Towards Data Format Standardization for X-ray Absorption Spectra

Matthew Newville[a], V. Armando Solé[b], James R. Hester[c], and Bruce Ravel[d]

[a]Center for Advanced Radiation Sources, University of Chicago, USA
[b]European Synchrotron Radiation Facility, Grenoble, France
[c]Bragg Institute, Australian Nuclear Science and Technology Organization, Sydney, Australia
[d]National Institute of Standards and Technology, Gaithersburg, MD USA

Abstract

To better facilitate exchanging and archiving of X-ray absorption data, we propose a data format for $\mu(E)$ or $\chi(k)$ using a simple, plain-text file. The XAS Data Interchange (XDI) format uses space-delimited columns for numerical absorption data, similar to the data formats currently used at many XAS beamlines. XDI offers a clearly defined structure for metadata in the file header. A programming library for reading XDI files and several example XDI datasets are publicly available. Because XDI accommodates only a single spectrum, we also discuss formats that are capable of combining multiple XAFS spectra and other data into multispectral libraries using either the hierarchical data format (HDF5) or relational databases based either on the CIF format or a SQL-based database engine.

X-ray Data Interchange Format: Exchanging XAFS Data

The exchange of XAFS data is a common need in the XAFS community. Retrieving archived data on *standards* or *model compounds*, and the ability to compare data taken at different facilities and beamlines are both continuing needs for both XANES and EXAFS analysis.

We need to share XAFS spectra between facilities for decades to come.

There is no commonly accepted data format for XAFS data, though most data collection and processing software use some form of a Plainttext (ASCII) file. Such files are human-readable and used in many third party applications.

XDI Design Principles

I. $\mu(E)$ and $\chi(k)$ are the spectral "unit of currency"
II. The XDI format should use Plaintext ASCII for maximum portability.
III. The XDI format should be fully, and publicly documented.
IV. The XDI format must support both array data and meta-data.
V. The XDI format should be extensible.
VI. Application/programming interfaces (APIs) should be available for many computer languages.

XDI: Current Status

We currently have, ready for testing:

12 example XDI files 12 different spectra, all valid XDI Files.
C library and Shared Object Library (DLL) for reading XDI files: Source code, Windows, Mac OSX, Linux binaries.
Python Library Built using C library.
Fortran, Perl Library, Documentation Underway. **Volunteers sought!**

Example XDI File

```
#XDI/1.0 ← Version Info
#Beamline.name: APS 10ID
#Detector.I0: N2 15cm
#Detector.I1: N2 15cm
#Sample.name: gamma-FeO(OH)
#Mono.name: Si 111
#Mono.d_spacing: 3.13553
#Element.symbol: Fe
#Element.edge: K
#Column.1: energy eV ← Column Labels, Units
#Column.2: mutrans
#Column.3: i0
#/// ← start multi-line comment
#Fe K-edge, Lepidocrocite powder
#on 4 layers of tape
#------- ← End Header, Begin Arrays Table
#
  6899.9609 -1.3070486 149013.70
  6900.1421 -1.3006104 144864.70
  6900.5449 -1.3033816 132978.70
  ...
```

XDI File description

Some metadata are **required** in all files (case insensitive):

Element.symbol Atomic symbol of absorbing element
Element.edge K, L3, M4, etc
Mono.d_spacing Required if mono angle is given and mono energy is not. Strongly encouraged for all data,

Supported column labels.

- **energy** Mono Energy, in eV or keV.
- **angle** Mono Angle, in degrees.
- **i0** Monitor, I_0.
- **time** Integration time.
- **itrans** Transmitted intensity, I_1.
- **ifluor** Fluorescence intensity, I_f.
- **irefer** Reference signal, I_2.
- **mutrans** $\mu(E) = -\ln(I_1/I_0)$.
- **mufluor** $\mu_f(E) = I_f/I_0$
- **murefer** $\mu_r(E) = -\ln(I_2/I_1)$.
- **normtrans** Normalized $\mu(E)$
- **normfluor** Normalized $\mu_f(E)$
- **normrefer** Normalized $\mu_r(E)$
- **k** Wavenumber
- **chi** $\chi(k)$

Other metadata describe additional information about the spectra, with entries arranged as **Family.Keyword: Value**. Families include:

- **Column** labels for array columns
- **Element** absorbing element (edge)
- **Mono** monochromator (d-spacing)
- **Facility** facility (name, x-ray source)
- **Beamline** beamline (name, optics)
- **Scan** scan (step size, date)
- **Detector** detector (types, gases)
- **Sample** sample (name, prep)

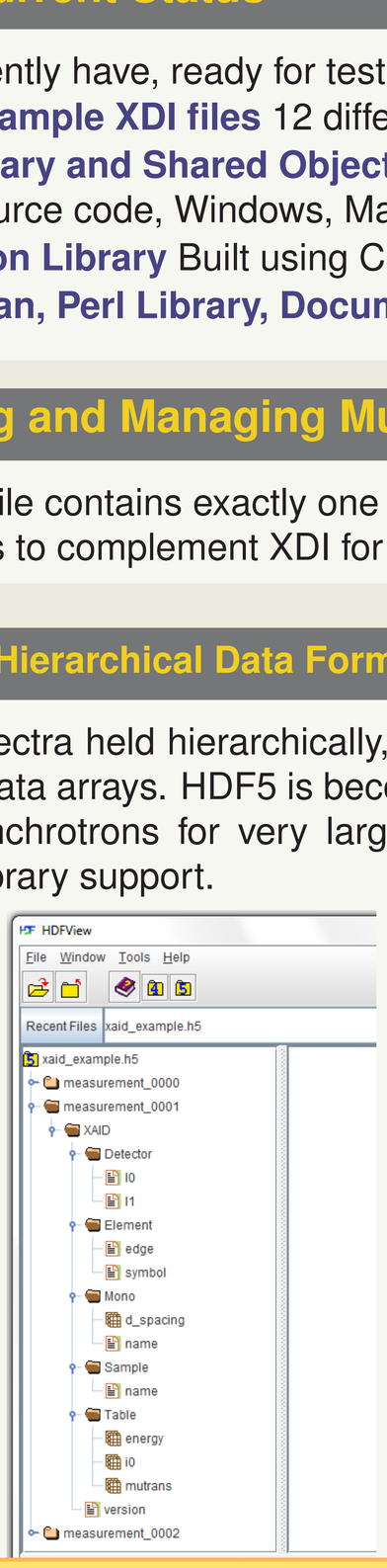
This list is not exhaustive. XDI is extensible, and new metadata fields can be added without affecting other applications.

Storing and Managing Multiple XDI Spectra: Data Libraries

An XDI file contains exactly one $\mu(E)$ or $\chi(k)$ XAFS spectrum. To support libraries of spectra or multi-spectral data sets, we propose three possible solutions to complement XDI for multiple spectra. These suggestions should not be viewed as exclusive.

HDF5 – Hierarchical Data Format

Multiple Spectra held hierarchically, with highly optimized data arrays. HDF5 is becoming standard at synchrotrons for very large datasets. Excellent library support.



Crystallographic Information Framework

CIF can encapsulate one or more data sets in a text file. Within each block, data and meta-data appear as key-value pairs or in a series of tables. This gives a text representation of a relational database.

```
data_v2o5_nanotube
_xafs_absorber.atom V
_xafs_absorber.edge K
_xafs_source.identification 'KEK-PF BL20B'
_xafs_source.location 'Tsukuba, Japan'
loop_
_xafs_detectors.label xafs_detectors.position
_xafs_detectors.type
monitor monitor ionisation
io-detector detector ionisation foil foil
ionisation

loop_
_xafs_ionisation_detector.label
_xafs_ionisation_detector.gas_pressure
_xafs_ionisation_detector.length
_xafs_ionisation_detector.amplifier_type
_xafs_ionisation_detector.amplifier_gain
monitor 1 10 'Keithley' 10
io-detector 1 20 'Keithley' 10
foil 1 5 'Keithley' 11

loop_
_xafs_reduced.energy _xafs_reduced.absorbance
5248.52108 0.813707373
5258.29435 0.798733337
5268.26606 0.781069442
...
```

SQLite – relational database in a file

A true relational database in a portable file. Supports SQL and many languages. An industry standard for data storage. Like ASCII, not high performance for numerical arrays.