HDF5, NeXus and beyond: Approach to Standard Data Format

V.A. Solé – ESRF Data Analysis

Q2XAFS 2011

December 8, 2011

Acknowlegdements:

Matthew Newville - CARS, Univ. Chicago

HDF group – http://www.hdfgroup.org

This talk

XA(F)S Data Files

Why a Standard Data format?

Do we need a binary format?

HDF5

NeXus

Conclusion

XAFS Data Files

XAFS data usually come in some flavor of column file

Example XAFS Data File:

```
#sample:    Cu foil Room Temperature
#notes:     data from cu_foil.001
#detectors: I0=N2 10cm; I1=N2 10cm
#beamline:  APS 13ID, vert slits = 0.3mm
#mono:      Si(111) focussed, detuned 50
#date:      Tue Jun 20 14:27:31 2006
#npts:      418
#------------------------
#  energy     xmu       i0
  8879.0000      -1.3276930       117383.70
  8889.0000      -1.3312944       117185.70
  8899.0000      -1.3336289       117058.70
  8909.0000      -1.3305114       117276.70
  8919.0000      -1.3385381       117332.70
  8929.0000      -1.3403222       117332.70
  8939.0000      -1.3419374       117756.70
  8949.0000      -1.3353518       117199.70
  8959.0000      -1.3394162       117458.70
  8959.5000      -1.3390900       118720.70
  8960.0000      -1.3386739       119008.70
  8960.5000      -1.3382262       120520.70
```

Some sort of header perhaps containing metadata

Columns of numbers in ASCII text

Energy is usually in eV

μ may be stored of just raw values for I_0, I_t, and/or I_f

There are many variations for ASCII column files

Client applications are supposed to understand the columns

A Case for ASCII Data Files?

ASCII Column Files have some clear advantages for such small data sets:

- Readable by humans
- Editable by humans
- Will be readable (and editable) for long time
- Readable by all standard applications: (Excel, Calc, Origin, etc.).
- Readable by current XAFS analysis programs.
- Readable by any program environment
- History: Lots of existing data, some of them still useful.

When asked, most XAFS users and **beamline scientists prefer ASCII** data files

Why a standard data format?

Easier data sharing

between beamlines, analysis applications, …

Better data quality

(adopting common best practice approaches)

Sharing at what level?

Treatment of raw data

Extraction of $\mu(E)$ vs E starting from raw data

Treatment of preprocessed data

Analysis of $\mu(E)$ vs E

Exchange of reference data

Library of $\mu(E)$ vs E datasets

Sharing of small datasets

This can be achieved with tagged ASCII files

```
# IXASIF/1.0 MX/2.0
# Crystal: Si 111
# Beamline: APS 10ID
# Mirrors: single harmonic rejection mirror
# Start-time 2005-03-08 20:08:57
# Edge-energy: 7112.00
# Mu-transmission: ln($2/$3)
# Mu-reference: ln($3/$5)
# MX-Offsets: 11408.00 11328.00 13200.00 10774.00
# MX-Gains: 8.00 7.00 7.00 9.00
#---
# Fe K-edge, Lepidocrocite powder on kapton tape, RT
# 4 layers of tape
# exafs, 20 invang
#---
# energy     mcs3     mcs4     mcs6     mcs5
 6899.9609  48120 19430 2250 54540
 6900.1421  48390 19540 2260 54860
 6900.5449  48520 19610 2250 55110
 6900.9678  48930 19780 2280 55650
 6901.3806  48460 19590 2250 55110
```

Simple, but an API is still needed!

- It has to be robust
- Available at many languages
- Supported by analysis applications

Is there anything already available?

The CIF case

If we think about reference compounds, one cannot avoid taking a look at the Crystallographic Information File

Powder diffraction presents analogies with our problem:1D data, reference spectra, ...

The problem got solved back to 1991. In our case it could resemble to:

```
data_
_xafs_d_spacing     3.1356
_xafs_sample_temperature  77
_xafs_scanning_mode  step_scan
_xafs_acquisition_mode transmission

loop_
_xafs_energy
_xafs_absorption
energy_value0      mux_value0
energy_value1      mux_value1
energy_value2      mux_value2
```

data_ introduces fields of type “_name value”

loop_ introduces tabular data

You are not limited to one data_ and one loop_

You will get a more detailed talk later

The XML case

Extensible Markup Language

Created in ~1998, robust format widely supported

It can be used in databases, text editors, spreadsheets, …

```
<?xml version="1.0" encoding="WINDOWS-1252" standalone="yes"?>
<XAFSSpectrum>
  <ClassInstance Type="XAFSSpectrum" Name="Whatever">
                <d_spacing>3.1356 </d_spacing>
                <acquisition_mode>transmission</acquisition_mode>
                <scanning_mode>step_scan</scanning_mode>
                <sample_temperature>77</sample_temperature>
                <energy>value1, value2, value3, ...</energy>
                <absorption>value1, value2, value3, ...</absorption>
  </ClassInstance>
</XAFSSpectrum>
```

I will not argue if you tell me that XML is to ASCII what the artichoke is to the flowers …

Do we need a binary format?

If our aim is pretreated and reference data exchange , no.

