NeXus – Metadata for Neutron and Photon Science

Tobias Richter

European Spallation Source

Workshop on Research Data Management
Helmholtz Zentrum Berlin, June 2019

EUROPEAN SPALLATION SOURCE

ESS Status

NeXus International Advisory Committee NIAC

- Mark Basham,
  Diamond Light Source, UK
  (Executive Secretary)
- Herbert Bernstein,
  CIF (non-facility member)
- Aaron Brewster,
  Lawrence Berkeley Laboratory, USA
- Stuart Campbell,
  Brookhaven National Laboratory, USA
  (Technical Manager)
- Bjorn Clausen,
  Los Alamos National Laboratory, USA
- Stephen Cottrell,
  Rutherford Appleton Laboratory, UK
- Ricardo Ferraz-Leal,
  SNS and HFIR at ORNL, USA
- Jens-Uwe Hoffmann,
  Helmholtz Zentrum Berlin, Germany
- Pete Jemian,
  Advanced Photon Source, USA
  (Documentation Release Manager)
- Mark Könnecke,
  Paul Scherrer Institute, Switzerland
- Raymond Osborn,
  Argonne National Laboratory, USA
  (non-facility member)
- Tobias Richter,
  European Spallation Source, Sweden
- Armando Sole,
  European Synchrotron Radiation Facility,
  France
- Jiro Suzuki,
  KEK, Japan
- Benjamin Watts,
  Swiss Light Source, Switzerland (Chair)

NeXus

NeXus provides a format that can hold:

- **raw experimental data (with all information required for processing)**
- **all data needed for diagnostics**
- **metadata**
- **processed data**

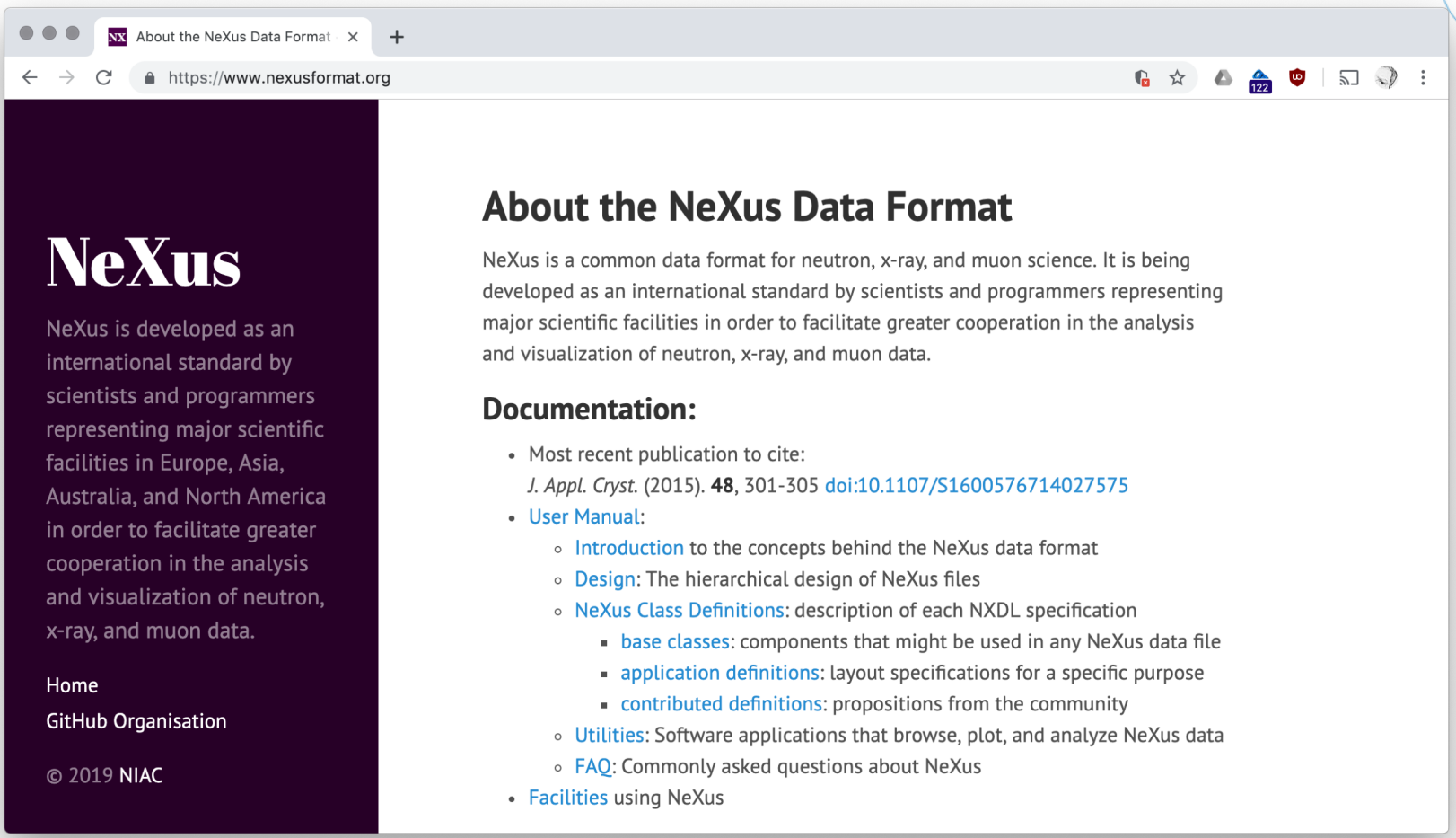
All of the above are optional.

