The NeXus Data Format for muon Spectroscopy and Neutron or X-ray Scattering

Mark Könnecke

Paul Scherrer Institute
Switzerland

September 23, 2012

- A different data format wherever she goes

The Predicament of the Traveling Scientist

- A different data format wherever she goes
- Spends lots of time converting formats or writing readers

The Predicament of the Traveling Scientist

- A different data format wherever she goes
- Spends lots of time converting formats or writing readers
- Waits even longer to load data from inefficient data formats

The Predicament of the Traveling Scientist

- A different data format wherever she goes
- Spends lots of time converting formats or writing readers
- Waits even longer to load data from inefficient data formats
- DA requires N files in different formats, notes, local knowledge

The Predicament of the Traveling Scientist

- A different data format wherever she goes
- Spends lots of time converting formats or writing readers
- Waits even longer to load data from inefficient data formats
- DA requires N files in different formats, notes, local knowledge
- Cannot read her collaborators data

NeXus

The Predicament of the Traveling Scientist

- A different data format wherever she goes
- Spends lots of time converting formats or writing readers
- Waits even longer to load data from inefficient data formats
- DA requires N files in different formats, notes, local knowledge
- Cannot read her collaborators data
- Has to keep extra information in yet another form

NeXus Mission

- Definition of a standard data format
  - Rules
  - Validation tools
- Promotion of NeXus
  - Documentation
  - NeXus API
  - Outreach to the scientific community

- Complete data for typical use
- Extendable, add additional data as you please
- Self describing
- Easy automatic plotting
- Platform independent, public domain, efficient
- Suitable for a wild variety of applications

NeXus History

- Devised from three independent proposals by Jonathan Tischler, APS, Przemek Klosowski, NIST and Mark Koennecke, ISIS, PSI in 94-96
- Improved during various NOBUGS conferences
- NeXus International Advisory Committee, NIAC, since 2003
- Since 2003 yearly meetings of the NIAC
- We already considered many issues!
- Except for one year, we never had money to develop NeXus

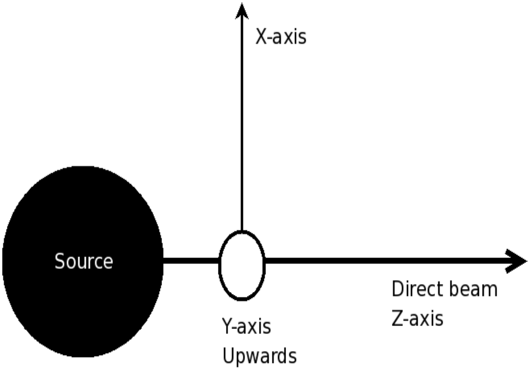
1. Physical file format and API for accessing files
2. Rules for storing data in files
3. Component and application definitions
4. NeXus Utilities

- Portable, self describing, extendable, public domain
- Hierarchical data format 5 (HDF-5)
- HDF-5:
  - grouping support
  - on the fly compression
  - reading/writing subsets
  - first dimension appendable
  - Public domain C, F77 access library
  - Used by: NASA, Boing, Deutsche Bank, the weathermen, ...
- XML for those who wish to edit their data
- For historical reasons we have support for the older version HDF-4

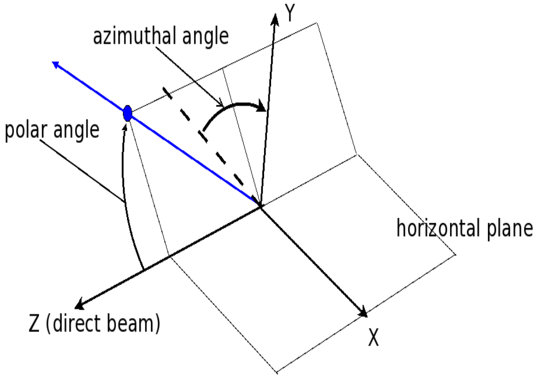
- NeXus-API hides complex HDF API
- Transparent access to all three supported physical file formats
- ANSI-C implementation
- Bindings: C++, F77, Java, python, IDL, SWIG
- January, 4, 2010: 1311217 files processed at PSI alone
- You do not have to use NAPI!
- You can write/read NeXus file with HDF-5 tools and APIs alone