We can make a long way just defining the minimal information needed to exchange a XAFS spectrum **and making our spectra available**

http://www.nature.com/authors/policies/availability.html

An inherent principle of publication is that others should be able to replicate and build upon the authors' published claims. Therefore, a condition of publication in a Nature journal is that authors are required to make materials, data and associated protocols promptly available to readers without undue qualifications in material transfer agreements. Any restrictions on the availability of materials or information must be disclosed to the editors at the time of submission. Any restrictions must also be disclosed in the submitted manuscript, including details of how readers can obtain materials and information.

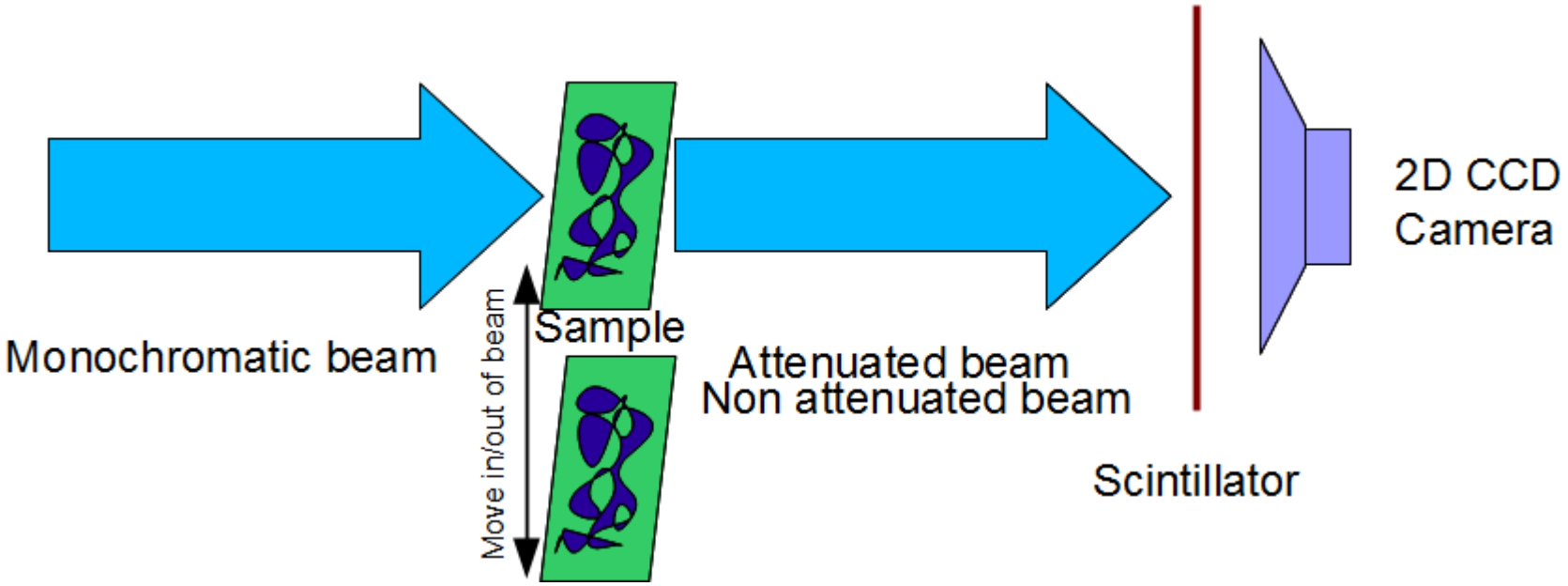
DISCLAIMER: I have just exposed a few metadata I considered useful in order to use a spectrum measured somewhere else. The list is not exhaustive and not agreed upon.