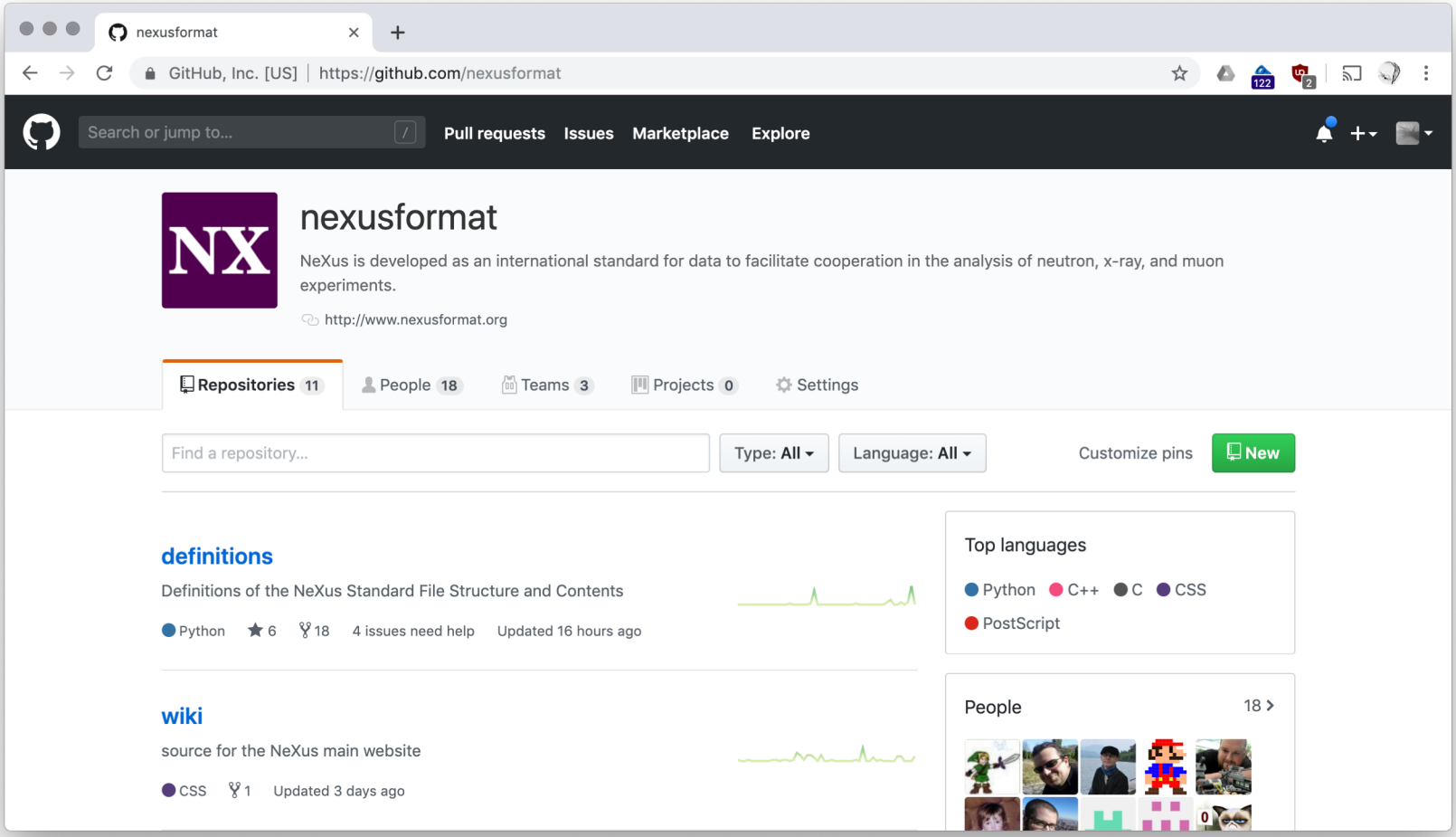
For all techniques at

- **neutron**
- **muon**
- **X-ray**
- **soft X-ray**
- **VUV**

research facilities.

Goal is to replace any formats that require implicit knowledge about the experiment.


About the NeXus Data Format
https://www.nexusformat.org
NeXus
NeXus is developed as an international standard by scientists and programmers representing major scientific facilities in Europe, Asia, Australia, and North America in order to facilitate greater cooperation in the analysis and visualization of neutron, x-ray, and muon data.
Home
GitHub Organisation
© 2019 NIAC
About the NeXus Data Format
NeXus is a common data format for neutron, x-ray, and muon science. It is being developed as an international standard by scientists and programmers representing major scientific facilities in order to facilitate greater cooperation in the analysis and visualization of neutron, x-ray, and muon data.
Documentation:
Most recent publication to cite:
J. Appl. Cryst. (2015). 48, 301-305 doi:10.1107/S1600576714027575
User Manual:
Introduction to the concepts behind the NeXus data format
Design: The hierarchical design of NeXus files
NeXus Class Definitions: description of each NXDL specification
base classes: components that might be used in any NeXus data file
application definitions: layout specifications for a specific purpose
contributed definitions: propositions from the community
Utilities: Software applications that browse, plot, and analyze NeXus data
FAQ: Commonly asked questions about NeXus
Facilities using NeXus


nexusformat
GitHub, Inc. [US] | https://github.com/nexusformat
Search or jump to...
Pull requests
Issues
Marketplace
Explore
nexusformat
NeXus is developed as an international standard for data to facilitate cooperation in the analysis of neutron, x-ray, and muon experiments.
http://www.nexusformat.org
Repositories 11
People 18
Teams 3
Projects 0
Settings
Find a repository...
Type: All
Language: All
Customize pins
New
definitions
Definitions of the NeXus Standard File Structure and Contents
Python 6 18 4 issues need help Updated 16 hours ago
wiki
source for the NeXus main website
CSS 1 Updated 3 days ago
Top languages
Python C++ C CSS
PostScript
People 18 >

What's in the box?

