punx Documentation

Release 0.gb180e5c.dirty

Pete R. Jemian

Oct 25, 2018

Contents

Python Utilities for NeXus HDF5 files: validation, structure, hierarchy

- Validation of NeXus NXDL files
- Validation of NeXus HDF5 data files
- Display of NeXus HDF5 data file tree structure
- Display of NeXus base class hierarchy (stretch goal, graphical output)

NOTE: project is under initial construction

author Pete R. Jemian

email prjemian@gmail.com

URL http://punx.readthedocs.io

git https://github.com/prjemian/punx

PyPI https://pypi.python.org/pypi/punx

TODO list https://github.com/prjemian/punx/issues

version 0.2.3

release 0.gb180e5c.dirty

published Oct 25, 2018

Use these steps to *install* and try the *demo*:

```
pip install punx
punx demo
```

CHAPTER 1

Contents

1.1 Project Overview

The **punx** program package is easy to use and has several useful modules. The first module to try is *demo*, which validates and prints the structure of a NeXus HDF5 data file from the NeXus documentation.

1.1.1 command line help

```
console> punx -h
usage: punx [-h] [-v]
            {configuration,demonstrate,structure,tree,update,validate} ...

Python Utilities for NeXus HDF5 files version: 0.2.0+9.g31fd4b4.dirty URL:
http://punx.readthedocs.io

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

subcommand:
  valid subcommands

  {configuration,demonstrate,structure,tree,update,validate}
    configuration       show configuration details of punx
    demonstrate         demonstrate HDF5 file validation
    structure           (deprecated) use ``tree``
    tree                show tree structure of HDF5 or NXDL file
    update              update the local cache of NeXus definitions
    validate            validate a NeXus file

Note: It is only necessary to use the first two (or more) characters of any
subcommand, enough that the abbreviation is unique. Such as: ``demonstrate``
can be abbreviated to ``demo`` or even ``de``.
```

1.1.2 Subcommands

User interface: subcommand: demo

The *demo* subcommand is useful to demonstrate HDF5 file validation and to verify correct program operation. It uses an example NeXus HDF5 data file supplied with the *punx* software, the *writer_1_3.hdf5* example from the NeXus manual.

command line help

```
console> punx demo -h
punx demo -h
usage: punx demo [-h]

optional arguments:
  -h, --help  show this help message and exit
```

Examples

One example of how to use **punx** is shown in the *demo* mode. This can be used directly after installing the python package.

Type this command . . . :

```
punx demo
```

. . . and this output will appear on the console, showing a validation of *writer_1_3.hdf5*, an example NeXus HDF5 data file from the NeXus documentation.

```
C:\Users\Pete\Documents\eclipse\punx\src\punx\main.py

!!! WARNING: this program is not ready for distribution.


console> punx validate C:\Users\Pete\Documents\eclipse\punx\src\punx\data\writer_1_3.
↪hdf5
data file: C:\Users\Pete\Documents\eclipse\punx\src\punx\data\writer_1_3.hdf5
NeXus definitions (branch): master, dated 2018-05-16 02:07:48,␣
↪sha=2dc081ee4265eebf80a953080a2ed275c1799a21

findings
============================ ====== ====================================␣
↪===========================================
address                      status test                                comments␣
↪
============================ ====== ====================================␣
↪===========================================
/                            TODO   NeXus base class                    NXroot: more␣
↪validations needed
/                            OK     known NXDL                          NXroot:␣
↪recognized NXDL specification
/                            OK     NeXus base class                    NXroot:␣
↪known NeXus base class
/                            OK     NeXus default plot                  found by v3:␣
↪/Scan/data/counts
```

(continues on next page)

