PaNOSC
Photon and Neutron Open Science Cloud
H2020-INFRAEOSC-04-2018
Grant Agreement Number: 823852

Deliverable: D3.5 NeXus Metadata Schema

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Project Deliverable Information Sheet

Project Reference No.	823852
Project acronym:	PaNOSC
Project full name:	Photon and Neutron Open Science Cloud
H2020 Call:	INFRAEOSC-04-2018
Project Coordinator	Andy Götz (andy.gotz@esrf.fr)
Coordinating Organization:	ESRF
Project Website:	www.panosc.eu
Deliverable No:	D3.5
Deliverable Type:	Report
Dissemination Level	Public
Contractual Delivery Date:	2022-05-31
Actual Delivery Date:	2022-05-31
EC project Officer:	Flavius Alexandru Pana

Document Control Sheet

Document	Title: NeXus Metadata Schema
	Version: 1
	Available at: https://github.com/panosc-eu/panosc
	Files:
Authorship	Written by:
	Contributors: Tobias Richter (ESS), Fredrik Bolmsten (ESS), Massimiliano Novelli (ESS), Simon Heybrook (ESS), Kenan Muric (ESS), Teodor Ivănoaica (ELI ERIC) Balázs Bagó (ELI-ALPS), Alessandro Olivo (CERIC-ERIC), Andrea Lorenzon (CERIC-ERIC), Andy Goetz (ESRF), Axel Bocciarelli (ESRF), Wout De Nolf (ESRF), Henri Payno (ESRF), Fabio Dall'Antonia (XFEL.EU), Yury Kirienko (XFEL.EU), Luis Maia (XFEL.EU), Sandor Brockhauser (XFEL.EU), Stuart Caunt (ILL)
	Reviewed by: Teodor Ivănoaica
	Approved: Jordi Bodera

List of Participants

Participant No.	Participant organisation name	Country
1	European Synchrotron Radiation Facility (ESRF)	France
2	Institut Laue-Langevin (ILL)	France
3	European XFEL (XFEL.EU)	Germany
4	European Spallation Source (ESS)	Sweden
5	Extreme Light Infrastructure ERIC (ELI-ERIC)	Czech Republic
6	Central European Research Infrastructure Consortium (CERIC-ERIC)	Italy
7	EGI Foundation (EGI.eu)	The Netherlands

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Introduction

The work package partners identified the work needed to map their experiments into NeXus and what steps are needed to make NeXus "fit" their needs. As a starting point a number of "*show and tell*" sessions were organised where the WP3 partners shared their practices regarding data management in relation to NeXus. This was done to find commonalities and learn from each other. These commonalities helped to identify several ways to extract metadata for storage in data catalogues.

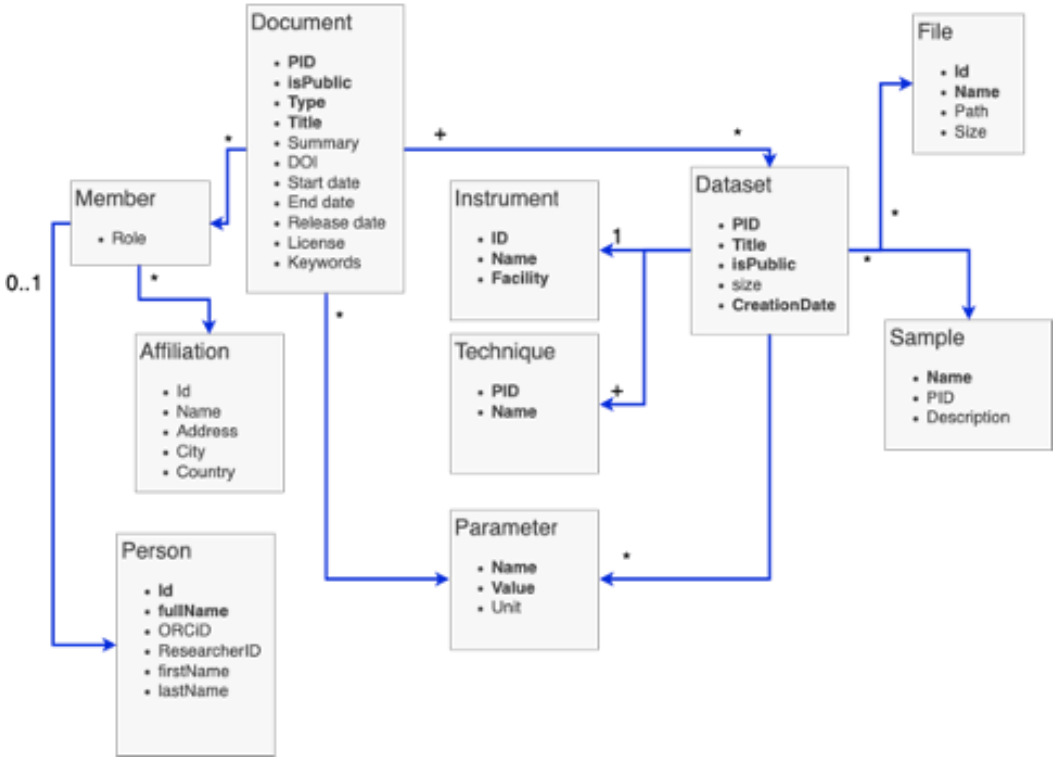

Figure 1: Search API Data Model

The work package also explored the use of NeXus defined terms to use as labels in the data catalogue, either directly or in some derived form. This would mean using NeXus derived terms for storage and/or query of dataset parameters. This relates to the "Parameter" in the data model for the (federatable, domain specific) search API depicted in Figure 1. The flexibility and hierarchical nature of NeXus made this approach cumbersome. The same physical property can appear in different locations under different names in a NeXus file. Interpreting a file always requires some context, which goes against extracting individual identifiers and using them as self-describing labels in a different environment. In addition, files can contain values in different forms, such as individual scalar values, arrays or time series, which requires more effort to arrive at a quantity to compare with for filtering purposes.

This deliverable gives some background on NeXus and summarises the key takeaway messages from the "Show & Tell" NeXus survey between the facilities. The survey mostly showed a need to further develop NeXus support at each facility and strive towards more compliance with the standard. This is reflected in the main part of the deliverable where the effort directed towards NeXus conformity and support are reported per partner.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

NeXus and Photon and Neutron Science

The NeXus Data Format

NeXus is a community defined format for data at photon and neutron facilities. It is mainly used for raw data and is actively written by at least a subset of instruments at most PaNOSC partners. As an underlying container format, NeXus uses HDF5 which enables a hierarchical organisation of any number of datasets or arbitrary dimensions within a file. NeXus puts constraints on the layout and naming of datasets and groups within the file, with the aim of making the resulting file self-describing to a domain expert. In principle, this allows software developers to develop data processing code that can work with files irrespective of what instrument produced the data. In practice, files acquired on different instruments using the same technique can be significantly different, as NeXus provides a large amount of flexibility in how data is organised and stored. This flexibility is required to accurately reflect the realities of an experiment but, it causes issues for software developers and scientists trying to write generic processing software.

NeXus and PaNOSC Survey

At the start of the project a number of "*Show and Tell*" sessions were organised where each site had a presentation on their current state with NeXus and an overview of how datasets with associated metadata are currently being handled. These sessions were then summarised and presented at the PaNOSC meeting in Grenoble in September 2019.

It was found that NeXus has made it into most facilities while the rest were planning to adopt it in the near term.

As NeXus allows for different levels of integration as well as the flexibility to store data in different areas of the structures, the usage of NeXus varies between facilities and instruments. The communalities and differences were noted in the sessions and guided the work moving forward. Additionally, in the session it was observed that facilities that have been operational for a long time have a more heterogeneous environment with multiple file formats and a more divergent set of metadata. The process of transitioning to the NeXus format from older formats is time consuming and requires insight into the data captured. An example of this can be seen in Figure 2, where the data collected has been stored in plain text and where the first lines contain the header information for the data collected.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

	CERIC	ESS	ELI	ESRF	ILL	XFEL
User Portal	VUO	UOS	—	SMIS	ILL Own	UPEX
Metadata Catalogue	VUO (ICAT is an option)	SciCat	TBD	ICAT	ILL Own	MyMdC
Datafiles	NeXus, HDF5, ASCII and many others	NeXus/ HDF5	—	EDF, SPEC, MCA, CBF, CCD, MCCD, HDF5, NeXus	NeXus and ILL Ascii	HDF5
NeXus	Yes*	Yes	No	Yes	Yes*	No
Logbook	DonkeyLog	SciChat	ELI Own	ESRF Own	ILL Own	Elog

**Partial usage*

Table 1: Summation of the "Show and tell" sessions (Status September 2019)