We have:

- Definition of terms for experiment equipment, sample and parameters
- Rules for expressing scans and geometries and various other relationships
- Agreement on required terms for various purposes
- A validation tool (cnxvalidate)
- Not to forget: A community of users

We also have:

- A legacy library (NAPI) and related inspection tools

Levels of NeXus adoption

EUROPEAN SPALLATION SOURCE

At a formal level:

1. Data stored in HDF5 container
2. NeXus base classes followed (use of dictionary terms)
3. Making guarantees about the file contents
   * Following application definitions
   * Advertising recipes for data extraction (“features”)

Obviously there is also the amount a quality of the information that goes into the file, which is hard to measure.

What does HDF5 do for you?

EUROPEAN SPALLATION SOURCE

- Offers a useful structured (hierarchical) storage container supporting many datatypes.
- Saves a lot of work accessing data randomly in many dimensions (less then 32)
- Has lots of clever features that make organising and writing data fast, efficient and flexible.
  - Custom compression filters
  - SWMR
  - Virtual Dataset
- HDF5 is well and widely supported - h5py, MATLAB, Igor Pro,
  however NAPI support for newer HDF5 features is lacking.

- As a rule:
  Don’t solve problems that HDF5 has already solved for you.

Base Classes

EUROPEAN SPALLATION SOURCE

Contain parameters common for particulars type of equipment or sample, user, etc.

```
base_classes$ ls
NXaperture.nxdl.xml                 NXdetector_module.nxdl.xml      NXlog.nxdl.xml                  NXsample.nxdl.xml
NXattenuator.nxdl.xml               NXdisk_chopper.nxdl.xml         NXmirror.nxdl.xml               NXsensor.nxdl.xml
NXbeam.nxdl.xml                     NXentry.nxdl.xml                NXmoderator.nxdl.xml            NXshape.nxdl.xml
NXbeam_stop.nxdl.xml                NXenvironment.nxdl.xml          NXmonitor.nxdl.xml              NXslit.nxdl.xml
NXbending_magnet.nxdl.xml           NXevent_data.nxdl.xml           NXmonochromator.nxdl.xml        NXsource.nxdl.xml
NXcapillary.nxdl.xml                NXfermi_chopper.nxdl.xml        NXnote.nxdl.xml                 NXsubentry.nxdl.xml
NXcharacterization.nxdl.xml         NXfilter.nxdl.xml               NXobject.nxdl.xml               NXtransformations.nxdl.xml
NXcite.nxdl.xml                     NXflipper.nxdl.xml              NXorientation.nxdl.xml          NXtranslation.nxdl.xml
NXcollection.nxdl.xml               NXfresnel_zone_plate.nxdl.xml   NXparameters.nxdl.xml           NXuser.nxdl.xml
NXcollimator.nxdl.xml               NXgeometry.nxdl.xml             NXpinhole.nxdl.xml              NXvelocity_selector.nxdl.xml
NXcrystal.nxdl.xml                  NXgrating.nxdl.xml              NXpolarizer.nxdl.xml            NXxraylens.nxdl.xml
NXdata.nxdl.xml                     NXguide.nxdl.xml                NXpositioner.nxdl.xml           nxdlformat.xsl
NXdetector.nxdl.xml                 NXinsertion_device.nxdl.xml     NXprocess.nxdl.xml
NXdetector_group.nxdl.xml           NXinstrument.nxdl.xml           NXroot.nxdl.xml
```

With these you can build up a hierarchy describing a fairly complete description of an experiment.

Application Definitions

EUROPEAN SPALLATION SOURCE

Aim is to guarantee the file contents (compiled from base class definitions) as a kind of contract between producer and consumer.