A case for a binary format

"Full Field" XANES Mapping

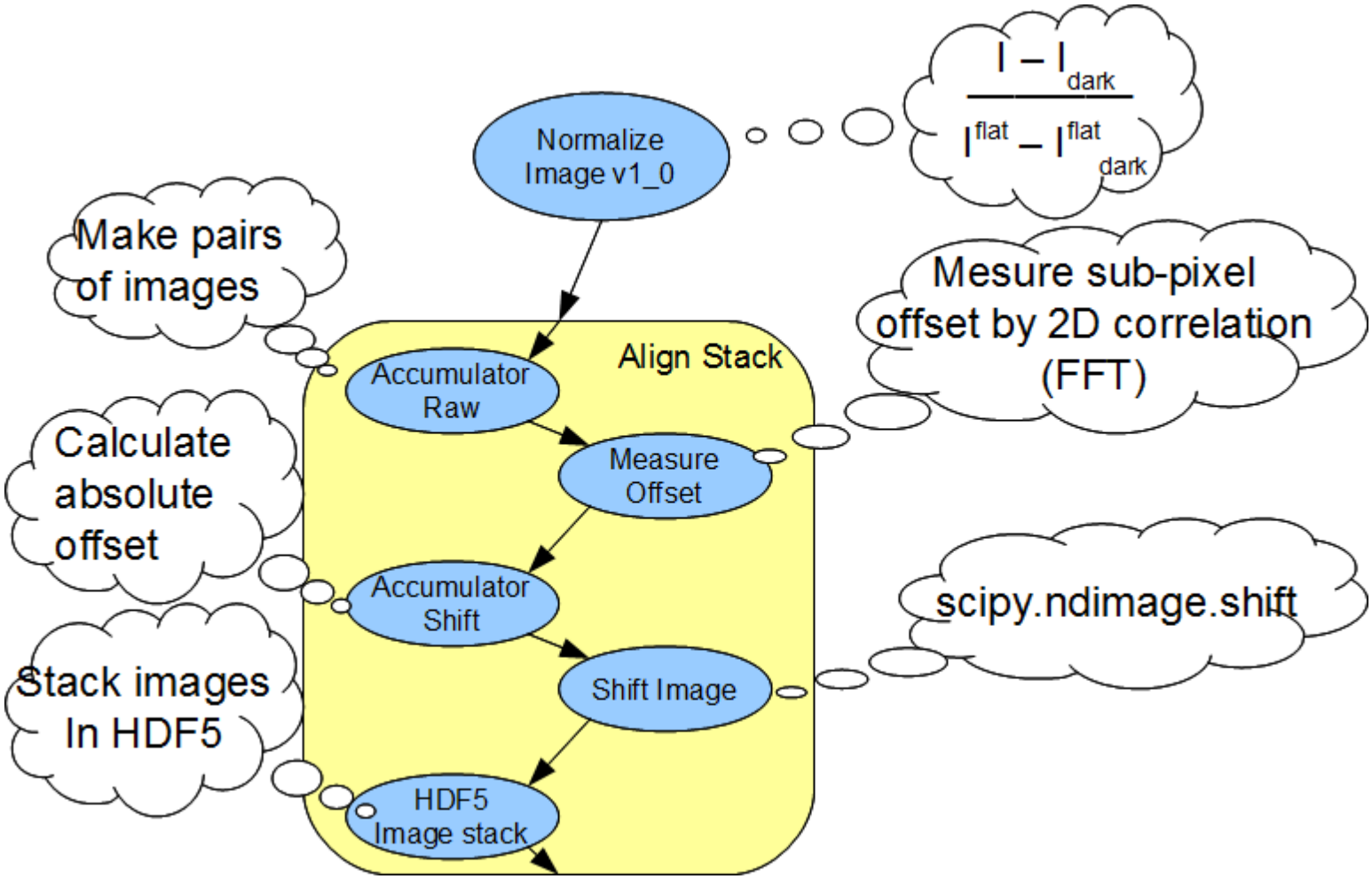
Scan in energy around an absorption edge

Spot size 1 mm
Resolution 500 nm

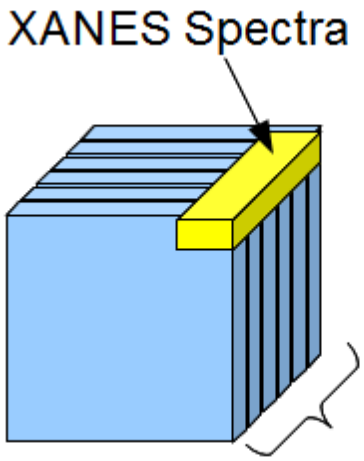


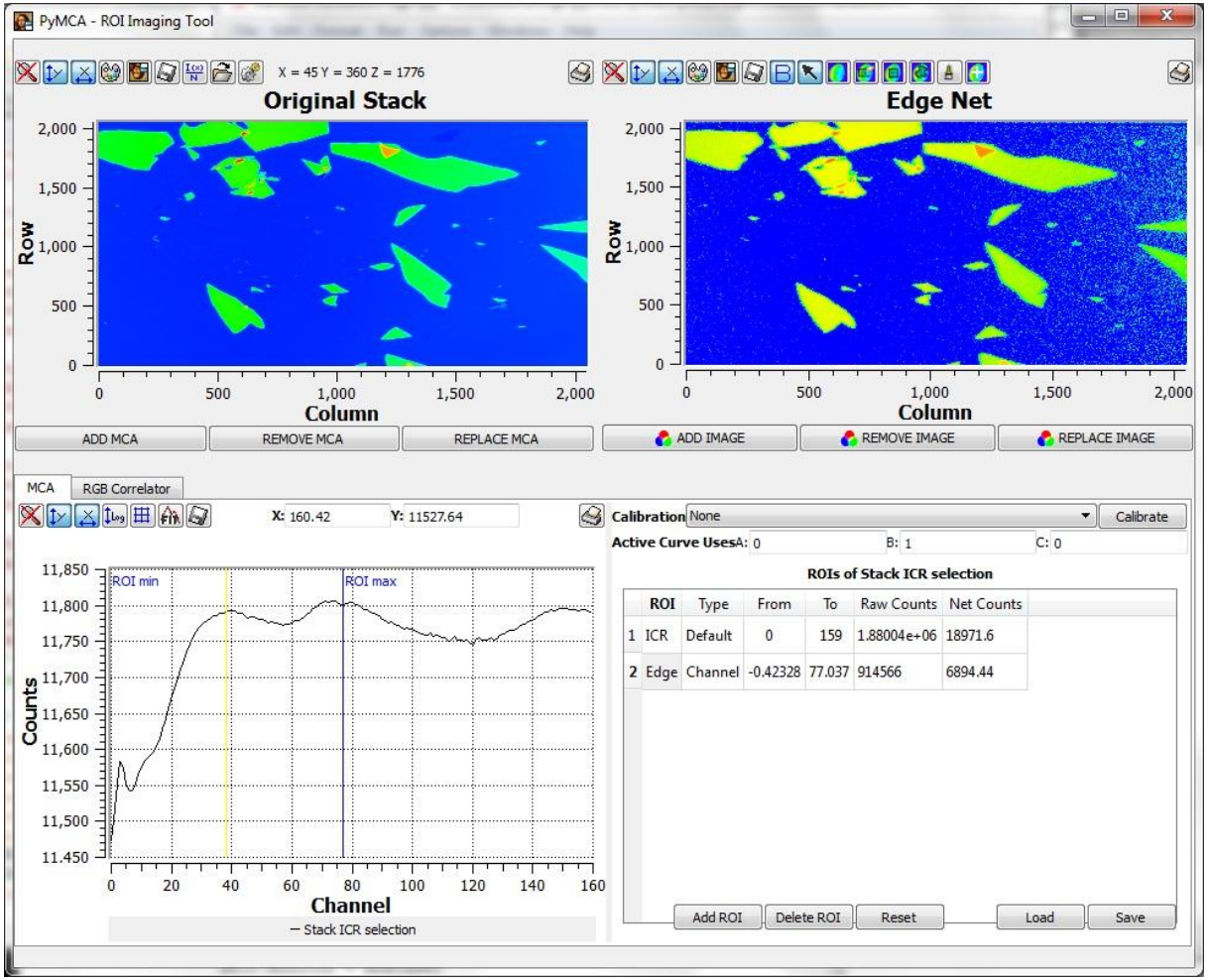
Align sample to correct submicron sample position changes