NeXus Objects

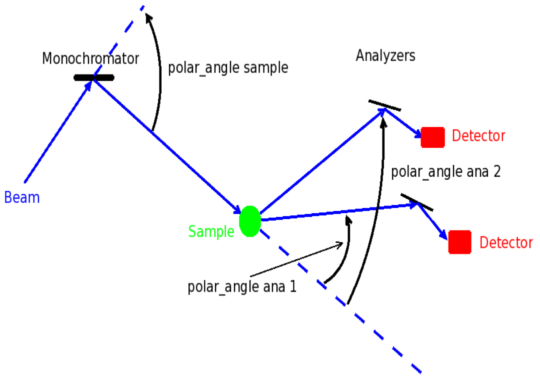
- Files
- Groups identified by name and a classname beginning with NX
- Scientific data sets
- Attributes
- Links


X-axis
Source
Direct beam
Z-axis
Y-axis
Upwards

NeXus Simple Coordinate System



Polar angle is always relative to the previous component



NeXus Transformations

- Learning from imageCIF: be precise enough in axis descriptions to construct transformation matrices
- Allows to calculate absolute positions of components in the laboratory coordinate systems
- Can directly convert from a detector coordinate system to vectors in Lab coordinate system
- Calculate things like impact of primary beam on detector, SAS
- Allows arbitray axis to be expressed
- Intuitively describe an instrument with angles and translations and still be able to recover absolute coordinates
- Full mapping between imageCIF and NeXus now possible

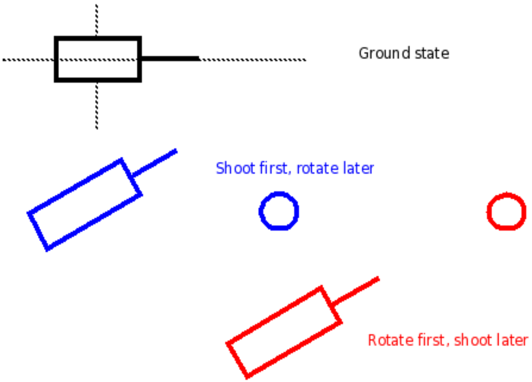
NeXus

$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} r11 & r12 & r13 & 0 \\ r21 & r22 & r23 & 0 \\ r31 & r32 & r33 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Combining Transformations



- Transformations can be combined by matrix multiplications
- Individual matrices can be derived by looking at the situation when everything else is 0
- Absolute positions can be obtained by multiplying the resulting matrix with its transpose
- Defines new coordinate systems at components

type	rotation or translation
offset	an optional translation to apply to the axis before application
direction	vector around which rotated or translated
value	The angle of rotation or the length of translation
dependency	The order of operations to place a component

NeXus Axis Mapped

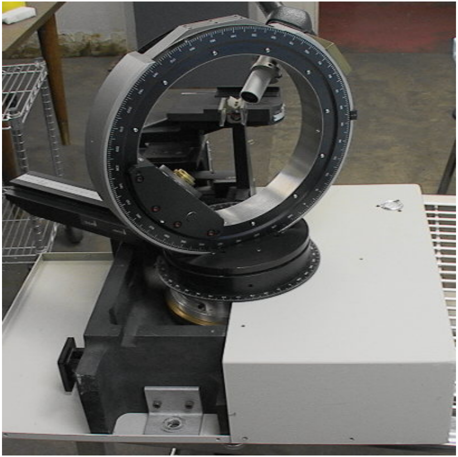
- rotation_angle, polar_angle, rotate 0 1 0
- azimuthal_angle, rotate 0 0 1
- distance, translate 0 0 1
- chi, rotate 0 0 1
- phi rotate, 0 1 0
- NeXus polar coordinate system: rotate azimuthal_angle, rotate polar_angle, translate by distance

- Axis will be annotated with attributes:

vector	The direction around which to rotate or along which to translate
transformation_type	rotation or translation
offset	The offset to apply before the axis
offset_units	The units of offset
depends_on	The name of the previous axis in the dependency chain. Or . when this axis is the bottom of the chain.

