XAS Data Interchange Format Draft Specification, version 1.0

XDI Working Group

- Matthew NEWVILLE (University of Chicago Center for Advanced Radiation Sources, APS)
- Bruce RAVEL (NIST) bravel AT bnl DOT gov
- V. Armando SOLÉ (ESRF)
- Gerd WELLENREUTHER (DESY)
- mailing list - http://millenia.cars.aps.anl.gov/mailman/listinfo/xasformat
- GitHub organization - https://github.com/XraySpectroscopy

Contents

1 Introduction

This document describes the XAS Data Interchange Format (XDI), version 1.0, a simple file format for a single X-ray Absorption Spectroscopy (XAS) measurement.

This document is an effort of an *ad hoc* working group reporting to the International X-ray Absorption Society (IXAS) and the XAFS Commission of International Union of Crystallography (IUCr-XC). The charge of this working group is to propose standards for the storage and dissemination of XAS and related data.

1.1 Purpose

We are define this format to accomplish the following goals:

- Establish a common language for transferring data between XAS beamlines, XAS experimenters, data analysis packages, web applications, and anything else that needs to process XAS data.
- Increase the relevance and longevity of experimental data by reducing the amount of *data archeology* future interpretations of that data will require. (The Farrel Lytle “database” is a particularly trenchant example of data archeology.)
- Enhance the user experience by promoting interoperability among data acquisition systems, data analysis packages, and other applications.
- Provide a mechanism for extracting and preserving a single XAS-like data set from a related experiment (for example, a DAFS or inelastic scattering measurement) or from a complex data structure (for example, a database or a hierarchical data file used to store a multi-spectral data set).
- Provide a representation of an XAS spectrum suitable for deposition with a journal.

In short, we are trying to share data across continents, decades, and analysis toolkits.

This format is intended to encode a single XAS spectrum in a data file with metadata. It is not intended to encode relationships between many XAS measurements or between an XAS measurement and other parts of a multi-spectral experiment.

In order to fulfill these goals, *XDI* files provide a flexible, consistent representation of information common to all XAS experiments. This format is simpler than a format based on XML, HDF, or a database; it yields self-documenting files; and it is easy for both humans and computers to read. Its structure is inspired by that of Internet electronic mail (See RFC822: Standard for ARPA Internet Text Messages), a plain-text data format which has proven to be robust, extensible, and enduring. It can be read as is by many existing programs for XAS and other data analysis and by many scientifc plotting programs.

Due to these advantages, and because of our intention to develop free software tools and libraries that support XDI, we hope that this file format described in this specification will see wide adoption in the XAS community.

1.2 Scope

We do not intend this specification to dictate the file formats used by data acquisition systems during XAS experiments, although this may be a suitable format for that purpose. Any attempt to do so would be unreasonable due to the number of different data acquisition systems currently deployed at synchrotrons around the world, the variety of experiments performed at these installations, and the continuing development of new experimental techniques.

This specification addresses the representation of a single scan of XAS data after an experiment has been completed.

A beamline which adopts this specification shall either use this format as its native file format or shall provide their users with tools that convert between their native file formats and *XDI*. In short, when that beamline sends a user home with XAS data that is ready to be analyzed, that XAS data will be stored in this format. We intend to encourage this practice by developing tools for reading, editing, writing, and validating *XDI* files. Beamlines may choose to modify their data acquisition systems to write data using this format in situations where that would be appropriate. We plan to assist in this effort by developing libraries for popular programming languages which can read, manipulate, and write *XDI* files.

With their experimental data stored in *XDI* files, users may choose data analysis packages which are capable of reading this format. It is our hope that, as this specification gains wider adoption, users will ultimately be freed from the responsibility of understanding file formats. With this aim in mind, we shall assist software developers in supporting *XDI* files.

2 Content of the XAS Data Interchange File

XDI files contain two sections, a header with information about one scan of an XAS experiment followed by the data collected during that scan. The header section consists of versioning information, a series of fields that contain information about the scan, an area for users to store comments about the experiment, and a sequence of labels for the columns of data. The data section contains these columns, with each row corresponding to one point of the scan.

The header has been designed to contain arbitrary metadata describing the contents of the file. This metadata is organized in a way that is easily readable by both humans and computers. These fields, described below, contain information about XAS experiments which is useful for both users and applications. Some headers are defined, see Sec. 4.1.

3 Definition of the XAS Data Interchange Format

This section of the *XDI* specification formally describes the structure of *XDI* files.