ID21 Alignment workflow



ID21 Full Field Tests





	ROI	Type	From	To	Raw Counts	Net Counts
1	ICR	Default	0	159	1.88004e+06	18971.6
2	Edge	Channel	-0.42328	77.037	914566	6894.44

A format for everything?

Efficient format to store different data types

Keep together counters, images, mca, …

Editable

Compression support

Widespread support

Efficient and easy access to the data for analysis

What is HDF5?

HDF stands for Hierarchical Data Format

- A file format for managing any kind of data
  - http://www.hdfgroup.org/HDF5/doc/H5.format.html
- Software system to manage data in the format

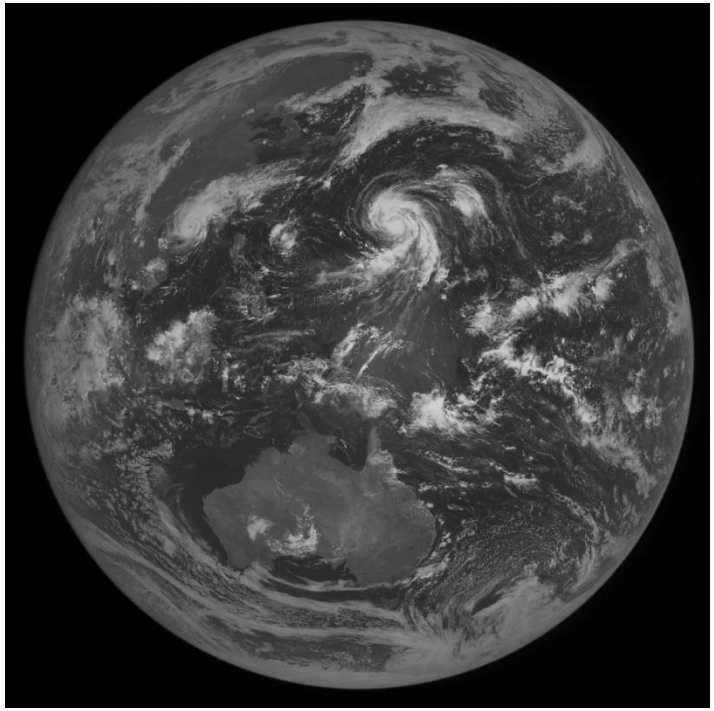
- Designed for high volume or complex data
- Designed for every size and type of system

Who uses HDF5?

- Applications that deal with big or complex data
- Over 200 different types of apps
- 2+million product users world-wide
- Academia, government agencies, industry
- ALBA, DESY, DIAMOND, ELETTRA, ESRF and SOLEIL are using it or committed to use it

Elements of perennity

- HDF format is the standard file format for storing data from NASA's Earth Observing System (EOS) mission.
- Petabytes of data stored in HDF and HDF5 to support the Global Climate Change Research Program.