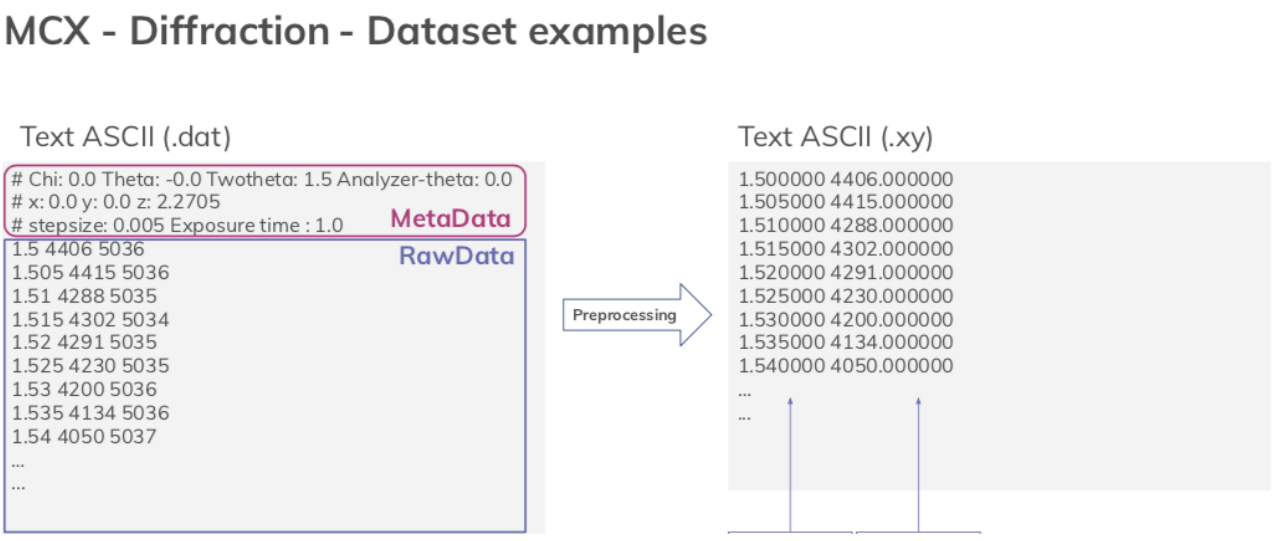

Figure 2: Data from MCX diffraction at CERIC.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Development Work as Part of PaNOSC

With the exception of the question of whether NeXus terms could be used for formulating queries to the search API, the development work to make NeXus work or work “better” for the facilities was largely an individual effort by the facilities. This is reflected in the following sections.

ESS

At the ESS the aim is to write relatively generic NeXus files, following a common scheme for all instruments. The idea is to use NeXus for raw data files to describe the experimental realities as accurately as needed, and leave the interpretation to the data reduction (downstream processing) step. This works with the existing NeXus base class definitions, with minor improvements and clarifications. The two major issues centre around tooling and verification as well as ambiguous interpretations. For the software support, ESS developed an application to create NeXus files (or their templates) and visually verify their correct representation of the experiment. Regarding the clasifications, a few improvements of the NeXus definitions have been proposed and accepted. In addition, the data processing library scipp has started documenting the necessary assumptions.

Geometry Definition and Tooling

ESS uses a file writer service, responsible for writing instrument-specific data in a NeXus file format. It fetches the relevant data from a Kafka cluster and populates the NeXus file with the data according to a predefined configuration given by a template file. This template file is written in a JSON format and is sent to the file writer for every file to be written. The configuration template file is instrument specific. The NeXus Constructor application (https://github.com/ess-dmsc/nexus-constructor) enables the user to build the file writer configuration files supported by the NeXus class definitions and with a 3D rendering of the modelled instrument geometry for easy verification.

The NeXus Constructor GUI can be seen in Figure 3. The instrument view on the right-hand side renders a 3D representation of the instrument and its devices. Any component of the instrument that has dimensions, location and a shape will be shown in the instrument view. On the left, the corresponding NeXus tree structure is given. The user can modify the instrument by editing this tree structure to change the configuration JSON file.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

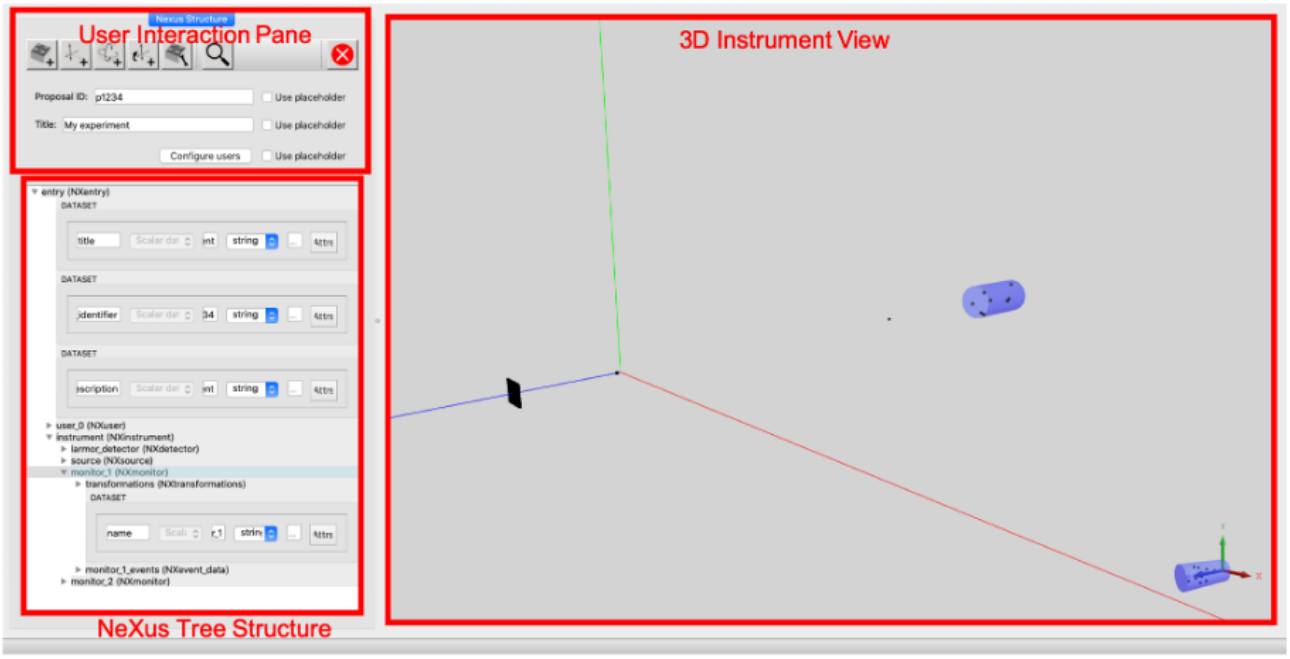

Figure 3: The NeXus Constructor GUI where the file writer configuration file used in the recent LARMOR tests of a LoKI detector.

Figure 4 shows how to create a chopper component using the NXdisk_chopper NeXus class. In this example the disk chopper has 2 slits, 1.0 m radius, 0.6 m slit height and the slit edges are located at (0.0, 60.0) and (180.0, 240.0) degrees. The information in the NeXus definition can be used to produce a rendering of the chopper, which is shown in the instrument view. This view can be used to compare it with CAD drawings and is equivalent to the information needed to drive an instrument simulation.

The NeXus Constructor provides several options to model instrument components with arbitrary geometries, such as detectors (of type NXdetector). The options are: box, cylinder and mesh geometries. For simple detector geometries, it is often sufficient to use repeated cylindrical shapes to define detector pixels or voxels. In case a box or cylinder array is not sufficient, one can provide a mesh geometry. This enables the NeXus Constructor user to define any type of geometry as long as it can be translated into an OFF-file format.

Time-dependent data, such as example chopper speed or location of some device in the instrument, cannot be statically stored in the JSON. Instead, the data has to be retrieved from the control system. The JSON template enables this via file writer stream modules to define where and how dynamic data is to be used and saved.

In summary, the main benefits of the NeXus Constructor are that it displays both the NeXus configuration structure and a 3D instrument rendering with geometric properties (e.g. choppers, detectors, monitors and slits) and that the definitions of NeXus base classes are integrated in the application for reference.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

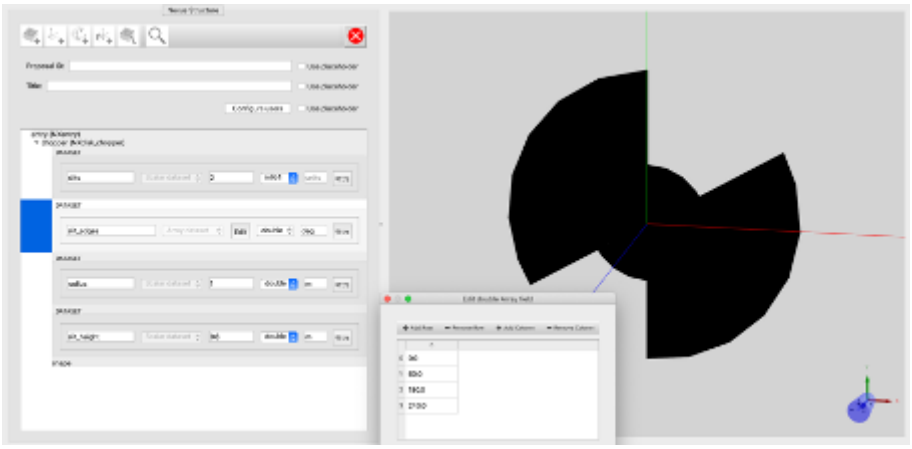

Figure 4: The creation of a NXdisk_chopper in the NeXus constructor. It is added on the top level in the NeXus tree structure, under the entry group, which is of type NXentry.