NXarchive.nxdl.xml	NXmonopd.nxdl.xml	NXsqom.nxdl.xml	NXtomoproc.nxdl.xml	NXxnb.nxdl.xml
NXarpes.nxdl.xml	NXmx.nxdl.xml	NXstxm.nxdl.xml	NXxas.nxdl.xml	NXxrot.nxdl.xml
NXcanSAS.nxdl.xml	NXrefscan.nxdl.xml	NXtas.nxdl.xml	NXxasproc.nxdl.xml	canSAS
NXdirecttof.nxdl.xml	NXreftof.nxdl.xml	NXtofnpd.nxdl.xml	NXxbase.nxdl.xml	nxdlformat.xsl
NXfluo.nxdl.xml	NXsas.nxdl.xml	NXtofraw.nxdl.xml	NXxeuler.nxdl.xml	
NXindirecttof.nxdl.xml	NXsastof.nxdl.xml	NXtofsingle.nxdl.xml	NXxkappa.nxdl.xml	
NXiqproc.nxdl.xml	NXscan.nxdl.xml	NXtomo.nxdl.xml	NXxlaue.nxdl.xml	
NXlauetof.nxdl.xml	NXspe.nxdl.xml	NXtomophase.nxdl.xml	NXxlaueplate.nxdl.xml	

Like the base classes they are defined via XML/XSD schema files with custom documentation elements that produce part of the NeXus manual.

Because one size does not fit all, modern application definitions carry optional entries. This and additional rules (dimensions, scans, naming) can weaken the useful contract.

Features and Recipes

- Every rule and exception puts a burden on developers of NeXus consuming software.
- Reading every scheme in active use needs to be implemented and tested for every data processing or analysis application.
- This is a downside from NeXus aiming to be flexible enough for all kinds of raw data.
- Developing precise enough documentation can be harder than writing the code to extract the information.

- Recently NeXus files can advertise “Features” that are linked to a Recipe, readable code written in Python, that extracts the associated information correctly. Serves as documentation and reference implementation.
- Features can be small and can work like a unit test for data file and processing code.
- Downside: Recent adoption, not fully mature.

Locations and Orientations

depends_on (Field or Attribute)
locates and orients components and is used to chain transformations corresponding to their physical setup

Transformations (Field with Attributes)
describe the dynamic or static placement of components with `@transformation_type, @vector, @depends_on, @units, @offset, etc`

Nxtransformations (Group)
used to group transformations, for example to have axes on one diffractometer together

```
entry:NXentry
  data:NXdata
    …
  instrument:NXinstrument
    detetctor:NXdetector
      …
  sample:NXsample
    depends_on=diffr/phi
    diffr:NXtransformations
      phi[…]
        @transformation_type=rotation
        @vector=0,1,0
        @depends_on=chi
      chi[…]
        @transformation_type=rotation
        @vector=0,0,1
        @depends_on=rotation_angle
      rotation_angle[…]
        @transformation_type=rotation
        @vector=0,1,0
```

NeXus Features / Recipes Example

EUROPEAN SPALLATION SOURCE

```
class recipe:
        """
                A demo recipe for finding the information associated with this demo feature.

                This is meant to help consumers of this feature to understand how to implement
                code that understands that feature (copy and paste of the code is allowed).
                It also documents in what preference order (if any) certain things are evaluated
                when finding the information.
        """

        def __init__(self, filedesc, entrypath):
                self.file = filedesc
                self.entry = entrypath
                self.title = "CIF-style sample geometry"

        def findNXsample(self):
                for node in self.file[self.entry].keys():
                        try:
                                absnode = "%s/%s" % (self.entry, node)
                                if self.file[absnode].attrs["NX_class"] == "NXsample":
                                        return absnode
                        except:
                                pass
                # better have custom exceptions
                raise Exception("no NXsample found")

        def process(self):
                dependency_chain = []
                try:
                        sample = self.findNXsample()
                        # this may need more attention for reading all possible types of string
                        depends_on = self.file[sample+"/depends_on"][0]
                        while not depends_on == ".":
                                dependency_chain.append(depends_on)
                                # this may need more attention for reading all possible types of string
                                depends_on = self.file[depends_on].attrs["depends_on"]

                except Exception as e:
                        raise Exception("this feature does not validate correctly: "+e)

                        # better have custom exceptions
                return { "dependency_chain" : dependency_chain }
```