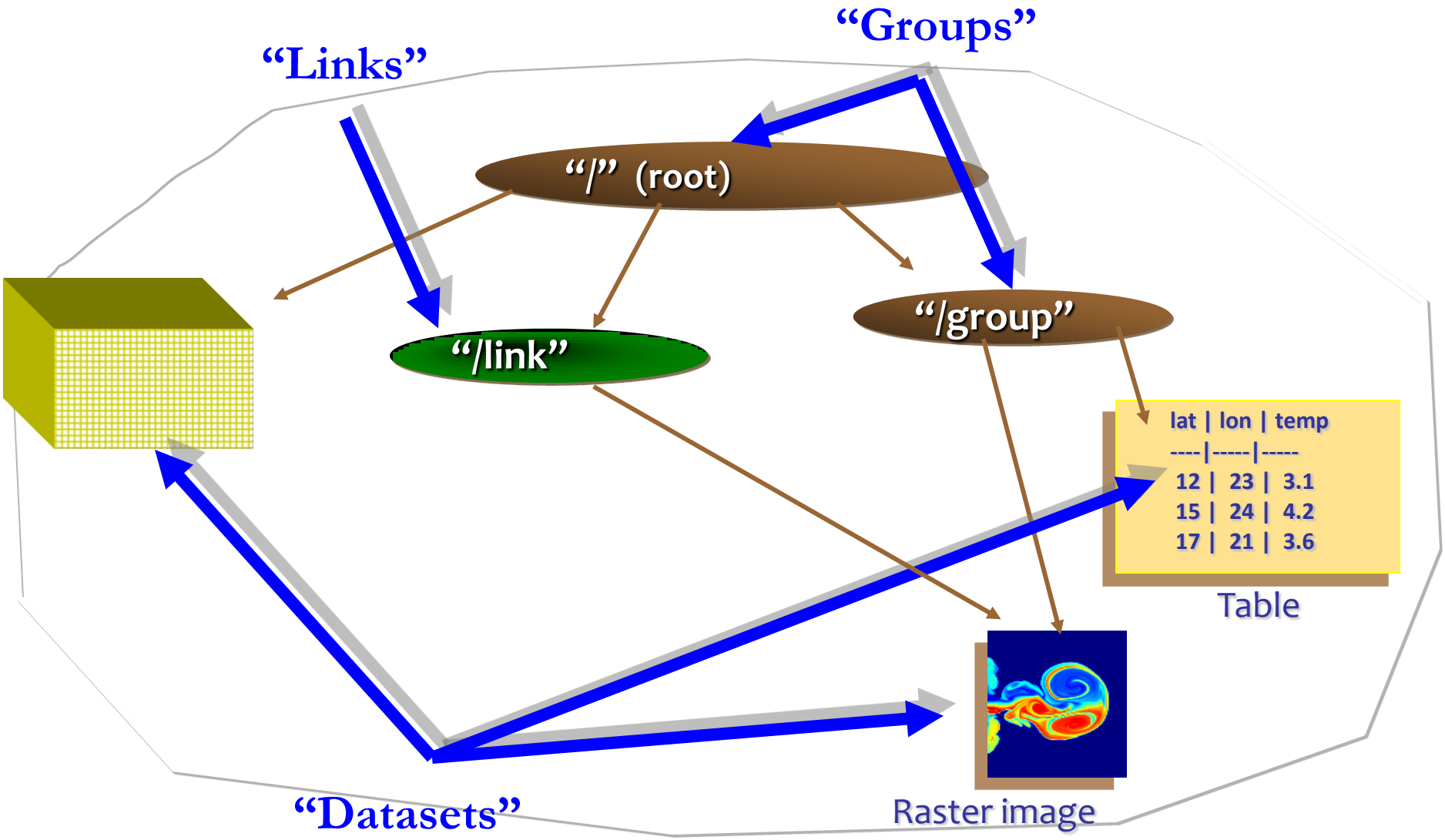


HDF5 model

- Groups – provide structure among objects
- Datasets – where the primary data goes
  - Rich set of datatype options
  - Flexible, efficient storage and I/O
- Attributes, for metadata annotations
- Links – point to other groups or datasets
  - Hard, soft and external flavors

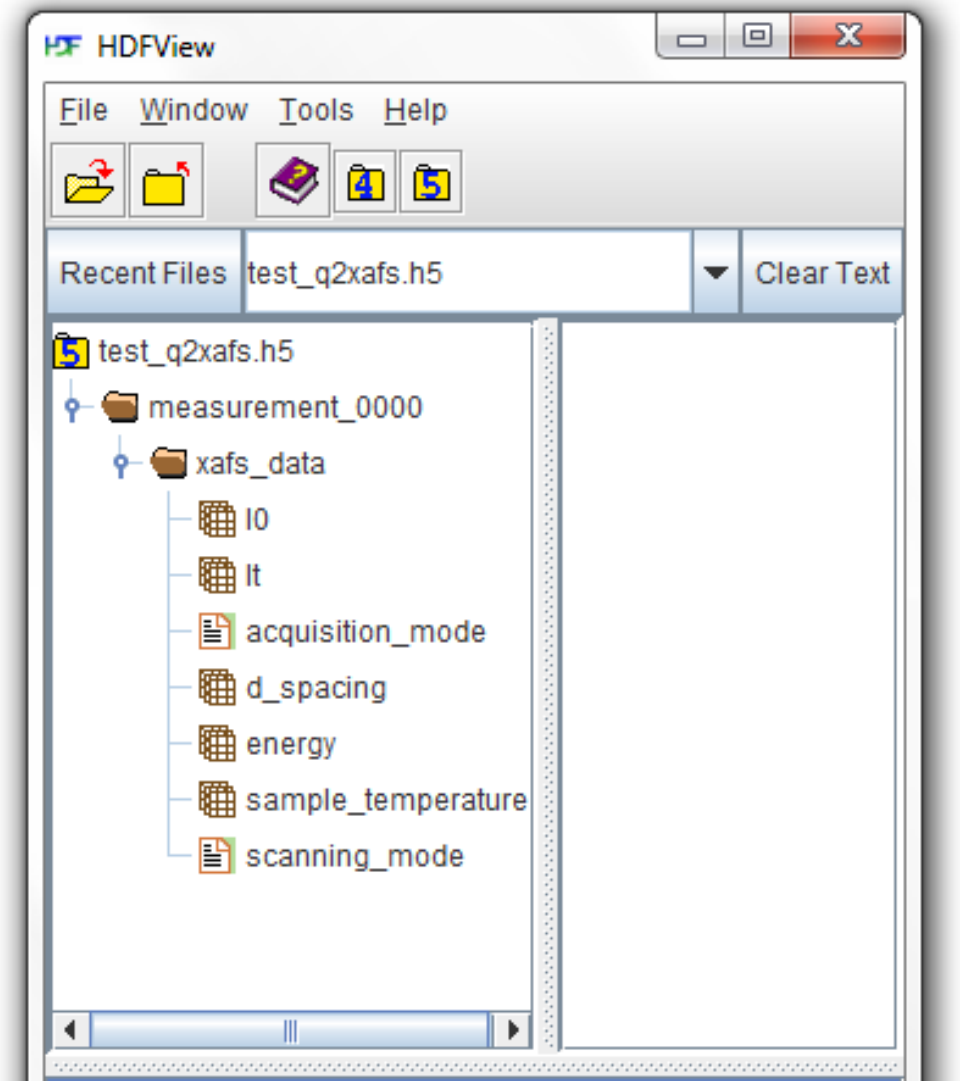
Everything else is built essentially from these parts