Clarifications for NeXus International Advisory Committee (NIAC)

Two main changes to the NeXus format have been formulated during the course of the PaNOSC project. They have been submitted to the NeXus International Advisory Committee (NIAC) for review and accepted. One change was directly related to the NeXus Constructor work and sought a clarification of the spatial origin of 29 different device types known to NeXus (Github pull request: https://github.com/nexusformat/definitions/pull/999). The change added the reference location to be used in geometric transformations. Without that additional reference location, it was not possible to unambiguously place (and orient) a larger 3D object in a 3D point.

The second addition concerned the use of "image_key" which was known in NeXus only in the NXtomo application definition, but was not part of a base class (GitHub pull request: https://github.com/nexusformat/definitions/pull/994). This has been fixed in order to address similar use cases as NXtomo in the same way.

Local Clarifications and Assumptions

The need for an interpretation is an unfortunate truth. The reason for this is partially historical, as the NeXus definitions have evolved over time, whereas files from older versions are still used and even produced. The second reason lies in unclear and/or incomplete formulations in the NeXus description that leave things open to interpretation or require reading between the lines. With files being produced by many facilities, it cannot be assumed that they all follow the same interpretation of the rules. It is difficult to create a set of effective and clear guidelines that can help in marking a file as incorrect and avoid misinterpretation. This section represents the current ESS interpretation as used by scipp NeXus (https://github.com/scipp/scippnexus).

NXroot

The NX_class attribute for the top-level of a NeXus hierarchy should be NXroot. However, this has been forgotten in the reference implementation and thus most file-writers do not include it. It is, therefore, to be considered implicit.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Relation between NXdata and NXevent_data with "subclasses"

NeXus does not formally specify subclassing but according to vague hints in the manual several classes are "similar" to NXdata. This includes NXdetector, NXlog, and NXmonitor, but there are likely more. By extension, NXevent_data may take the role of NXdata for event-mode monitors or detectors.

It is unclear whether NeXus requires fields and attributes of the base class (NXdata or NXevent_data) as part of the subclass (such as NXdetector), or whether NXdata or NXevent_data should be stored as a child. We have observed both approaches in practice. Therefore, we go with the following interpretation:

1. *Both* approaches are permitted.
2. For a concrete group in a NeXus hierarchy, there must not be more than one NXdata or NXevent_data child (or field thereof) within the group. If there is more than one, the group is considered invalid.
3. Certain classes pre-define aspects that also NXdata could deliver. For example, NXdata uses the signal attribute to define the field providing the signal array. For NXdetector NeXus pre-defines that its signal is data, so if that is found it will be used, even if no signal attribute is present.

More concretely this means that, e.g., for loading an NXdetector from a NeXus hierarchy, the implementation will:

1. Find all NXdata children.
2. Find all NXevent_data children.
3. Search the group for fields defined in NXdata. This includes looking for the signal attributes on the group and on fields.
4. Search the group for fields defined in NXevent_data.
5. Search the group for fields pre-defined in the class that are equivalents of what is defined in NXdata, even if the NXdata requirements (such as signal attributes) are not met.

If the above yields no more than one item, the group can be loaded.

Datetime fields and their offsets

NXlog and NXevent_data specify specific attributes for fields that have to be interpreted as date and time, in particular NXlog/time@start and NXevent_data/event_time_offset@offset. HDF5 does not directly support storing date and time information such as the Python (Numpy) np.datetime64. Therefore we search *all* attributes of a field for a date and time offset, provided that the field's unit is a time unit. It is unclear what should be done in the case of multiple matches. As of April 2022, we ignore the date and time offsets in this case, since guessing the correct one does not seem desirable.

Bin edges

For NXdetector, the NeXus defines a time_of_flight field, exceeding the data shape by one, i.e., it is meant as bin-edges. NXdata does not appear to allow this explicitly. Since what is recorded in NXdetector may not actually be time-of-flight, in practice this coordinate may be named differently, e.g., time_offset. Therefore, we assume that this is valid in general, i.e., also for other axis values (axis tick labels) that may be defined

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

using the axes attribute. This is also reportedly used in some fields/applications of NX_data.

Missing axis labels

NX_data uses the axes attribute to define the names of fields that store coordinates for axes. There is a legacy mechanism where the signal field has an axes attribute and this should not be used according to the standard.

The axes attribute uses '.' to define an axis without values, i.e., without a field. The implication of the above is that there is no way to define the *label* of an axis, unless values are also defined. For example, an NXdata group storing a stack of images may not define a coordinate for the "image" dimension of the signal dataset. NeXus does not specify any other way to define such an axis name. We therefore have to fall back to a generic and meaningless dimension label.

Note that the axes attribute of the signal field could in principle be used to define such axis labels. The axes attribute of the group would then define which axis has a corresponding field with values. That is, both attributes would work in conjunction, but this is not considered valid currently.

XFEL

NXmx Gold Standard

The so-called NXmx gold standard (Bernstein et al. 2020[1]) is an extension of NeXus application definitions for crystallographic metadata, adopting descriptions from the domain-specific imgCIF/CBF dictionary. The main purpose of these specifications is the full annotation of raw diffraction data, i.e., detector images so that the complete diffraction geometry can be derived from pixel positions and crystallographic data processing software can in principle be run automatically on these raw data.

The NXmx Gold Standard standard has been designed to cover all crystallographic experiment categories including those performed at FEL facilities, namely serial femtosecond crystallography (SFX, PaNET ontology identifier: http://purl.org/pan-science/PaNET/PaNET01168).

European XFEL, having added this experimental technique as a tag for the open subset of the metadata catalogue, developed a prototype workflow for provision of NeXus/NXMx master files from raw diffraction data in HDF5 format, a detector geometry description, and additional metadata as required by the object application definitions, most importantly photon energy. Moreover, FEL pulse-resolved values for photon flux are incorporated into the NXmx master file as a supplementary data set. The master file is written in the HDF5 file format.

One of the challenges with typical FEL image data is the modular assembly of the Megahertz pixel detectors such as the AGIPD-1M (Adaptive Gain Integrating Pixel Detector), where pixel positions can be defined as a sum of translation vectors from the detector centre via quadrant centre to the centre of a module and finally the origin (corner position) of an ASIC tile as smallest unit of the hierarchical topology.

The master file creation procedure was implemented in Python, based on a CCTBX (Computational Crystallography Toolbox) framework template, with all necessary extensions and adaptations to handle the

1 Bernstein, Herbert J. et al. (2020). *Gold Standard for macromolecular crystallography diffraction data*. **IUCrJ 7** (5), 784-792. https://doi.org/10.1107/S2052252520008672

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

AGIPD-1M geometry and the total pulse flux data as obtained from an Xray gas monitor (XGM) data source. Interaction-point-to-detector distance and photon energy are given by command-line arguments, while other required application definitions are hardcoded. When going from prototype to a production version, it is planned to replace this by ingestion of metadata from a toml-file or equivalent configuration, that an experimentalist (in case the instrument operator) can provide - and/or ultimately from the actual entry in the metadata catalogue. The resulting code has been committed to the CCTBX project on GitHub: https://github.com/cctbx/cctbx_project/blob/master/xfel/euxfel/agipd_cxigeom2nexus.py.