Different Rules for Metadata

Different ways of recording scans is
an example where extracting
simple metadata information like:

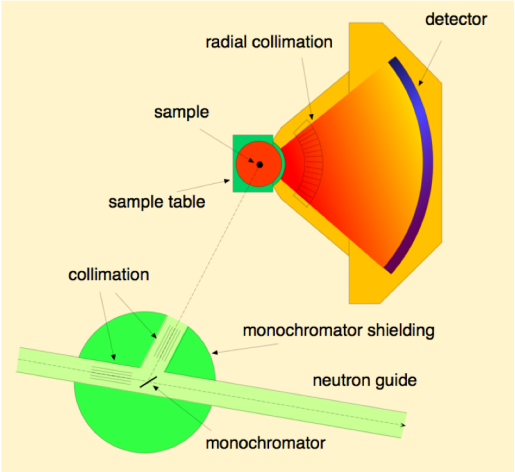
“What was the scan range?”

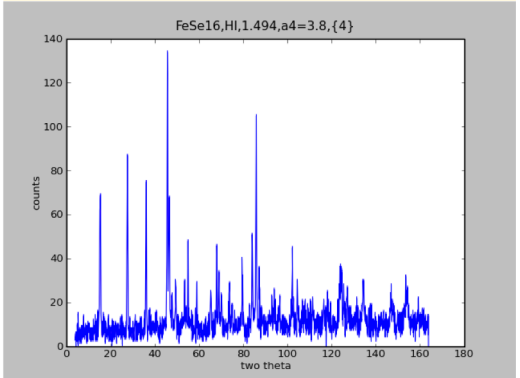
is dependent on NeXus rules
outside of application definitions.

Static Exposure

“sit and count” – old school

```
entry:NXentry
  data:NXdata
     data[1024]
     two_theta[1024]
  instrument:NXinstrument
    detector:NXdetector
       data[1024]
```

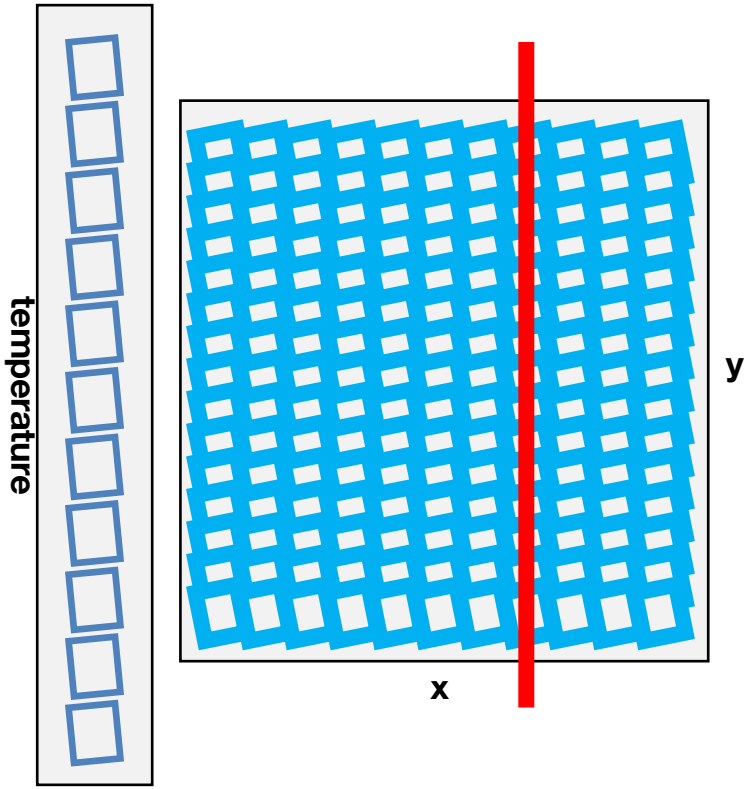




This is what most NeXus application definitions specify.