(continued from previous page)

```
/Scan                          TODO   NeXus base class                    NXentry: more validations needed
/Scan                          OK     group in base class                 not defined: NXroot/Scan
/Scan                          OK     known NXDL                          NXentry: recognized NXDL specification
/Scan                          OK     NeXus base class                    NXentry: known NeXus base class
/Scan                          OK     NXDL group in data file             found:  in /Scan/data
/Scan                          NOTE   validItemName                       relaxed pattern: [A-Za-z_][\w_]*
/Scan@NX_class                 OK     validItemName                       pattern: NX.+
/Scan@NX_class                 OK     attribute value                     recognized NXDL base class: NXentry
/Scan@NX_class                 OK     known attribute                     known: NXentry@NX_class
/Scan/data                     TODO   NeXus base class                    NXdata: more validations needed
/Scan/data                     OK     validItemName                       strict pattern: [a-z_][a-z0-9_]*
/Scan/data                     OK     group in base class                 defined: NXentry/data
/Scan/data                     OK     known NXDL                          NXdata: recognized NXDL specification
/Scan/data                     OK     NeXus base class                    NXdata: known NeXus base class
/Scan/data@NX_class            OK     validItemName                       pattern: NX.+
/Scan/data@NX_class            OK     attribute value                     recognized NXDL base class: NXdata
/Scan/data@NX_class            OK     known attribute                     known: NXdata@NX_class
/Scan/data@axes                TODO   attribute value                     implement
/Scan/data@axes                OK     validItemName                       strict pattern: [a-z_][a-z0-9_]*
/Scan/data@axes                OK     known attribute                     known: NXdata@axes
/Scan/data@signal              OK     validItemName                       strict pattern: [a-z_][a-z0-9_]*
/Scan/data@signal              OK     valid name @signal=counts           strict pattern: [a-z_][a-z0-9_]*
/Scan/data@signal              OK     attribute value                     found: @signal=counts
/Scan/data@signal              OK     known attribute                     known: NXdata@signal
/Scan/data@signal              OK     value of @signal                    found: /Scan/data/counts
/Scan/data@signal              OK     NeXus default plot v3, NXdata@signal correct default plot setup in /NXentry/NXdata
/Scan/data@two_theta_indices   TODO   attribute value                     implement
/Scan/data@two_theta_indices   OK     validItemName                       strict pattern: [a-z_][a-z0-9_]*
/Scan/data@two_theta_indices   OK     known attribute                     unknown: NXdata@two_theta_indices
```

(continues on next page)

(continued from previous page)

```
/Scan/data/counts            OK     validItemName                          strict␣
↪pattern: [a-z_][a-z0-9_]*
/Scan/data/counts            OK     field in base class                    not defined:␣
↪NXdata/counts
/Scan/data/counts@units      TODO   attribute value                        implement␣
↪
/Scan/data/counts@units      OK     validItemName                          strict␣
↪pattern: [a-z_][a-z0-9_]*
/Scan/data/two_theta         OK     validItemName                          strict␣
↪pattern: [a-z_][a-z0-9_]*
/Scan/data/two_theta         OK     field in base class                    not defined:␣
↪NXdata/two_theta
/Scan/data/two_theta@units   TODO   attribute value                        implement␣
↪
/Scan/data/two_theta@units   OK     validItemName                          strict␣
↪pattern: [a-z_][a-z0-9_]*
============================ ====== ====================================␣
↪============================================


summary statistics
======== ===== ============================================================ =========
status   count description                                                  (value)
======== ===== ============================================================ =========
OK       33    meets NeXus specification                                    100
NOTE     1     does not meet NeXus specification, but acceptable            75
WARN     0     does not meet NeXus specification, not generally acceptable 25
ERROR    0     violates NeXus specification                                 -10000000
TODO     7     validation not implemented yet                               0
UNUSED   0     optional NeXus item not used in data file                    0
COMMENT  0     comment from the punx source code                            0
OPTIONAL 38    allowed by NeXus specification, not identified               99
         --
TOTAL    79
======== ===== ============================================================ =========

<finding>=99.125000 of 72 items reviewed

console> punx tree C:\Users\Pete\Documents\eclipse\punx\src\punx\data\writer_1_3.hdf5
C:\Users\Pete\Documents\eclipse\punx\src\punx\data\writer_1_3.hdf5 : NeXus data file
  Scan:NXentry
    @NX_class = NXentry
    data:NXdata
      @NX_class = NXdata
      @signal = counts
      @axes = two_theta
      @two_theta_indices = 0
      counts:NX_INT32[31] = [1037, 1318, 1704, '...', 1321]
        @units = counts
      two_theta:NX_FLOAT64[31] = [17.92608, 17.92591, 17.92575, '...', 17.92108]
        @units = degrees
```

Problems when running the demo

Sometimes, problems happen when running the demo. In this section are some common problems encountered and what was done to resolve them.