- Addionally: a depends_on field per component which gives the name of the top axis in the dependency chain

Example: A Eulerian Cradle



```
sample,NXsample
    rotation_angle
        @vector=0,1,0
        @transformation_type=rotation
    chi
        @depends_on=rotation_angle
        @vector=0,0,1
        @transformation_type=rotation
    phi
        @depends_on=chi
        @vector=0,1,0
        @transformation_type=rotation
    depends_on
        phi
```

- Special group structure which can be added to any base class
- Directly specify engineering coordinates

geometry:NXgeometry
  translation:NXtranslation
    translation[3]
  shape:NXshape
    shape: nxbox|nxcylinder|nxsphere
    size[]
  orientation:NXorientation
    vector[3]

- NeXus reserves the prefix NX for group names.
- Store as much as possible
- A NeXus file has one to many NXentry groups
- There are two types of entries: raw data and processed data
- Multiple different techniques in one file go into separate NXsubentries
- If there is only one entry, the preferred name is entry, else entry1, entry2... entryn
- If an entry or an NXsubentry conforms to an application definition, the application definitions must be stated in the entries definition field.

NeXus Raw Data File Structure

```
entry:NXentry
    sample:NXsample

    instrument:NXinstrument
        source:NXsource
        velocity_selector:NXvelocity_selector
        detector:NXdetector
            data[xsize,ysize], signal=1 (1)
    control:NXmonitor
        data
    data:NXdata
        link to (1)
```

entry:NXentry
- sample:NXsample
- processing_name:NXprocess
  - program
  - version
  - parameters:NXparameter
    - raw_file
- data:NXdata
  - data[nx,ny,nz], signal=1

entry:NXentry
    sample:NXsample
    instrument:NXinstrument
    .....
    sas:NXsubentry
        sample:NXsample

        instrument:NXinstrument
            source:NXsource
            velocity_selector:NXvelocity_selector
            detector:NXdetector
                data[xsize,ysize], signal=1 (1)
        control:NXmonitor
            data
        data:NXdata
            link to (1)

NXcollection

```
entry,NXentry
    measurement:NXcollection
        positions:NXcollection
            om
            two_theta
        scalars:NXcollection
            title
            wavelength
        data:NXdata
            detector1
            mca5
```

Why the Hierarchy??

- Supports self description and allows short names in components

- Supports self description and allows short names in components
- Name, classname pair allows for multiple components of the same type

- Supports self description and allows short names in components
- Name, classname pair allows for multiple components of the same type
- NXentry allows for multiple datasets in the same file

Why the Hierarchy??

- Supports self description and allows short names in components
- Name, classname pair allows for multiple components of the same type
- NXentry allows for multiple datasets in the same file
- NXdata supports automatic plotting

Why the Hierarchy??

- Supports self description and allows short names in components
- Name, classname pair allows for multiple components of the same type
- NXentry allows for multiple datasets in the same file
- NXdata supports automatic plotting
- Hierarchy offers itself naturally for organising data

Why the Hierarchy??

- Supports self description and allows short names in components
- Name, classname pair allows for multiple components of the same type
- NXentry allows for multiple datasets in the same file
- NXdata supports automatic plotting
- Hierarchy offers itself naturally for organising data
- Take care once when writing, use n times

Rules for Storing Data Items

- Store physical values in C storage order
- Use NeXus components and dictionary names
- Missing names will be quickly accepted by the NIAC
- Names: full words separated by _
- Specify units in same format as used by UDunits
- Application definitions may restrict units

- There are situations were data has to be dumped as fast as possible in order to keep up with a high data rate. Or to save disk space.
- **Data not in C storage order:** use attributes stride and offset to describe the memory layout of the data.
- **Data needs scaling:**Use a NXformula group to specify a formula in muParser notations plus the parameters and data necessary to do the scaling.
- Details on both methods will be in the NeXus manual

Associating Axis and Data, Preferred method

```
entry:NXentry
    data:NXdata
        data[nx,ny,nz], signal=1, axes=x_axis,y_axis,z_axis
        x_axis[nx]
        y_axis[ny]
        z_axis[nz]
```

Associating Axis and Data, deprecated method

- Data and axis live in the same NXgroup

```
entry:NXentry
    data:NXdata
        data[nx,ny,nz], signal=1
        x_axis[nx], axis=1
        y_axis[ny], axis=2
        z_axis[nz], axis=3
```

```
entry:NXentry
    data:NXdata
        data[nx,ny,nz], signal=1
        x_axis[nx], axis=1, primary=1
        alternate_x_axis[nx], axis=1
        y_axis[ny], axis=2
        z_axis[nz], axis=3
```