Structures to organize objects



HDF5 as a portable file system

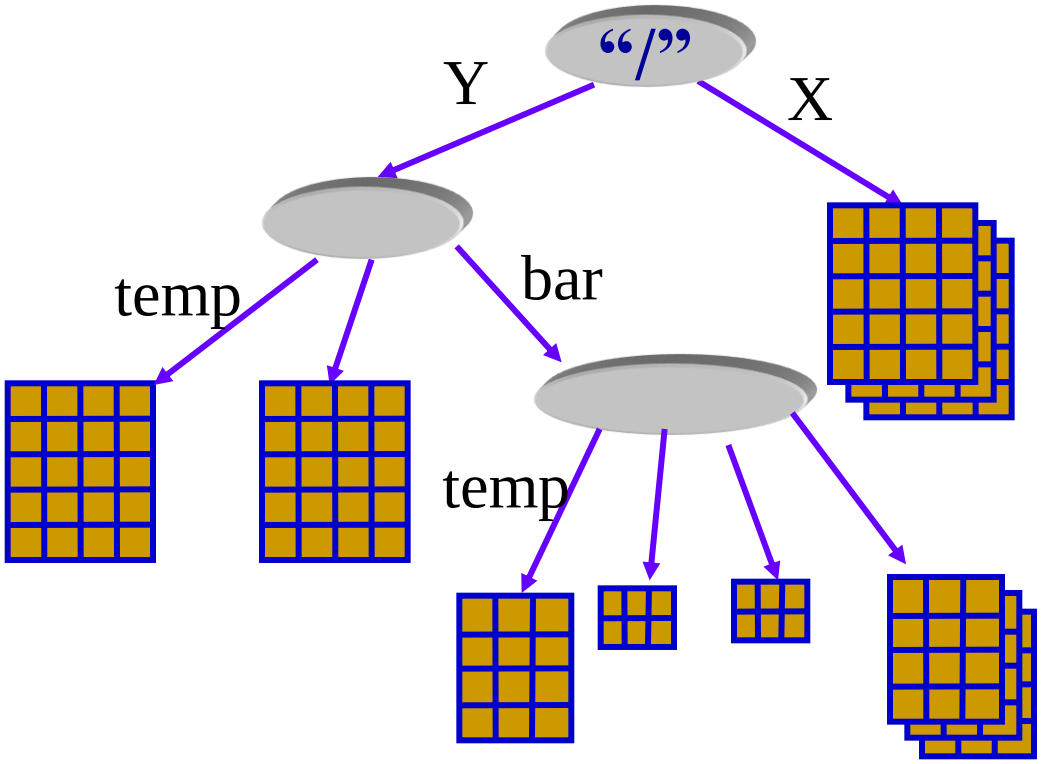
Think about an HDF5 file as a portable hard disk



- You can write whatever you want to it
- It supports links
- It supports compression
- It is widely supported