3.1 Requirements

The key words “**must**”, “**must not**”, “**required**”, “**should**”, “**should not**”, “**recommended**”, “**may**”, and “**optional**” in this document are to be interpreted as described in RFC 2119. See Key words for use in RFCs to Indicate Requirement Levels.

An *XDI* implementation is *not compliant* if it fails to satisfy one or more of the **must** or **required** level requirements presented in this specification.

3.2 Notational Conventions

Several *XDI tokens* are used throughout the definition of the *XDI* file.

- The white-space token is a space (ASCII 32) or a tab (ASCII 9)
- The comment token is a hash (`#`, ASCII 35)
- The end-of-line token can be carriage return (ASCII 13, `CR`, Mac-style), newline (ASCII 10, `LF`, Unix-style), or a sequence of one carriage return + one newline (Windows-style)
- The namespace-separator token is a dot (`.`, ASCII 46)
- The metadata-end token is a colon (`:`, ASCII 58)
- The field-end token consists of three or more foreward slash characters (`/`, ASCII 47)
- The header-end token consists of three or more dash characters (`-`, ASCII 45)

3.3 Text Encoding

The header and data sections of an *XDI* file are comprised of structured US-ASCII (see ASCII table text. Header field values that are “free-form” or “text” **may** contain UTF-8 encoded Unicode text, although Unicode support in applications that use *XDI* files **should not** be assumed, particularly those written in languages with weak or non-existent Unicode suport (e.g. Fortran). Unicode support in applications that use *XDI* files is **optional**, but **recommended**. The US-ASCII coded character set is defined formally by ANSI X3.4-186 (see section 4.1.2 of Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types). The Universal Character Set (Unicode) is defined by ISO/IEC 10646. The UTF-8 translation format is defined by IETF RFC 3629.

3.4 Structure of the Header Section

The header section of an *XDI* file appears at the beginning of the file and is comprised of structured text.

Header line rules:

- Every line of the header **must** begin with a comment token and must end with an end-of-line token
- Header lines may be of any length, but users of *XDI* **should** remember that XAS software may be implemented in a programming language without dynamic memory allocation (e.g. Fortran) and so should restrict lines to 2048 characters.

Header lines are subdivided into four subsections — versioning information, header fields, user comments, and column labels — with two separators, one of which is always **required**. These subsections **must** occur in the following sequence:

1. The **required** first line of the file is the version line, described in Sec. 3.4.1.
2. This is followed by header lines, which can be defined headers or extension headers. These two header types are explained in Sec. 3.4.2. Some headers are **required**, as explained in Sec. 4.4.
3. The header lines are separated from the user comments by the field-end line. If the comment section is present, this separator line **must** also be present. If the comment section is absent, the header lines **may** terminate with the end-of-header line. The field-end line is defined at the end of this section.
4. The **optional** comment section is for user-supplied, free-format text. Each line begins with a comment token and ends with an end-of-line.

5. The comment section ends with the **required** header-end line. The header-end line is defined at the end of this section.
6. The last line before the data is a line of **optional** column labels which identify the columns of data. If present, there **must** be as many labels as there are columns. The label line begins with a comment character and ends with an end-of-line. See Sec. 3.4.4.

The field-end and header-end separator lines serve specific, syntactic purposes in the *XDI* grammar. For the human reader, the line of dashes is a visual cue denoting the end of the headers and beginning of the data. The field-end line serves to separate and distinguish field lines from freely-formatted user comments, which may resemble a header fields or other grammatical constructs. Similarly, the header-end line serves to distinguish column labels from user comments, which are otherwise grammatically identical elements of the data file.

Definitions of separator lines

- **Field-end line:** comment token + field-end token + end-of-line token

  ```
  # ////////////
  ```

- **Header-end line:** comment token + header-end token + end-of-line token

  ```
  # ------------
  ```

3.4.1 Version Information

The first line of the *XDI* header contains the *XDI* version to which the file conforms. *XDI* represents versions of the file format with a `<version>.<subversion>.<release>` numbering scheme. The `<subversion>` number is incremented when changes are made to the format that do not affect compatibility with previous versions, as when new defined header fields are defined. (A parser compliant with an earlier minor version would treat the newly defined header as an extension field. Propagated to an output file as an extension field, this field would then be interpreted correctly by a more recent parser.) The `<version>` number is incremented when major changes are made to the format, as when the definition of the contents of a defined header field is altered. The `<release>` is incremented when the library or its documentation is altered without altering the specification in any way. Use of the `<release>` number in *XDI* files is **optional**.