- Preserve original dimensionality of detector, if possible
- Time-of-flight becomes last dimension
- Highly irregular detectors:

```
entry:NXentry
    instrument:NXinstrument
        detector:NXdetector
            data[ndet], signal=1
            polar_angle[ndet], axis=1
            azimuthal_angle[ndet]
            distance[ndet]
```

Scans

- Come in all shapes and sizes
- Captured by rules:
  - Store all varied parameters as arrays of length NP at the appropriate place in the NeXus hierarchy
  - For multi detectors, NP, number of scan points is always the first dimension
  - In NXdata: create links to counts and varied variables

Scan Example 1: rotating sample

entry:NXentry
- sample:NXsample
  - rotation_angle[NP], axis=1 (1)
- instrument:NXinstrument
  - detector:NXdetector
    - data[NP],signal=1 (2)
- control:NXmonitor
  - data[NP]
- data:NXdata
  - link to (1)
  - link to (2)

```
entry:NXentry
    sample:NXsample
        rotation_angle[NP], axis=1 (1)
        phi[NP], axis=1 (2)
        chi[NP], axis=1 (3)
        h[NP], axis=1 (4), primary=1
        k[NP], axis=1 (5)
        l[NP], axis=1 (6)
    instrument:NXinstrument
        detector:NXdetector
            data[NP],signal=1 (7)
            polar_angle[NP],signal=1 (8)
    data:NXdata
        link to (1)
        link to (2)
        link to (...)
        link to (8)
```

Scan Example 3: sample rotation, area detector

- entry:NXentry
  - sample:NXsample
    - rotation_angle[NP], axis=1 (1)
  - instrument:NXinstrument
    - detector:NXdetector
      - data[NP,xsize,ysize],signal=1 (2)
  - control:NXmonitor
    - data[NP]
  - data:NXdata
    - link to (1)
    - link to (2)

- This is rastering a sample at different wavelengths, positions etc.
- Same treatment as scans, NP replaced by NR number of raster points
- For the common case of rastering on a 2D grid one can store [nx,ny,detdim]. Be aware, though, that this causes problems if the rasterisation is aborted in mid operation.

- Component definitions: dictionaries of allowed field names for the various NeXus groups
- Application Definitions
    - Define what has to be in a NeXus file for a certain application
    - Defines standards
    - Another view: Contract between file producers and users about what has to be in a NeXus file for a well defined purpose
    - Validation by NXvalidate
- Written in NeXus Definition Language, NXDL

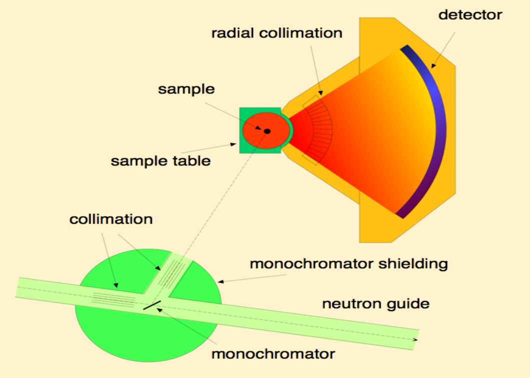
All Base Classes

- NXaperture
- NXattenuator
- NXbeam_stop
- NXbeam
- NXbending_magnet
- NXcharacterization
- NXcollimator
- NXcrystal
- NXdata
- NXdetector
- NXdisk_chopper
- NXentry
- NXenvironment
- NXevent_data
- NXfermi_chopper
- NXfilter
- NXflipper
- NXgeometry
- NXguide
- NXinsertion_device
- NXinstrument
- NXlog
- NXmirror
- NXmoderator
- NXmonitor
- NXmonochromator
- NXnote
- NXorientation
- NXparameters
- NXpolarizer
- NXprocess
- NXsample
- NXsensor
- NXshape
- NXsource
- NXtranslation
- NXuser
- NXvelocity_selector

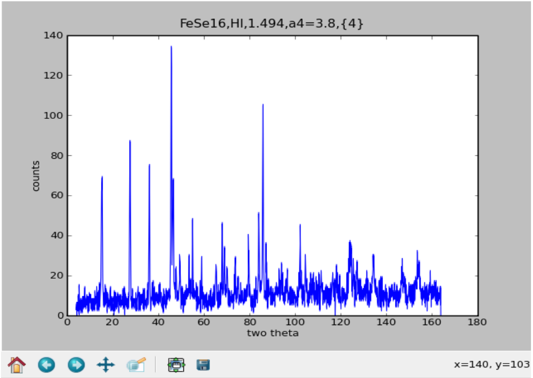
1. Construct an application definition with advice from the NIAC
2. You can also inherit from and extend an existing definition
3. Cure for a year; data should be produced in the new format in this time
4. After curation and review: this is the standard for this application type.