Path to HDF5 object in a file

/ (root)
/X
/Y
/Y/temp
/Y/bar/temp



HDF5 Dataset

Metadata

Dataspace

Rank: 3

Dimensions:
- Dim_1 = 4
- Dim_2 = 5
- Dim_3 = 7

Datatype

IEEE 32-bit float

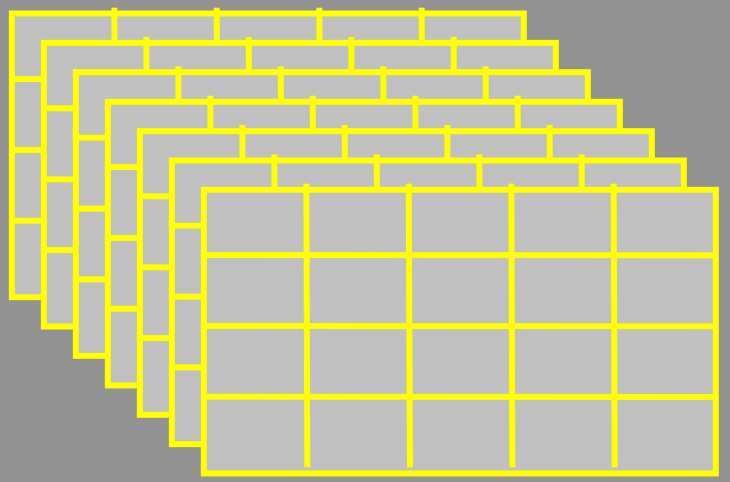
Storage info

- Chunked
- Compressed

Attributes

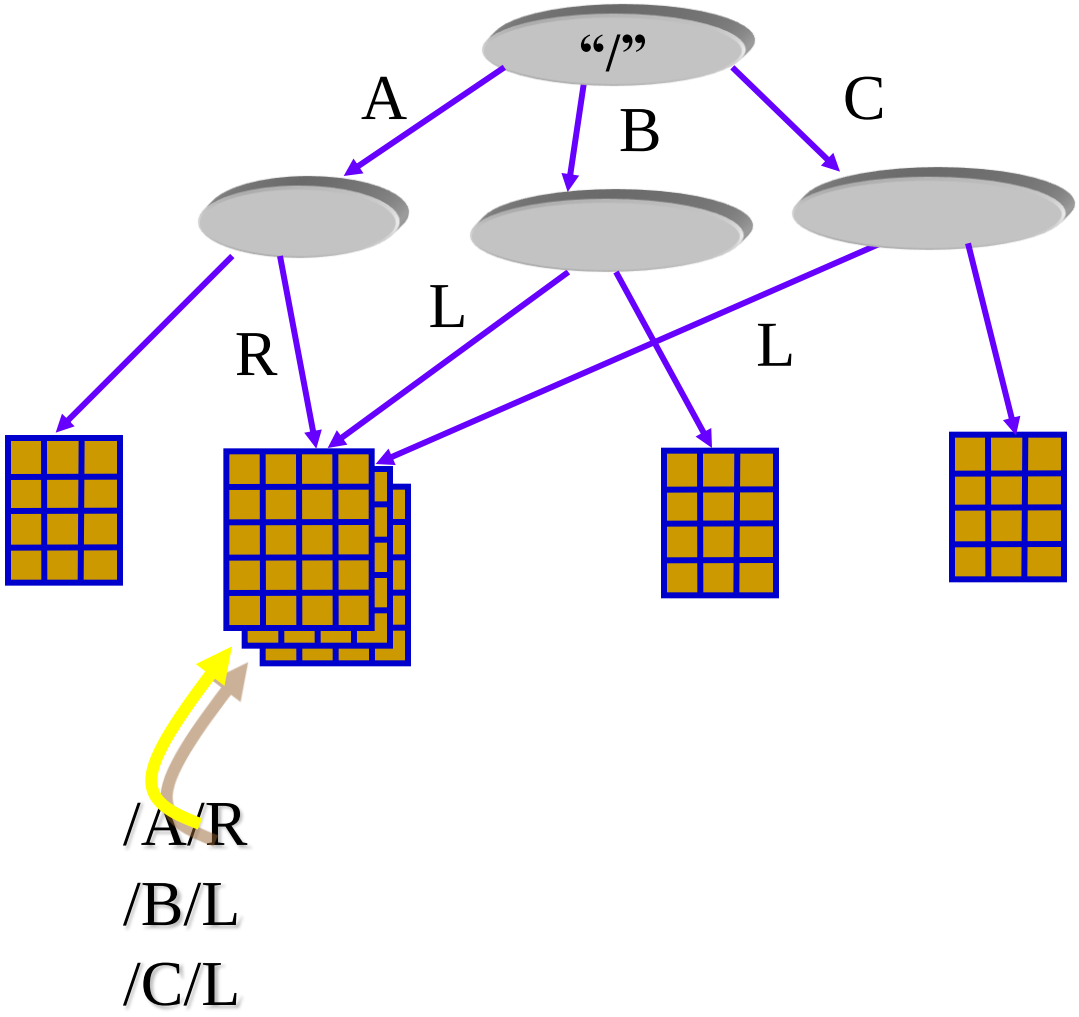
- Time = 32.4
- Pressure = 987
- Temp = 56