A series of **optional** entries denoting further versioning information, separated by white space, **may** follow the *XDI* version. There **may** be any number of extra versioning strings. These version entries allow programs to annotate the file as it proceeds through the collection and analysis process. Such annotation is **optional** although version information **should** be included in this sequence by software that create *XDI* files containing extension fields (see Sec. 4.3). When an application adds versioning information to this line, it **should** be appended to the end of the line. The order of the optional version entries is undefined but **should** be preserved by application reading the file in order to accurately represent the time sequence in which applications have manipulated the file.

Note that the *XDI* version, subversion, and release numbers **must** be treated as integers that **may** contain more than a single digit. `XDI/1.12` is a higher (more recent) version than `XDI/1.2`.

This specification does not impose a restriction on how applications identify and version themselves. However, a single application **must** identify and version itself using a single text sequence without white space. Some acceptable examples follow. The first example shows an application which uses the same format as the *XDI* version rule, which is the **recommended** format for application versioning. The second shows names of the data acquisition and data processing programs, each is specified by name but without the **recommended** version numbers.

```
    # XDI/1.0 Datacollectatron/7.75

    # XDI/1.0 XDAC Athena
```

The name of the the additional applications **must** be used for any extension headers associated with that application (see Sec. 4.3).

The following is an example of a data acquisition program with an odd name and which uses non-standard versioning.

```
    # XDI/1.0 XAS!Collect-3000
```

There are two problems with this. The versioning information will not be recognized as such. Also the requirement that the application name be used as the family name of any extension headers added by the program will result in a non-compliant family name. The exclamation point is not an allowed character for family names.

3.4.2 Header Fields

Immediately following the version line is the header fields subsection. These fields are arranged in a manner similar to the header of an Internet electronic mail message, although *XDI* fields **must not** span multiple lines. Each field consists of a case-insensitive name, a separating colon, and an associated value. The structure of the name is presented in Sec. 4. When multiple occurrences of the same field are present the value of the last occurrence **must** be used as the value for the field.

Except in the case of a defined header whose value has a defined structure, values are assumed to be free-form text, as explained in Sec. 3.3. The defined fields are explained in Sec. 4.1.

When a user comments section is present, the header fields subsection must end with a field-end line. When a comments section is absent, the header fields subsection **must** end with a field-end line. See Sec. 3.4 for the definitions of the separator lines.

3.4.3 User Comments

Following the dividing line at the end of the header fields subsection is the area of the header that contains user comments. This area is reserved for comments supplied by the experimenter and **must not** be used by software as a place to store other information. Refer to Sec. 4.3 for information about using extension fields for this purpose.

This section **may** contain zero lines of commentary or empty lines containing no text other than the **required** comment token. An empty line **must** be treated as a zero-length comment line. This section **must** end with a header-end separator line.

When extracting the comment subsection from an *XDI* file, software **may** remove no more than one leading space and any trailing white space from each comment line but **must not** further alter the line's contents, all interior white space **must** be preserved.

Applications **must** preserve all user comments, including empty lines and interior white space, when exporting the *XDI* data as an *XDI* file.

3.4.4 Column Labels

The final line of the *XDI* header contains the labels for each column of data in the data section of the file, separated by white space. There **must** be one label present for each column of data present in the data section.

The number of column labels **must** equal the number of columns of data in the data section.

Note that each column label **must** be a word, white space **must** separate the labels, and labels **must not** contain white space. For specific column labels which, in natural language, would consist of two or more words, the use of CamelCase, underscores, or some other way of substituting for white space is **required**.

The column labels in the column label line **must** match the values of the headers in the `Column` namespace. See Sec. 4.2.

Several common array labels are defined in the Dictionary of Metadata and **must** be used when those arrays are present in a file.

3.5 Data Section

The data section of the file contains white-space-delimited columns of integers or floating-point numbers. Lines in the data section **must not** begin with comment tokens. Lines in the data section **may** begin with white space. Leading white space on a line in the data section **must** be ignored.

The first (left-most) column of data **must** contain the abscissa (energy or angle) array.

Blank lines in this section **must** be discarded. The number of columns **must** be the same for all lines that contain data. All columns, including columns containing a measurement of time, **must** be represented as inegers or as floating point numbers.

Locale is **not** respected when interpreting floating point numbers. The decimal mark **must** be a dot (`.`, ASCII 46). The decimal mark **must not** be a comma (`,`, ASCII 44).