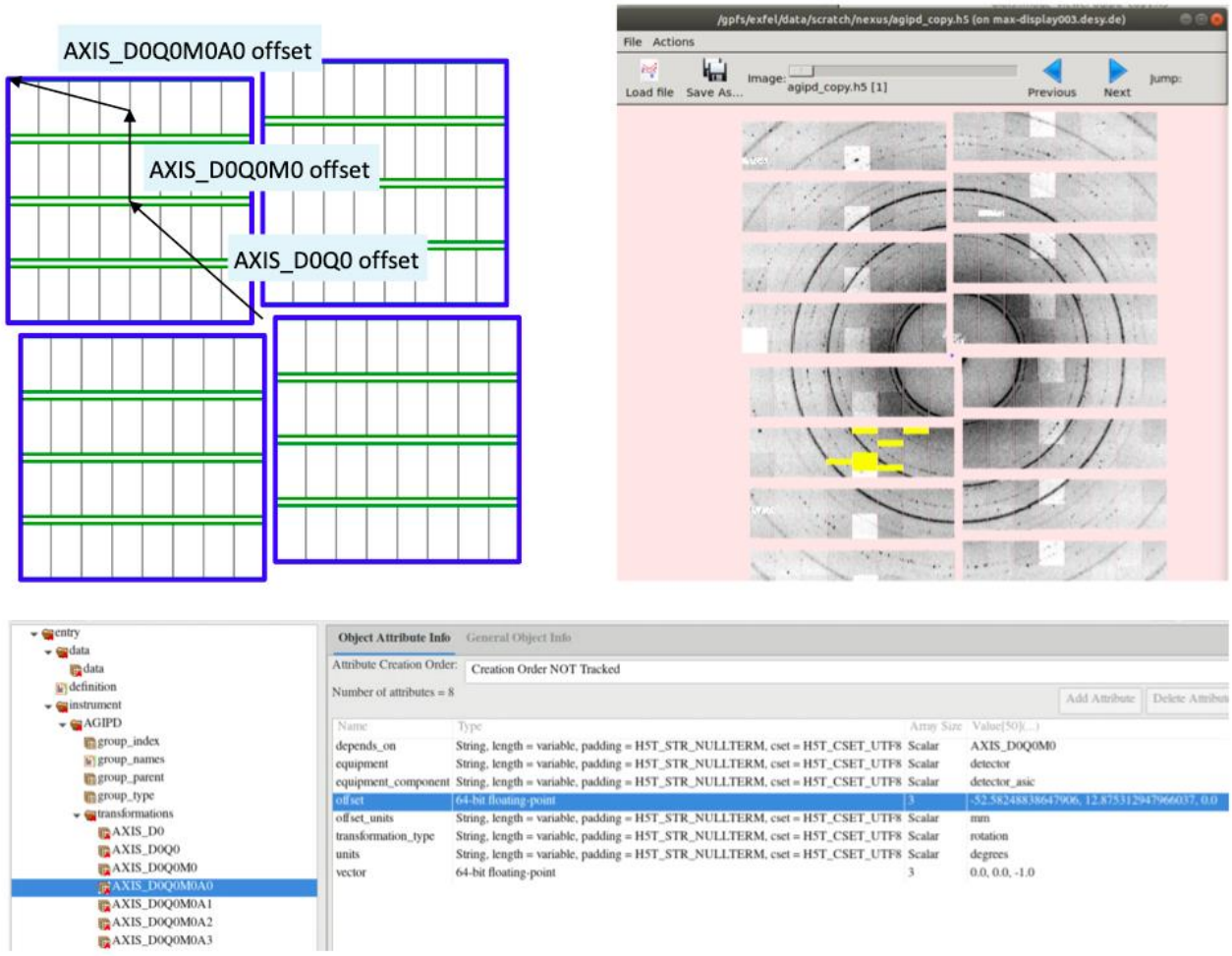

Fig. 5: Schematic hierarchy of the AGIPD-1M detector (top left) including the topological transformations to obtain pixel positions as per NXmx definition, with example highlighted in the HDF5 navigation tool HDFview (bottom). A resulting detector image with a powder diffraction pattern displayed with the DIALS image viewer (top right).

The produced NeXus/NXmx master file contains references (links) to the datasets of detector image frames in order to reduce file size. A self-contained data collection from EuXFEL thus requires both the master file and (access to) the original image data. Given these prerequisites, the outcome of the prototype fulfils two success verification criteria:

- the detector image data can be displayed readily with the DIALS image viewer through the master file, and is thus compliant to actual data processing with this package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

- the master file passes validation with nxvalidate tool, proving that all required application definitions are contained

European XFEL (https://www.xfel.eu/), has 6 scientific instruments, which are performing experiments since late 2017. It is planned to expand the NeXus descriptions of metadata to other experimental techniques, ideally covering all the instruments.

ESRF

There are more than 40 beamlines currently performing experiments at the ESRF. Their data and metadata are archived and stored. The ESRF aims at full transition towards NeXus compliance for raw and processed data in order to make the data reusable and shareable.

Raw Data

With the introduction of the *Bliss* acquisition control system (https://bliss.gitlab-pages.esrf.fr/bliss/master/) all raw data is saved in NeXus compliant HDF5 files. The *NeXus base classes* (https://manual.nexusformat.org/classes/base_classes/) are used for storing data and metadata from detectors and other beamline components. User defined information like sample description and default plots complement the raw data. (https://bliss.gitlab-pages.esrf.fr/bliss/master/data/data_nutshell.html#data-structure).

Data Processing Input

The NeXus base classes allow representing a wide variety of data and metadata. This flexibility is not convenient for data processing software. The NeXus standard introduced *application definitions* to solve this problem (https://manual.nexusformat.org/classes/applications). These classes describe the required and optional input for technique specific data processing. The fields in those application definitions merge information from scan data stored in NeXus base classes. The *HDF5* file format provides soft links, external links and virtual datasets to achieve this without duplicating data.

As opposed to the NeXus base classes, application definitions are still very much a work in progress. The ESRF assists its scientists in defining NeXus application definitions for all techniques used at the ESRF beamlines.

Currently only the *NXtomo* application definition for tomography data processing is actively supported and maintained by the ESRF (https://manual.nexusformat.org/classes/applications/NXtomo.html#nxtomo) in the form of the *nxtomomill* library (https://tomotools.gitlab-pages.esrf.fr/nxtomomill/). It creates *HDF5* files with NXtomo application definitions based on raw data stored using the NeXus base classes or legacy formats like *EDF* or *dxfile*.

nxtomomill provides a library to define and edit *NXtomo* converters and standalone applications. More information can be found in the standalone tutorials, notebook to create an NXtomo from scratch or general training notebook.

The ESRF maintained tomography tools (including *nabu* [http://www.silx.org/pub/nabu/doc/] and *tomwer* [http://www.edna-site.org/pub/doc/tomwer/latest/]) accept *NXtomo* as the default input format.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

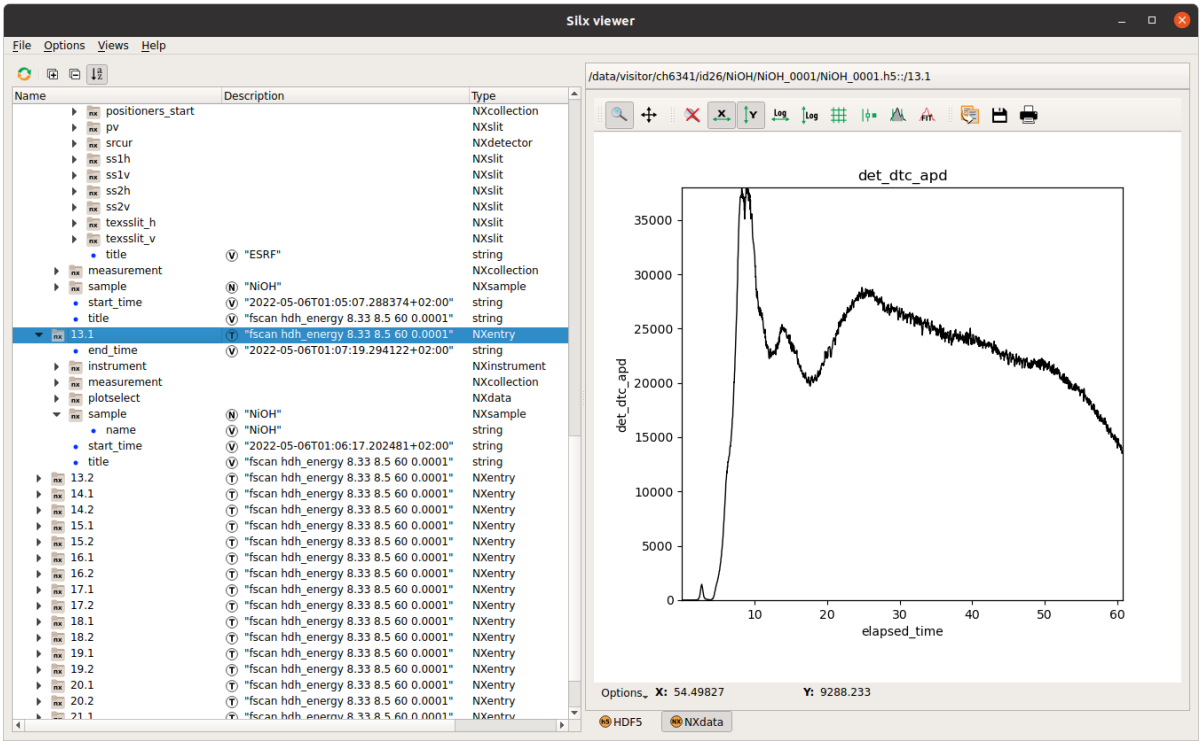

Figure 6: Example of raw data produced by Bliss at ID26 (XANES scan).