Data

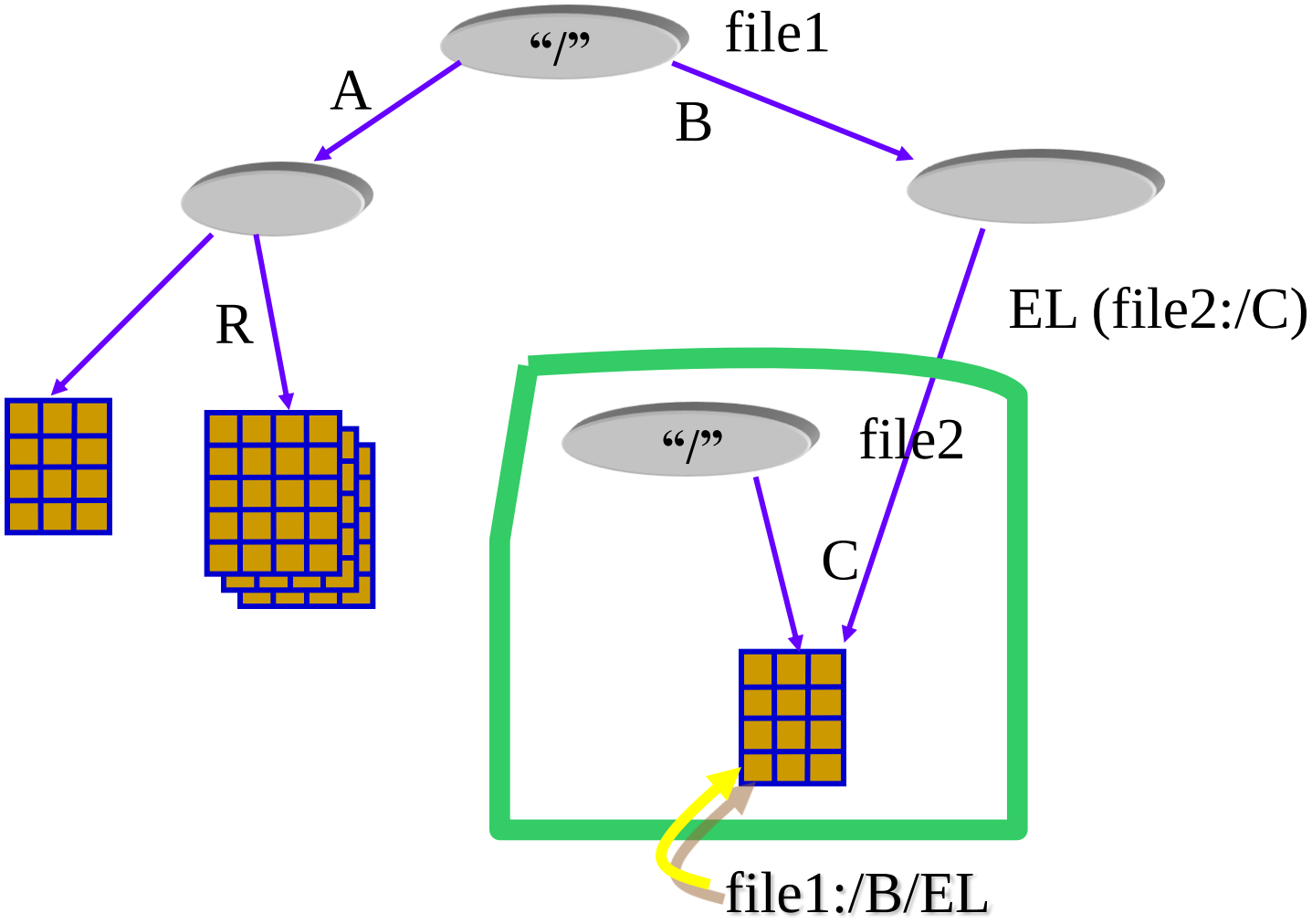


Attribute: data of the form “name = value”, attached to an object

HDF5 links



HDF External Links



More detailed information can be found at http://www.nexusformat.org
A convention (and an API) on top of HDF5 or (less used) XML

NXroot

Top level. One per file.

NXentry

One group per measurement

NXinstrument

Describe the instruments.

NXsample

Define the physical state of the sample during the scan

NXmonitor

Monitor data, i.e., counts, integrals, *etc*.

NXdata

The data to be plotted.
One NXdata group per plot

NXuser

Details of a user, i.e., name, affiliation, email address, *etc*

Defines a series of groups (“Directories”)

Groups tagged with an NXclass attribute

Many items discussed:

- grammar
- equipment
- units …

Really SAXS minded till now

No clear path for extension

Slow reactivity

API lags respect to HDF5

Opinion on Nexus

It is an **instrument minded approach**

It can be convenient for archival

It is **far from a data analysis minded approach**

Imagine having to browse three times three directory levels each time you want to retrieve a motor position, an intensity monitor and a measured spectrum

Nexus definitions can help solving this issue

A dictionary of links to datasets needed to perform a particular analysis

A data analysis approach would only require the definition

You can keep the freedom of HDF5 and decide up to what level you follow Nexus

NXentry simplifies handling several measurements in one file

NXdata is ideal for default measurement plots

Example of NeXus approach

```
NXroot
  entry_000 (@NXentry)
    title
    start_time
    end_time
    beamline_name (@NXinstrument)
      I0_detector(@NXdetector)
        data
      It_detector(@NXdetector)
        data
      Monochromator(@NXmonochromator)
        energy
        Si111(@NXCrystal)
          d_spacing
    sample(@NXsample)
      temperature
    monitor(@NXmonitor)
      data(link to I0_detector/data)
    xafs_data(@NXsubentry)
```

The NXsubentry would contain the actual definition as:

definition (string set to xafs for instance)
d_spacing (link to d_spacing)
sample_temperature (link to temperature)
I0 (link to I0_detector/data)
It (link to It_detector/data)
acquisition_mode (string)
scanning_mode (string)

Basically what we already had before…

Why not to force the existence of the definition while leaving the rest as optional?

ASCII versus Binary

ASCII

- Human readable
- Not well suited for large datasets
- Potential accuracy losses converting binary data to text
- Widely supported by XAFS analysis codes

Binary

- Machine readable
- Suited to large datasets
- No data conversion
- Not supported by many (any?) XAFS analysis codes

If we need an API, the physical format is not such a big issue

The real issues:

The definition of a clear policy for data exchange
The availability of analysis codes supporting the format

Conclusion

Different needs can require different solutions

If your target is pre-treated data exchange, ASCII based formats are certainly enough (IXAS cooking, CIF, XML, …) and combined with databases (next talk) can take us quite far

If you want a format for everything, HDF5 is certainly up to the task

NeXus adds very little value. An analysis minded version of it can be considered

Remember:

- The important decision is “what” is to be written, not “how”
- The developers of analysis applications have to be involved