It is **recommended** that measurements of time be represented as a numerical offset relative to the value of the `Scan.start_time` header.

4 *XDI* Fields

When present, header fields **must** comply with the associated parsing rules. All fields which fail to do so **must** be ignored by an application.

XDI fields use a simple namespace concept as their structure. The name of the field **must** be of two words. The first word in the name **must** start with a letter and **must not** start with a number, underscore, or dash. The second word **must** consist of letters, numbers, underscore, or dash. Letters are ASCII 65 through 90 (`A-Z`) and ASCII 97-122 (`a-z`). Numbers are ASCII 48-57 (`0-9`). Underscore (`_`) is ASCII 95 and dash (`-`) is ASCII 45.

The two words in the name **must** be separated by the dot character (`.`, ASCII 46). The name **must** end with a colon (`:`, ASCII 58), which is the character which delimits the field name from its value. The colon **may** be followed by white space, then **must** be followed by the value of the field. A missing value **must** be interpreted as an empty string.

Here are some examples which demonstrate both the format of the *XDI* field and the *namespace* concept:

```
# Beamline.name: APS 20BM
# Beamline.source: bend magnet
# Column.1: energy eV
# Column.3: i0
```

The namespaces are used to group related fields. In the example above, two namespaces are shown. The `Beamline` namespace conveys characteristics of the beamline at which the data were measured, while the `Column` namespace explains how to interpret the columns in the data section.

There are two kinds of namespaces. Defined namespaces (see Sec. 4.1) are defined in the Dictionary of Metadata. Extension namespaces (see Sec. 4.3) may be added by application developers to insert new metadata into the data file.

Header fields are case insentitive. As an example, the following lines **must** be interpreted identically:

```
# Beamline.name: APS 20BM
# beamline.name: APS 20BM
# BEAMLINE.NAME: APS 20BM
# bEAmlINe.naME: APS 20BM
```

Capitalization (like the first of these examples) of the namespace is **recommended**.

4.1 Defined namespaces

See the Dictionary of Metadata for the current list of defined namespaces and defined metadata.

Three defined fields are **required** in a valid *XDI* file:

1. `Element.symbol`: The symbol of the absorber element
2. `Element.edge`: The measured absorption edge
3. `Mono.d_spacing`: The d-spacing of the monochromator crystal. This is only **required** when the energy axis is conveyed as monochromator angle or encoder step count. When the energy axis is conveyed in energy units or pixel count, providing the d-spacing is strongly **recommended** to enable correction of the energy axis for a miscalibration due to inaccuracies in the translation from angular position of the monochromator to energy.

All other fields are **optional**, although some are **recommended** and constitute good practice, as explained in the Dictionary of Metadata.

A header in a defined namespace **should not** appear more than once in a file. When multiple occurrences of the same field are present, the value of the last occurrence **must** be used as the value for the field.

4.2 The Column namespace

The Column namespace is the mechanism by which *XDI* files provide directions about how to extract useful information from the columns in the data section of the file.

1. All fields in this namespace **must** be of the form `Column.N`, where `N` represents an integer. The integer is used to identify a particular column in the data file. These integers begin at 1 and count from the left-most column in the data section. The value of a Column field is used to indicate the contents of that column.

2. There are several defined column labels. These are words that **must** be used to describe a column when that column is present in the data file and identified among the header fields. The list of defined column labels is given in the Dictionary of Metadata.
3. The abscissa of the data **must** be in the first (left-most) column and **must** be identified by the `Column.1` header.
4. Data **may** be stored using any reasonable units for the abscissa, but that choice of units must be identified in the value of the `Column.1` header. Allowed abscissa choices include energy (in units of eV or keV), pixel (appropriate for dispersive detection of XAS), or angle (in units of degrees, radians, or motor steps). eV units are **recommended**. If units of motor steps are chosen, then adequate information **must** be provided via headers in the `Mono` namespace to translate the abscissa into energy units.
5. The header identifying the abscissa **must** provide two values: the column label for the abscissa and the corresponding units. Here is an example:

   ```
   # Column.1: energy eV
   ```

   All other headers in the Column namespace **must** provide one value – the column label – and **should** provide units, if appropriate.

A list of column labels and their meanings along with unit definitions for the abscissa are defined in the Dictionary of Metadata. Any such array included in an *XDI* file must use those label definitions. Along with column labels defining the abscissa and various detectors, labels for representing EXAFS data in various stages of data processing ($\mu(E)$, normalized $\mu(E)$, $\chi(k)$, the Fourier transform of $\chi(k)$, or the Fourier filter of $\chi(k)$) are provided.