- No promises, but the NIAC may do it for you
  - Description of experiment
  - Minimum set of data items necessary form common use
  - Example data

Example: WONI



WONI Plot



FeSe16,HI,1.494,a4=3.8,{4}
counts
two theta
0
20
40
60
80
100
120
140
160
180
x=140, y=103

1. Think! what ought to go into the file
2. Map this into the NeXus file structure
3. Cast this mapping into a NXDL file
4. Standardize your application definition together with the NIAC

- What has to go into the file?
- Minimum data necessary for common usage scenarios
- Haggle it out with your community
- Coverage ratio: $>$ 80 % of use cases

- Common usage is Rietveld analysis or profile analysis
- Data required:
    - Title
    - Sample name
    - Wavelength
    - Counts versus two_theta
    - Monitor, for normalisation

- Consider into which NeXus group an item might belong
- Look in the base class for a suitable data field
- Link the data items required for the default plot into NXdata

entry:NXentry

- title
- definition

entry:NXentry
- title
- definition
- sample:NXsample
  - name

- entry:NXentry
  - title
  - definition
  - sample:NXsample
    - name
  - instrument:NXinstrument
    - monochromator:NXmonochromator
      - wavelength

entry:NXentry
- title
- definition
- sample:NXsample
  - name
- instrument:NXinstrument
  - monochromator:NXmonochromator
    - wavelength
  - detector:NXdetector
    - data[ndet], signal=1 (1)
    - polar_angle[ndet], axis=1 (2)

```
entry:NXentry
    title
    definition
    sample:NXsample
        name
    instrument:NXinstrument
        monochromator:NXmonochromator
            wavelength
        detector:NXdetector
            data[ndet], signal=1 (1)
            polar_angle[ndet], axis=1 (2)
    control:NXmonitor
        data
```

```
entry:NXentry
    title
    definition
    sample:NXsample
        name
    instrument:NXinstrument
        monochromator:NXmonochromator
            wavelength
        detector:NXdetector
            data[ndet], signal=1 (1)
            polar_angle[ndet], axis=1 (2)
    control:NXmonitor
        data
    data:NXdata
        link to (1)
        link to (2)
```

- The structure defined by the application definition is the minimum

From an application definition to a real NeXus file

- The structure defined by the application definition is the minimum
- Practical files strive to capture much more

From an application definition to a real NeXus file

- The structure defined by the application definition is the minimum
- Practical files strive to capture much more
- Suggested procedure:
  - Look at each of your instruments components and the matching NeXus base class
  - Add whatever you feel like adding or the instrument scientists wants to have
  - Add whatever management wants to have (may be not in a NeXus group)

NeXus

From an application definition to a real NeXus file

- The structure defined by the application definition is the minimum
- Practical files strive to capture much more
- Suggested procedure:
  - Look at each of your instruments components and the matching NeXus base class
  - Add whatever you feel like adding or the instrument scientists wants to have
  - Add whatever management wants to have (may be not in a NeXus group)
- Remember: Adding more fields does not break application definition compliance!

What about ONOKI?

- ONOKI: ONe Of a Kind Instrument

NeXus

- ONOKI: ONe Of a Kind Instrument
- Think! what you want to store

What about ONOKI?

- ONOKI: ONe Of a Kind Instrument
- Think! what you want to store
- Map the list into the appropriate NeXus base classes. It helps to look at each of the components ONOKI is assembled from and to decide what you wish to store for each of them.

What about ONOKI?

- ONOKI: ONe Of a Kind Instrument
- Think! what you want to store
- Map the list into the appropriate NeXus base classes. It helps to look at each of the components ONOKI is assembled from and to decide what you wish to store for each of them.
- The next one to copy ONOKI is well advised to copy what you did NeXus file wise, otherwise she will not be able to reuse your software!