Pros	Cons
Easy	Very limiting

Multi dimensional scan data




```
entry:NXentry
  data:NXdata
     data[11,1024,1024,512]
     x_scan[1024]
     y_scan[1024]
     energy[1024,1024,512]
     temperature[11]
```

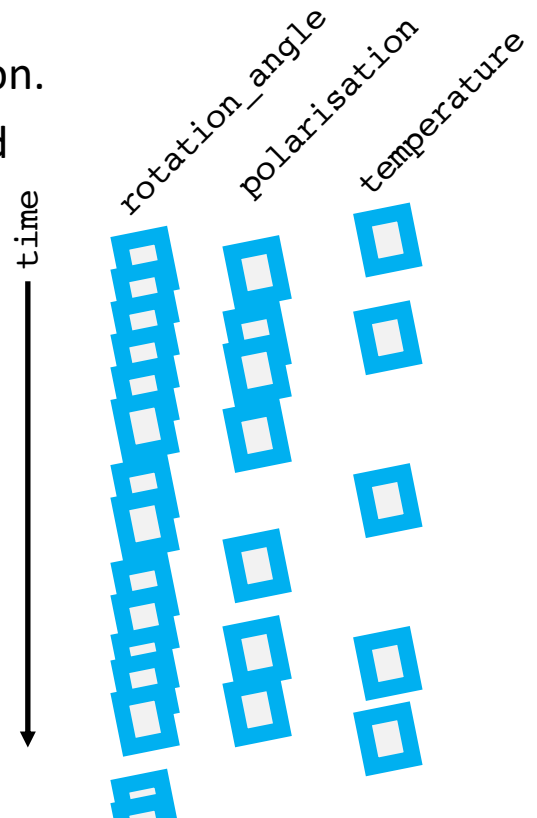
Pros	Cons
Easy slicing and plotting, captures scan intend	Limited to rectangular geometries

Hidden in the examples:
Scheme for defining the alignment of indices.

Asynchronous Scan Recording

- **NXlog** Group for asynchronous time stamped data can replace any dataset in a base class or application definition.
- Fits well with neutron event recording, data only updated on change.

```
entry:NXentry
  data:NXdata
     data:NXlog
     polarisation:NXlog
     temperature:NXlog
     rotation_angle:NXlog
```



Pros	Cons
very flexible, efficient storage, little DAQ hardware support needed	No default plot, requires post processing

In an ideal world…

Some consumers would like to read data in the same way and do not care in which of the supported ways the experiment was recorded:

- Static Exposure
- NeXus Scan Rules (1D)
- Multiple Scan Dimensions
- Time Stamp Everything

For metadata cataloguing purposes this could be solved.

PaNOSC Partners

EUROPEAN SPALLATION SOURCE

NEUTRONS FOR SCIENCE

ESRF The European Synchrotron

European XFEL

EGI

panosc

photon and neutron open science cloud

eli

EUROPEAN SPALLATION SOURCE

CERIC Central European Research Infrastructure Consortium

ESFRI European Strategy Forum on ResearchInfrastructures

Meta Ontology for Catalogues

Objective: Explore how we can use the NeXus definitions, rules and infrastructures to enable domain specific searches in data repositories and add relevant new definitions.

PaNOSC Task 3.5

M1-M42

Extend NeXus metadata standards to enhance interoperability. For large parts of the PaN communities NeXus is the most commonly used file format. To explore relevant foreign datasets in the public domain, searches on the scientific metadata need to yield the correct results. Building search terms and keywords from the NeXus dictionary, would make use of the community buy in and expertise that went into this standard.
In this task we can add missing raw data definitions for raw data, as well as some for processed derived data. Working towards an Ontology will help harmonize processing codes between facilities.

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

Get involved!

- http://www.nexusformat.org documentation - “wiki”

- join the open Google Hangout session twice a month

- subscribe to the mailing list

- https://github.com/nexusformat

NeXus