document it

Any of these values:

- `cubic`
- `tetragonal`
- `orthorhombic`

- `monoclinic`
- `triclinic`

unit_cell_group[n_comp]: *NX_CHAR*

Crystallographic point or space group

space_group: *NX_CHAR*

Space group of crystal structure

Source: Automatically generated from https://github.com/nexusformat/definitions/blob/master/base/NXIFsinglecrystal.nxdl.xml

---

Publishing Information

This manual built 2016-07-15 13:55:20 CEST.

See also:

This document is available in different formats:

online HTML http://download.nexusformat.org/doc/html/index.html

A very brief overview is also available (separate from the manual).

HTML http://htmlpreview.github.io/?https://github.com/nexusformat/communications/blob/master/impatient/_build/html/index.l