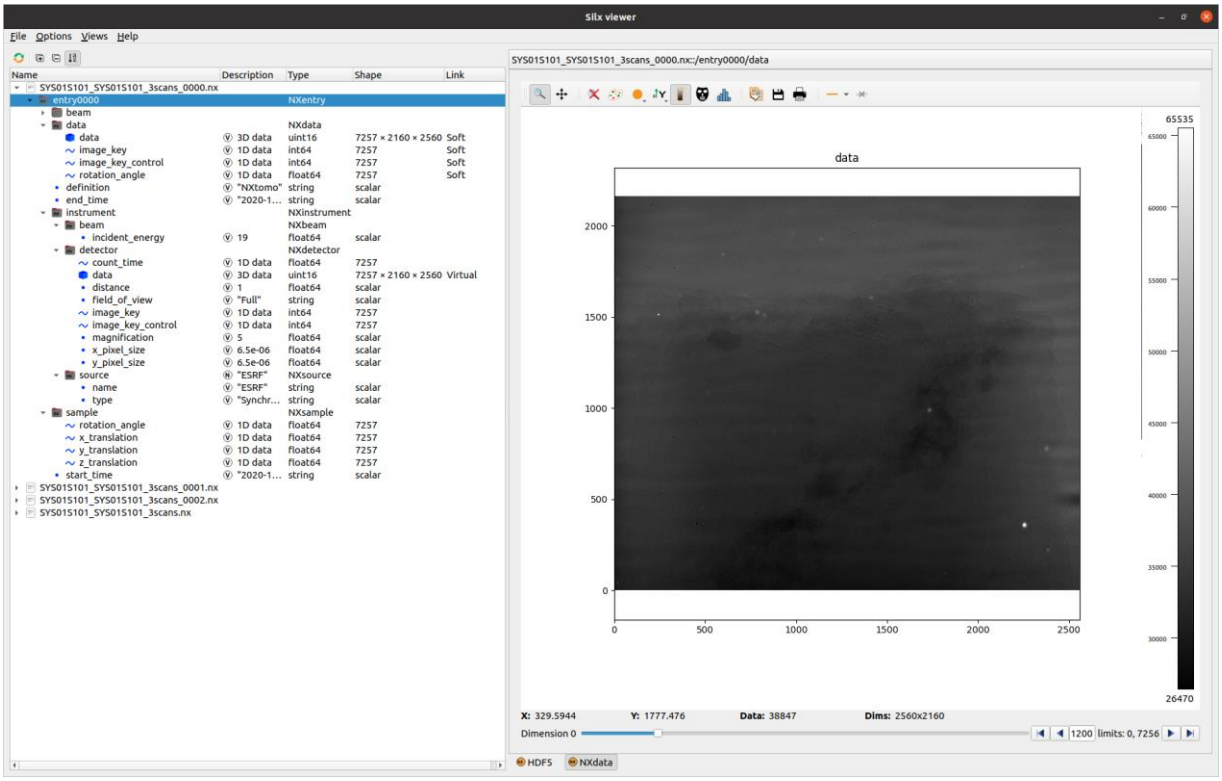

Figure 7: Example of an NXtomo application definition from ID19.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

entry: (required) NXentry

title: (optional) NX_CHAR

start_time: (optional) NX_DATE_TIME

end_time: (optional) NX_DATE_TIME

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXtomo`

instrument: (required) NXinstrument

SOURCE: (optional) NXsource

type: (optional) NX_CHAR

name: (optional) NX_CHAR

probe: (optional) NX_CHAR

Any of these values: `neutron` | `x-ray` | `electron`

detector: (required) NXdetector

data[nFrames, xSize, ySize]: (required) NX_INT

image_key[nFrames]: (required) NX_INT

In order to distinguish between sample projections, dark and flat images, a magic number is recorded per frame. The key is as follows:

- projection = 0
- flat field = 1
- dark field = 2
- invalid = 3

x_pixel_size: (optional) NX_FLOAT {units=NX_LENGTH}

y_pixel_size: (optional) NX_FLOAT {units=NX_LENGTH}

distance: (optional) NX_FLOAT {units=NX_LENGTH}

Distance between detector and sample

x_rotation_axis_pixel_position: (optional) NX_FLOAT

y_rotation_axis_pixel_position: (optional) NX_FLOAT

sample: (required) NXsample

name: (required) NX_CHAR

Descriptive name of sample

rotation_angle[nFrames]: (required) NX_FLOAT {units=NX_ANGLE}

In practice this axis is always aligned along one pixel direction on the detector and usually vertical. There are experiments with horizontal rotation axes, so this would need to be indicated somehow. For now the best way for that is an open question.

x_translation[nFrames]: (optional) NX_FLOAT {units=NX_LENGTH}

y_translation[nFrames]: (optional) NX_FLOAT {units=NX_LENGTH}

z_translation[nFrames]: (optional) NX_FLOAT {units=NX_LENGTH}

control: (optional) NXmonitor

data[nFrames]: (required) NX_FLOAT {units=NX_ANY}

Total integral monitor counts for each measured frame. Allows a to correction for fluctuations in the beam between frames.

data: (required) NXdata

data: link (suggested target: `/NXentry/NXinstrument/detector:NXdetector/data`)

rotation_angle: link (suggested target: `/NXentry/NXsample/rotation_angle`)

image_key: link (suggested target: `/NXentry/NXinstrument/detector:NXdetector/image_key`)

Figure 8: The NXtomo application definition as described in the NeXus format manual (https://manual.nexusformat.org/classes/applications/NXtomo.html#nxtomo).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Support Legacy Software

Some legacy tools do not support *HDF5* as input. At the ESRF this mainly applies to software for processing spectroscopy data. The *nxtoascii* library allows converting NeXus *HDF5* files produced by *Bliss* to different ASCII formats (https://gitlab.esrf.fr/spectroscopy/nxtoascii).

Data Processing Output

Data processing often involves applying a chain of different tools where the output of one tool becomes the input of the following tool. The intermediate results can be stored in NeXus compliant *HDF5* files to allow chaining tools which are not aware of each other, provided those tools support NeXus as input and output. The NeXus standard provides base classes to store generic data processing results but if those results should be used as input for other tools, application definitions are required as well.

Similar to application definitions for raw data, application definitions for processed data are very much a work in progress. The ESRF provides support for its beamline and software developers that want to define new application definitions. Currently the *NXtress* application definition is being developed by the EASI-STRESS consortium (https://easi-stress.eu) for stress and strain analysis of crystalline material.

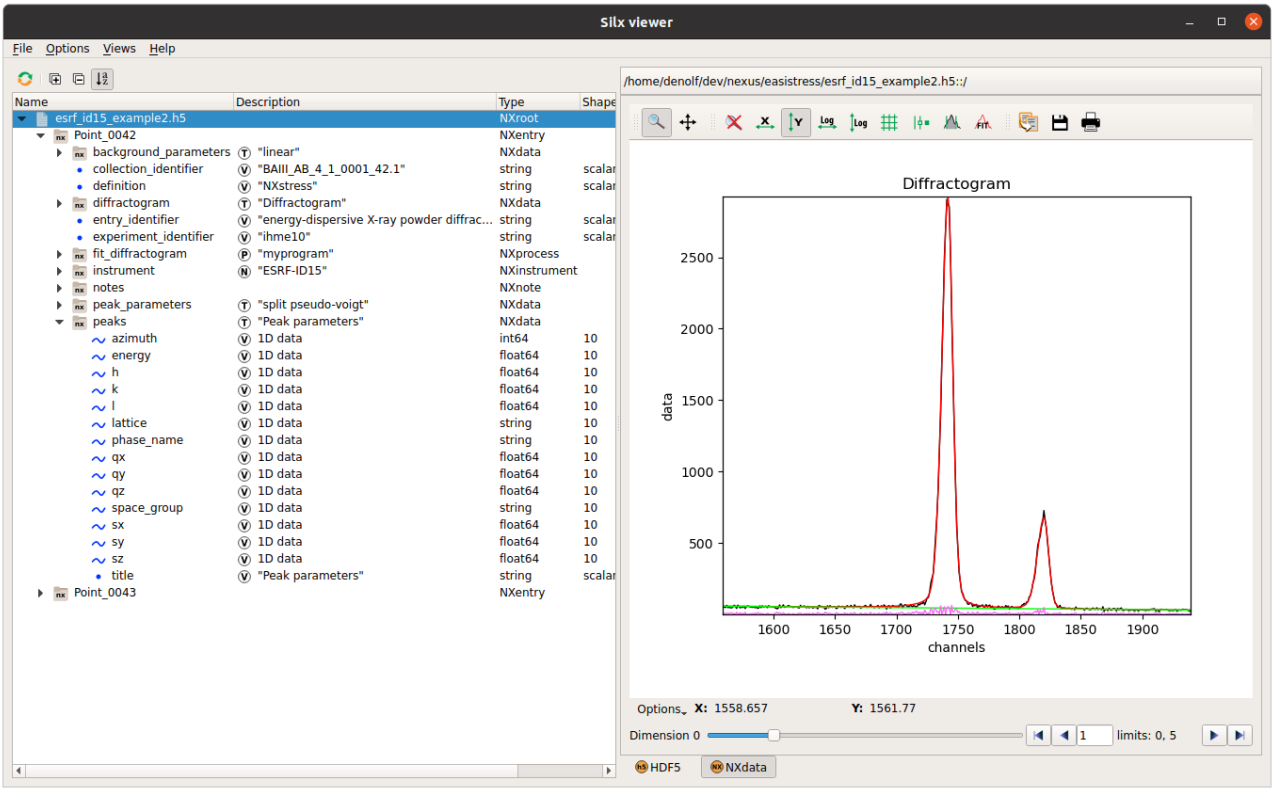

Figure 9: Example of an NXtomo application definition from ID15.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