Available NeXus Application Definitions

NXarchive	NXmonopd	NXrefscan
NXreftof	NXsas	NXscan
NXtas	NXtofraw	NXtomo
NXtomophase	NXxeuler	NXxkappa
NXxnb	NXxrot	NXiqproc
NXtomoproc	NXtofsingle	NXdirectof
NXindirectof	NXiqproc	NXlauetof
NXsastof	NXsqom	NXtofraw
NXtofsingle	NXxas	NXxasproc

Challenge 1 in science you are supposed to do new, non standard, things. These of course cannot be easily cast into a standard.

Challenge 1
: in science you are supposed to do new, non standard, things. These of course cannot be easily cast into a standard.

Challenge 2
: in order to establish a standard a lot of people need to agree

Data Format Challenges

Challenge 1 in science you are supposed to do new, non standard, things. These of course cannot be easily cast into a standard.

Challenge 2 in order to establish a standard a lot of people need to agree

Challenge 3 a standard requires scarce scientific programming resources for adoption

Chance 1 By using a discoverable data format like NeXus, XML, HDF-5, people can at least figure out what is in the data file.

Chance 1 By using a discoverable data format like NeXus, XML, HDF-5, people can at least figure out what is in the data file.

Chance 2 Using predefined names from a dictionary gives meaning to the data in a file.

Chance 1 By using a discoverable data format like NeXus, XML, HDF-5, people can at least figure out what is in the data file.

Chance 2 Using predefined names from a dictionary gives meaning to the data in a file.

Chance 3 Using a shared API reduces learning costs and increases application stability.

Data Format Chances

Chance 1 By using a discoverable data format like NeXus, XML, HDF-5, people can at least figure out what is in the data file.

Chance 2 Using predefined names from a dictionary gives meaning to the data in a file.

Chance 3 Using a shared API reduces learning costs and increases application stability.

Chance 4 With NeXus, HDF-5 plus professional programming techniques a DA application can read any file which contains the required data.

NeXus

Chance 1 By using a discoverable data format like NeXus, XML, HDF-5, people can at least figure out what is in the data file.

Chance 2 Using predefined names from a dictionary gives meaning to the data in a file.

Chance 3 Using a shared API reduces learning costs and increases application stability.

Chance 4 With NeXus, HDF-5 plus professional programming techniques a DA application can read any file which contains the required data.

Chance 5 Storing as much data as possible increases the likelihood that the needed data is actually on file, even for unforeseen uses.

Chance 1 By using a discoverable data format like NeXus, XML, HDF-5, people can at least figure out what is in the data file.

Chance 2 Using predefined names from a dictionary gives meaning to the data in a file.

Chance 3 Using a shared API reduces learning costs and increases application stability.

Chance 4 With NeXus, HDF-5 plus professional programming techniques a DA application can read any file which contains the required data.

Chance 5 Storing as much data as possible increases the likelihood that the needed data is actually on file, even for unforeseen uses.

Chance 6 Application Definitions

- Soleil: 20 out of 26 instruments do NeXus, 2 mill files
- PSI-SINQ: 11 from 16 instrument on NeXus, 1.4 Mill files
- Lujan/LANL: 11 instruments, no change, 1 million files
- PSI-SLS: 2 planned,
- KEK: 10, 6 planned
- ANSTO: 7 going to 10
- ESRF: 2 beamlines, limited to NXentry, NXcollection, NXdata, moving to 4
- HZB: 3 Neutron, 1 synchrotron, 3 planned
- SNS: 14,3 in the pipeline
- DESY: 0, 11 in 2 Jahren
- Diamond: 7 NeXus only, 17 writing, moving to 18 as primary format
- ISIS: 8 using, 20 writing, planned: 20 using
- Muons: 4 instruments

1. Store and archive data from a wild variety of instruments
2. Store processed data
3. Store a complete workflow from raw data to publication ready data in several NXentries in one file
4. Store a set of related experiments in one file
5. Define strict and validatable standards

- New systems tend to use NeXus
- No competitor for a general purpose data format
- Planned:
  - DECTRIS will make NeXus/HDF-5 the data format for EIGER detectors
  - Collaboration with CIF
  - Better organisation for base classes
  - Enhance NXvalidate