4.3 Extension headers

Extension fields are fields present in the header of an *XDI* file that are not defined in the *XDI* specification. Such fields **must** be structured by the same syntax as a defined field. The values of extension fields **must** be interpreted as free-form text. Any field not defined from a defined namespace (see Sec. 4.1) **must** be considered an extension field.

Data acquisition systems and data analysis packages may embed additional information in an *XDI* file by adding extension fields to the header. Extension fields created by applications **should** begin with a form of the application name used in the version line, followed by a separator dot and an additional word. In the sample *XDI* file in Sec. 5, an example of an extension field is `GSE.EXTRA` and takes a value of `config 1`. Here `GSE` denotes the data acquisition software and `EXTRA` denotes a parameter relevant to that software.

Extension field namespaces and tags **should not** collide with the defined namespaces and tags. That is, applications which use extension namepaces or which define new tags in defined namespaces should choose words not already used in the Dictionary of Metadata.

Applications that read *XDI* files **may** attempt to parse the values of extension fields to extract the additional information about the scan. They **should** propagate these fields into output files they create, and **must** propagate any associated version information (see Sec. 3.4.1) if they do so.

Multiple occurrences of the same field are discouraged. When present, the value of the last occurrence (reading linearly from the beginning of the file) **must** be preserved.

4.4 Required elements

The following is a summary of the required elements of an *XDI* file:

1. The first line of the file **must** contain version information. See Sec. 3.4.1.
2. The column containing the abscissa of the data and the units of the abscissa **must** be identified by a header field in the `Column` namespace. For example, if the first column of the data file contains energy in eV units, the following header field **must** appear in the file:

   ```
   # Column.1: energy eV
   ```

3. The column containing the abscissa of the data **must** be the first (left-most) column in the data section.
4. The `Mono.d_spacing` header field **must** be specified if the abscissa is conveyed as monochromator angle.
5. The `Element.symbol` and `Element.edge` headers are **required** in order to definatively identify the XAS measurement.
6. If user comments (see Sec. 3.4.3) are present in the header, the field-end line **must** be present to separate headers from user comments.
7. The header-end separate line **must** be present.
8. A data section **must** be present and each line of data **must** contain the same number of data fields and each field **must** be interpretable as an integer or a floating point number.

All other content is **optional**. When present, certain content **must** meet further requirements as explained in the Dictionary of Metadata.

5 Example *XDI* File

Here is an example of a file conforming to this specification and providing substantial metadata. This was edited by hand from a real data file measured at beamline 13-ID at the APS in 2001. The line beginning `GSE.EXTRA` is an extension fields denoting parameters of the data acquisition system in use at the beamline.

```
# XDI/1.0 GSE/1.0
# Column.1: energy eV
# Column.2: i0
# Column.3: itrans
# Column.4: mutrans
# Element.edge: K
# Element.symbol: Cu
# Scan.edge_energy: 8980.0
# Mono.name: Si 111
# Mono.d_spacing: 3.13553
# Beamline.name: 13ID
# Beamline.collimation: none
# Beamline.focusing: yes
# Beamline.harmonic_rejection: rhodium-coated mirror
```

```
# Facility.name: APS
# Facility.energy: 7.00 GeV
# Facility.xray_source: APS Undulator A
# Scan.start_time: 2001-06-26T22:27:31
# Detector.I0: 10cm  N2
# Detector.I1: 10cm  N2
# Sample.name: Cu
# Sample.prep: Cu metal foil
# GSE.EXTRA:  config 1
# ///
# Cu foil Room Temperature
# measured at beamline 13-ID
#----
# energy i0 itrans mutrans
  8779.0  149013.7  550643.089065  -1.3070486
  8789.0  144864.7  531876.119084  -1.3006104
  8799.0  132978.7  489591.10592  -1.3033816
  8809.0  125444.7  463051.104096  -1.3059724
  8819.0  121324.7  449969.103983  -1.3107085
  8829.0  119447.7  444386.117562  -1.3138152
  8839.0  119100.7  440176.091039  -1.3072055
  8849.0  117707.7  440448.106567  -1.3195882
  8859.0  117754.7  442302.10637  -1.3233895
  8869.0  117428.7  441944.116528  -1.3253521
  8879.0  117383.7  442810.120466  -1.327693
  8889.0  117185.7  443658.11566  -1.3312944
```