ENTRY: (required) NXentry

definition: (required) NX_CHAR

Official NeXus NXDL schema to which this file conforms

Obligatory value: `NXstress`

title: (optional) NX_CHAR

Extended title for the entry.

peaks: (required) NXdata

This group contains all diffraction peak parameters that could be needed for stress-strain calculations. These parameters are derived from /NXstress/ENTRY/peak_parameters-group and additional metadata.

h[nPeaks]: (required) NX_INT {units=NX_UNITLESS}

First Miller index.

k[nPeaks]: (required) NX_INT {units=NX_UNITLESS}

Second Miller index.

l[nPeaks]: (required) NX_INT {units=NX_UNITLESS}

Third Miller index.

lattice[nPeaks]: (optional) NX_CHAR

Crystal lattice systems (*cubic*, *hexagonal*, ...)

space_group[nPeaks]: (optional) NX_CHAR

Crystallographic space group (*Fm3m*, *Im3m*, ...)

phase_name[nPeaks]: (optional) NX_CHAR

Name of the crystallographic phase (hematite, goethite, a-Al2O3, ...).

qx[nPeaks]: (required) NX_NUMBER {units=NX_DIMENSIONLESS}

First component of the *normalized* scattering vector *Q*.

qy[nPeaks]: (required) NX_NUMBER {units=NX_DIMENSIONLESS}

Second component of the *normalized* scattering vector *Q*.

qz[nPeaks]: (required) NX_NUMBER {units=NX_DIMENSIONLESS}

Third component of the *normalized* scattering vector *Q*.

scattering_angle[nPeaks]: (optional) NX_NUMBER {units=NX_ANGLE}

Scattering angle 2θ. Is required when /NXstress/ENTRY/peaks/energy-field and /NXstress/ENTRY/peaks/momentum_transfer-field are missing.

energy[nPeaks]: (optional) NX_NUMBER {units=NX_ENERGY}

Is required when /NXstress/ENTRY/peaks/scattering_angle-field and /NXstress/ENTRY/peaks/momentum_transfer-field are missing.

momentum_transfer[nPeaks]: (optional) NX_NUMBER {units=NX_WAVENUMBER}

Length of the scattering vector *Q*. Is required when /NXstress/ENTRY/peaks/energy-field and /NXstress/ENTRY/peaks/scattering_angle-field are missing.

sx[nPeaks]: (required) NX_NUMBER {units=NX_LENGTH}

First component of the sample position.

sy[nPeaks]: (required) NX_NUMBER {units=NX_LENGTH}

Second component of the sample position.

sz[nPeaks]: (required) NX_NUMBER {units=NX_LENGTH}

Third component of the sample position.

Figure 10: The NXstress application definition as described in the NeXus format manual (in preparation).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Data Visualisation on the Web

Development work to support the NeXus format in H5Web (DOI: 10.5281/zenodo.6458452), the web-based HDF5 file viewer and JupyterLab extension, is ongoing. H5Web already supports finding plottable data when opening a file or selecting a group (i.e. by finding the file's default group or the group's signal and axes datasets), displaying meaningful plots (title, axes values and labels, units), and choosing the best visualisation based on the NeXus interpretation (spectrum, image, or rgb-image) or signal dataset dimensions.

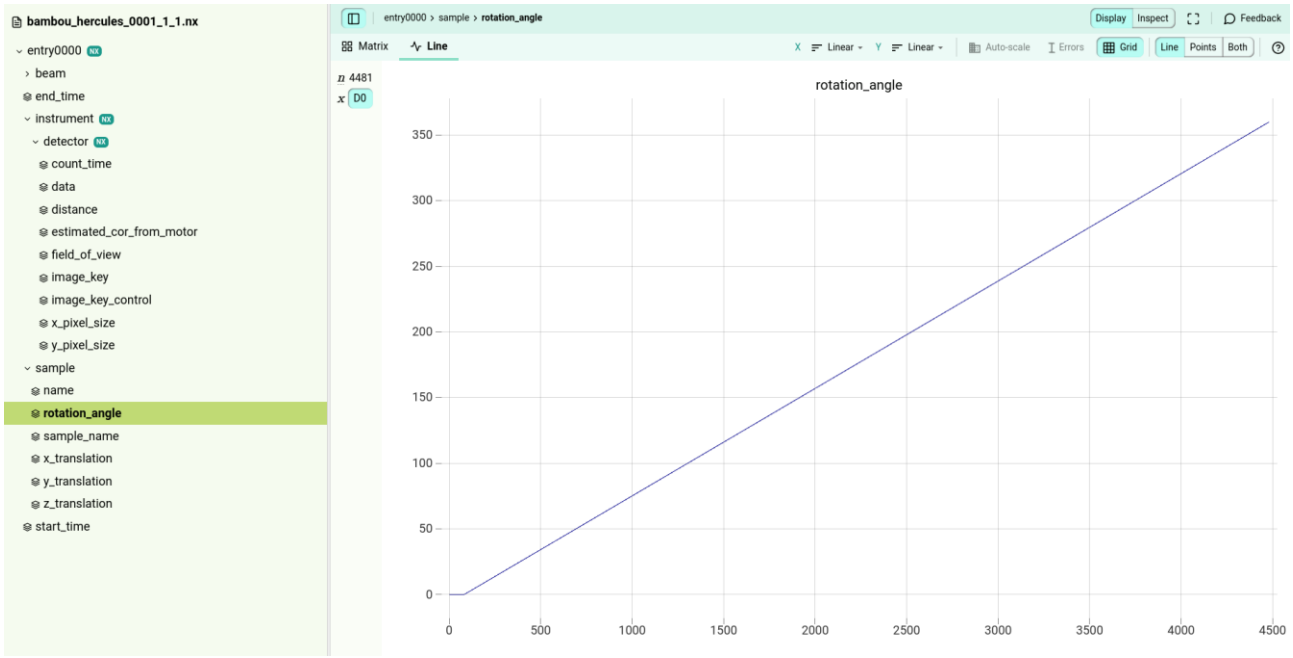

Figure 11: NXtomo entry displayed from h5web.

Recent releases brought the following additional features and improvements: scatter visualisation; legend for auxiliary signals in tooltip; fallback to NeXus v2 algorithm when searching for plottable data.

H5Web has been successfully integrated in a number of web applications which need to display NeXus formatted data. At the ESRF these include (1) the data portal displays NeXus data directly in the portal, and (2) Braggy the web tool used to display macromolecular crystallography data during the experiments. Outside the ESRF other institutes have adopted H5Web e.g. (1) the NOMAD database of FAIR materials, (2) The Diamond Light Source has deployed H5Web for displaying NeXus formatted data, and (3) recently the NIST institute in Washington has adopted H5Web and integrated it in h5wasm and web application for displaying Nexus files.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

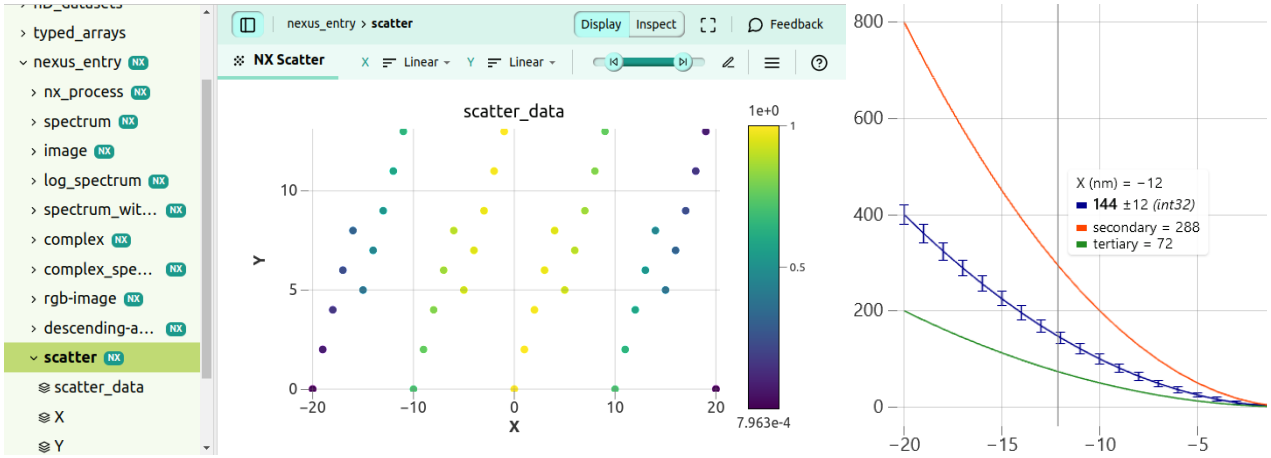

Figure 12: Recent developments in H5Web (from left to right): NeXus scatter visualisation; tooltip with legend in NeXus spectrum visualisation.

ILL

Introduction

Over the past 50 years, the ILL has provided scientists with a very high flux of neutrons feeding some 40 state-of-the-art instruments, which are constantly being developed and upgraded. Historically the ILL supported an ASCII format which was very efficient for compression algorithms (> 90%) and hence ideal for limited storage resources. With increased storage capacity the requirement for compression of ASCII has become less important whereas the interoperability of data has increased greatly.

Therefore, over the past decade or more a transition to the NeXus format has been in progress at the ILL. Currently, 75% of the ILL instruments are writing data in the NeXus format and this transition is ongoing. This transition is complex since many legacy data analysis and data reduction applications have to be updated to support the new format.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Instrument metadata entries and application support

The instruments using NeXus format all have a similar data files structure (Fig 13)

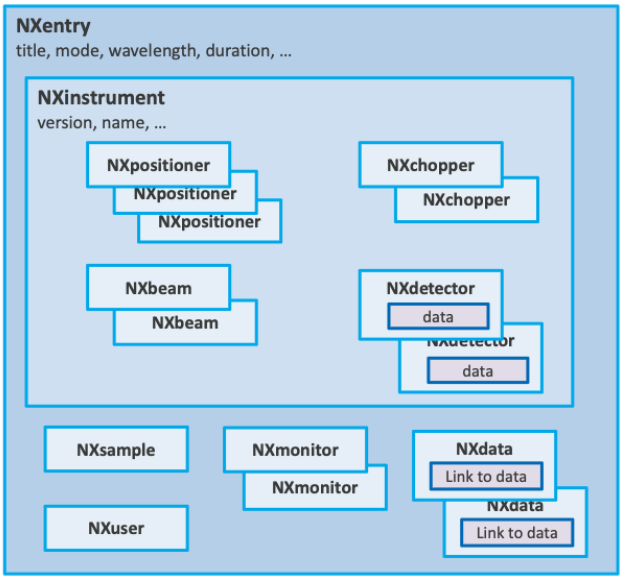

Fig. 13: Data structure shared across ILL instruments using NeXus file format

The metadata entries vary greatly from instrument to instrument and even for a particular instrument, different acquisition modes can produce different metadata entries. For example, trying to consistently extract the wavelength of the incoming neutron beam is not always possible through a standard entry. Table in fig. 14 shows some of the metadata entries and their presence across some the instruments:

Metadata	Type	Description	in4	in5	in6	in16b	d7	d9	d10	d11	d17	d19	d22	d33	d50	brisp	figaro	saisa
Entry																		
title/experiment_title	char	Title	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
subtitle/sample_description	char	Subtitle				■				■			■	■		■		
start_time	date	Start time	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
end_time	date	End time	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
experiment_identifier	char	Exp Id/Sub title/exp number	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
run_number	int32	File numor	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
wavelength	float32	Wavelength	■	■	■	■		■	■		■	■	■			■		■
Reactor power	float32					■				■			■	■				
duration	float32	Count duration	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
mode	char	Acquisition mode	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
modestring	char					■	■	■	■	■	■	■	■		■	■	■	■
acquisition_mode	int32	Acquisition mode		■	■	■	■			■	■		■		■	■	■	
preset	float32	Wanted count time	■	■	■	■	■	■	■		■	■	■	■		■	■	■
inhibit_time	float32	Inhibit time	■			■		■	■		■	■	■			■	■	
actual_time/time	float32	Count duration	■			■		■	■		■	■	■		■	■	■	■
instrument_name	char	Instrument name								■			■	■				

Fig. 14: Metadata entries across instruments at ILL

Since both Nexus and HDF5 are highly supported, reading the ILL Data files is much simpler now than several years ago. The problem exists though that since the metadata entries vary greatly, a generic means of interpreting the data is not an easy task.

As a consequence, each instrument has its own integration into specific data analysis and processing applications such as LAMP (Large Array Manipulation Program, developed specifically for processing ILL data) and the Mantid data analysis and manipulation package. The instrument scientists will show ILL users how to process their data using these software packages and often support the users in the manipulation of their data after they have left the ILL.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

CERIC-ERIC

Introduction

The adoption of the NeXus format at CERIC-ERIC has to face the heterogeneity of the instruments to which the consortium provides access. These differences concerning acquisition systems, techniques, data processing and analysing needs and so on sometimes can represent an obstacle or make it impossible to adopt this format. This is the case of NMR which is a nuclear spectroscopic technique that cannot be fitted in the NeXus format, or proprietary acquisition systems that don't allow the integration of other data formats than the one for which they have been designed or even analysis and process software which require a specific format file as input.

Multiple approaches

Due to this heterogeneity, the NeXus format is being integrated when possible using multiple approaches like extending the original data file with an NXentry group, this may be possible for labs and beamlines using acquisition systems or analysis software that handle HDF files.

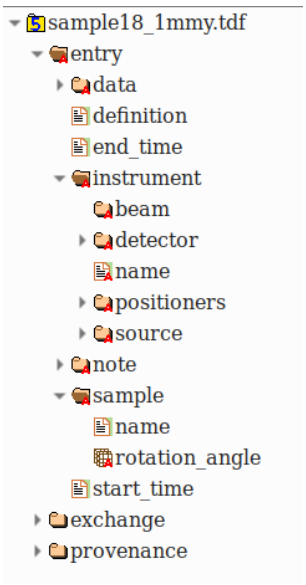

Figure 15: NeXus integrated in a "TDF" file (based on HDF format)

Another approach that has been followed is to adapt the acquisition system to produce NeXus files. In this case, the possibility of acting directly or indirectly in the development of the acquisition system is required. Moreover, the analysis software also has to be changed or adapted somehow. Furthermore, in order to raise awareness on NeXus format and mainly to collect metadata in a more structured way, converters have been developed. In those cases, only the metadata contained in the original file are saved into the NeXus file.

All of these approaches involve the scientists in charge of the instruments, who have to choose the NeXus application more suitable for their measurements.

In CERIC two applications are currently being used: NXtomo and NXmonopd, while NXscan is in the process of being adopted.

The NeXus files produced by the instruments contain the mandatory NeXus entities, together with all other entities required by the leading scientists which have been added following the proper class hierarchy of the application to which they refer.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Developments

A Python module, so-called “2nexus”, has been developed in order to simplify the adoption of the NeXus format by the scientists, and to make it simple for the development of converter tools.

This module was integrated into the acquisition system of an XRD beamline by the scientist who developed the system; the acquisition workflow now includes the production of an NXmonopd file.

The module is based on a system of hierarchical YAML files that map generic or instrument-based parameters to some NeXus classpaths, it is also possible to specify default values or default units that are suitable for a specific instrument measure type (Fig. 16).

```
# List of all parameters that can be passed to the NXApp in order to create a NeXus file
# N:B: The parameters are agreed with the scientist responsible for the specific experiment
meta_parameters:
  # Parameter item format:
  # param_name: {nx_path: '<NX element meta-path>',  nx_links: [<NX element meta-path>,],   def_value: <default value>,   def_units: <default units>,   dependency_func: '<function>'}
  #
  # "nx_path":  Define the hierarchical path of the NX field or attribute which "param_name" refers
  #             The NX classes that composed the path can be written in these formats:
  #             1) ../NX<class_name>/.. (e.g. NXentry/NXsample/distance)
  #             2) ../name:NX<class_name>/.. (e.g. NXentry/sample_01:NXsample/distance)
  #             In the first case the name of the NX class will be <class_name> itself

  #
  inst_name:        {nx_path: 'NXentry/NXinstrument/name',         nx_links: [],   def_value: 'Materials Characterisation by X-ray diffraction',   def_units: '',   dependency_func: ''}
  inst_tag:         {nx_path: 'NXentry/NXinstrument/name/short_name',   nx_links: [],   def_value: 'MCX',def_units: '',   dependency_func: ''}
  crystal_usage:    {nx_path: 'NXentry/NXinstrument/NXcrystal/usage',   nx_links: [],   def_value: 'Bragg',   def_units: '',   dependency_func: ''}
  crystal_type:     {nx_path: 'NXentry/NXinstrument/NXcrystal/type',   nx_links: [],   def_value: 'Si',   def_units: '',   dependency_func: ''}
  crystal_chem_form: {nx_path: 'NXentry/NXinstrument/NXcrystal/chemical_formula',   nx_links: [],   def_value: '',   def_units: '',   dependency_func: ''}
  crystal_reflect:  {nx_path: 'NXentry/NXinstrument/NXcrystal/reflection',   nx_links: [],   def_value: [ 1, 1, 1 ],   def_units: '',   dependency_func: ''}
  crystal_wavelen:  {nx_path: 'NXentry/NXinstrument/NXcrystal/wavelength',   nx_links: [],   def_value: '',   def_units: 'angstrom',   dependency_func: ''}
  detect_descr:     {nx_path: 'NXentry/NXinstrument/NXdetector/description',   nx_links: [],   def_value: '',   def_units: '',   dependency_func: ''}
  detect_dist:      {nx_path: 'NXentry/NXinstrument/NXdetector/distance',   nx_links: [],   def_value: '',   def_units: 'mm',   dependency_func: ''}
  polar_angle:      {nx_path: 'NXentry/NXinstrument/NXdetector/polar_angle',   nx_links: ['NXentry/NXdata/polar_angle'],   def_value: '',   def_units: 'deg',   dependency_func: ''}
  data:             {nx_path: 'NXentry/NXinstrument/NXdetector/data',   nx_links: ['NXentry/NXdata/data'],   def_value: '',   def_units: '',   dependency_func: ''}
  count_time:       {nx_path: 'NXentry/NXinstrument/NXdetector/count_time',   nx_links: [],   def_value: '',   def_units: 's',   dependency_func: ''}
  samp_name:        {nx_path: 'NXentry/NXsample/name',   nx_links: [],   def_value: '',   def_units: '',   dependency_func: ''}
  samp_situation:   {nx_path: 'NXentry/NXsample/situation',   nx_links: [],   def_value: '',   def_units: '',   dependency_func: ''}
  samp_temp:        {nx_path: 'NXentry/NXsample/temperature',   nx_links: [],   def_value: '',   def_units: 'K',   dependency_func: ''}
  samp_dist_z:      {nx_path: 'NXentry/NXsample/distance',   nx_links: [],   def_value: '',   def_units: 'mm',   dependency_func: ''}
  samp_rotat:       {nx_path: 'NXentry/NXsample/rotation_angle',   nx_links: [],   def_value: '',   def_units: 'deg',   dependency_func: ''}
  monit_mode:       {nx_path: 'NXentry/NXmonitor/mode',   nx_links: [],   def_value: 'monitor',   def_units: '',   dependency_func: ''}
  monit_samp_frac:  {nx_path: 'NXentry/NXmonitor/sampled_fraction',   nx_links: [],   def_value: 1.0,   def_units: '',   dependency_func: ''}
  monit_data:       {nx_path: 'NXentry/NXmonitor/data',   nx_links: [],   def_value: '',   def_units: '',   dependency_func: ''}
  monit_sum:        {nx_path: 'NXentry/NXmonitor/integral',   nx_links: [],   def_value: '',   def_units: '',   dependency_func: 'sum(%monit_data%)'}
  monit_count_time: {nx_path: 'NXentry/NXmonitor/count_time',   nx_links: [],   def_value: '',   def_units: 's',   dependency_func: ''}
```

Figure 16: Mapping YAML file

The following example shows how to create a simple nexus files using the NXApp object

```
# Load and prepare NeXus file
mcx_nx = NXApp(nx_filepath, 'elettra', 'mcx', 'monopd')

# Add parameters to NeXus file. All parameters are optional, in the following call all available parameters are passed
# except those parameters that are read from the MCX configuration files placed in conf directory file
mcx_nx.set(title={'value': 'Experiment title'}, inv_name={'value': '2020xxxx'}, start_time={'value': datetime.utcnow()},
           ring_curr={'value': 0.0}, ring_energy={'value': 0.0}, top_up={'value': True},
           crystal_chem_form={'value': 'C12H22O11'}, crystal_wavelen={'value': [0.0, ]},
           detect_descr={'value': 'Detector name'}, detect_dist={'value': 5.0}, polar_angle={'value': polar_ang},
           data={'value': data}, count_time={'value': count_time}, samp_name={'value': 'Sample name'},
           samp_situation={'value': "air"}, samp_temp={'value': -10.0, 'units': 'C°'}, samp_rotat={'value': [0.0, 45.0, 90.0]},
           monit_data={'value': ion_chamber_data}, monit_count_time={'value':ion_chamber_count_time})
```

Figure 17: Code snippet showing how to use the “2nexus” module

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Critical issues

A critical point for upgrading from custom HDF5 or even other formats to specific NeXus application definition is convincing beamline staff and visiting scientists that there is a substantial advantage. Such an advantage may come from services like data-cataloguing and analysis code that are compatible with NeXus and not with HDF5. For now, such cases are very limited and do not justify the investment of changes in the acquisition systems and existing data analysis workflows. The limited use of NeXus today is due to legacy reasons based on adopting technical solutions that were more suitable in the past.

ELI

Introduction

The ELI facilities, built as individual construction projects are now coming together as an integrated organisation. The two ELI Facilities (ELI Alps and ELI Beamlines) are now preparing for the integration to operate under a single ERIC, the Extreme Light Infrastructure ERIC (ELI ERIC).

Considering the objectives of ELI ERIC and the FAIR commitment expressed by ELI ERIC via its FAIR Data Policy[2], the two facilities are now starting to prepare for the first user experiments.

As a next and natural step in the FAIR Data Policy implementation process, the ELI Attosecond Light Pulse Source (ELI-ALPS) facility, a unique attosecond facility providing ultrashort light pulses between THz (1012 Hz) and X-ray (1018-1019 Hz) frequency range with high repetition rate for developers and end-users, is now working together with ELI ERIC in the implementation and development of FAIR Data Pilot Projects for its beamlines and scientific instruments. These FAIR Data Pilot projects are impacted by the fact that the facility is still in the implementation phase and therefore it does not have fully-fledged User experiments. However, the facility offers scientific access for commissioning Users and aims at using the timing advantage and the experience of the first ELI ERIC Users to develop FAIR Users' data and data-related processes.

Developments

ELI-ALPS has already installed research instruments as standalone systems. They are partially integrated with the facility infrastructure and one of these pieces of equipment is the NanoESCA. The NanoESCA is a state-of-the-art powerful tool to study the electronic structure of ordered solid surfaces. The equipment supports multiple measurement modes and we were focusing on the most used one: the momentum microscopy mode.

The manufacturer's software does not have built-in support for NeXus and it exports the data files of an experiment as TIFF and configuration files. Based on a discussion with the instrument scientists, a decision was made to use the NXscan application definition as a starting point to have NeXus support for kinetic energy scanning with the equipment.

The following mapping were applied to create a NeXus file from the original data files:

- *Title:*
  The name attribute from the configuration file for the measurement is used as the title.
- *Start and end times:*

2 https://zenodo.org/record/6515903#.YoN-TpNBwUo

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

The configuration files have different timestamp attributes, but these are related to the creation of the configurations itself, so instead of these timestamps, the timestamps from the TIFF files are used to determine the start and the end time.

- *User:*
  This field is filled using the name of the operator from the configuration file of the measurement.
- *Sample:*
  The configuration file of the measurement has an attribute to define the sample. The angle values are defined as 0 for each step of the scan.
- *Monitor:*
  The electron volt values are used to define the data of the "monitor". These values are extracted by using the 'AXIS_START', 'AXIS_STOP' and 'AXIS_STEP' attributes.
- *Instrument:*
  The instrument definition has a detector. The detector holds the raw data loaded from the TIFF files.
- *Data:*
  The manufacturer's software creates more than one image per scan step and during the measurement it exports not only the raw but the averaged data as well. The TIFF files of the averaged data are used to fill the data field of the NeXus file.
- *Collection:*
  The collection field is used to store the original configuration files of the measurement.

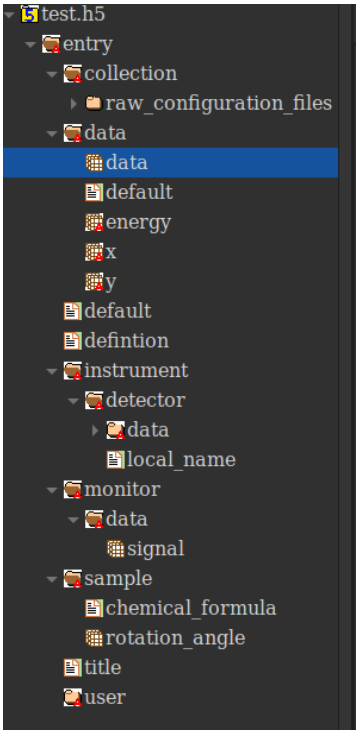

Figure 18: Example of a measurement output from NanoESCA as an NXScan

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Our current experience applying the NeXus application definition to the output of the NanoESCA equipment has shown it will be worth it to create a NanoESCA specific application definition, once we have more input from the Users related to expected content of such a file.

.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Summary

The work package has had a positive impact on the NeXus adoption of the partner sites. The "*Show and Tell*" sessions were very useful as an overview specifically for the "newer" facilities, such as ESS, ELI and XFEL. Especially the ILL could offer a lot of experience in recording different techniques in somewhat similar ways.

It was clear early on that using NeXus terms for formulating search queries would offer little benefit, if it could be made to work at all. But this exercise also provided good insights into the working of NeXus to the partners.

XFEL, ESS and ELI have made or prepared individual contributions to NeXus definitions. So, the work package had an "upward" effect on the NeXus community. However, at least the same amount of work was dedicated to tools and local integrations that are shared as open-source projects. The software can be re-used to better write NeXus files with a higher degree of compliance (nexus constructor for example) and commonality or to better visualise and process files using common code (h5web and scipp for example). This is a benefit to the photon and neutron community as a whole.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852