Cannot reach GitHub

See *GitHub API rate limit exceeded*

User interface: subcommand: hierarchy

-tba-

User interface: subcommand: tree

show tree structure of HDF5 or NXDL file

command line help

```
console> punx tree -h
usage: punx tree [-h] [-a] [-m MAX_ARRAY_ITEMS] infile

positional arguments:
  infile                HDF5 or NXDL file name

optional arguments:
  -h, --help            show this help message and exit
  -a                    Do not print attributes of HDF5 file structure
  -m MAX_ARRAY_ITEMS, --max_array_items MAX_ARRAY_ITEMS
                        maximum number of array items to be shown
```

Examples

–tba–

User interface: subcommand: update

punx keeps a local copy of the NeXus definition files. The originals of these files are located on GitHub.

+.. caution:: The update process is being refactored, this may not work correctly now

To *update* the local cache of NeXus definitions, run:

```
console> punx update

INFO: get repo info: https://api.github.com/repos/nexusformat/definitions/commits
INFO: git sha: 8eb46e229f900d1e77e37c4b6ee6e0405efe099c
INFO: git iso8601: 2016-06-17T18:05:28Z
INFO: not updating NeXus definitions files
```

This shows the current cache was up to date. Here's an example when the source cache needed to be updated:

```
console> punx update

INFO: get repo info: https://api.github.com/repos/nexusformat/definitions/commits
INFO: git sha: 8eb46e229f900d1e77e37c4b6ee6e0405efe099c
INFO: git iso8601: 2016-06-17T18:05:28Z
INFO: updating NeXus definitions files
INFO: download: https://github.com/nexusformat/definitions/archive/master.zip
INFO: extract ZIP to: C:/Users/Pete/Documents/eclipse/punx/src/punx/cache
```

command line help

```
console> punx update -h
punx update -h
usage: punx update [-h] [-f]

optional arguments:
  -h, --help   show this help message and exit
  -f, --force  force update (if GitHub available)
```

Examples

–tba–

Problems

GitHub API rate limit exceeded

A common problem happens when updating the NXDL definitions from GitHub. Here's what it looks like:

```
$ python ./src/punx/main.py update --force

('INFO:', 'get repo info: https://api.github.com/repos/nexusformat/definitions/commits
↪')

Traceback (most recent call last):

  File "./src/punx/main.py", line 416, in <module>

    main()

  File "./src/punx/main.py", line 412, in main

    args.func(args)

  File "./src/punx/main.py", line 170, in func_update

    cache.update_NXDL_Cache(force_update=args.force)

  File "/home/travis/build/prjemian/punx/src/punx/cache.py", line 257, in update_NXDL_
↪Cache

    info = __get_github_info__()     # check with GitHub first
```

(continues on next page)

(continued from previous page)

```
  File "/home/travis/build/prjemian/punx/src/punx/cache.py", line 246, in __get_
↪github_info__

    punx.GITHUB_NXDL_REPOSITORY)

  File "/home/travis/build/prjemian/punx/src/punx/cache.py", line 228, in␣
↪githubMasterInfo

    raise punx.CannotUpdateFromGithubNow(msg)

punx.CannotUpdateFromGithubNow: API rate limit exceeded for nn.nn.nn.nn.
(But here's the good news: Authenticated requests get a higher rate limit.
Check out the documentation for more details.)
```

GitHub imposes a limit on the number of unauthenticated downloads per hour[1]. You can check your rate limit status[2]. Mostly, this means try again later.

A GitHub issue has been raised to resolve this for the **punx** project.[3]

Validation

Data File Validation

NeXus HDF5 data files can have significant structure and variation. It can be a challenge to determine that a given file is compliant with any of the rules specified in the *NeXus definitions* (here, we refer to the the applicable NXDL files and NeXus XML Schema in aggregate as the *NeXus definitions*). Additionally, there are various releases and version of the NeXus standard.

The first test for any file to be considered a NeXus data file is whether or not the file is a valid HDF5 file. If the file is not HDF5, it is not a valid NeXus HDF5 data file.

General

In general, validation of data files proceeds through several steps:

1. Is file HDF5?
2. Does file contain one or more **NXentry**[1] groups?
3. Test the *NeXus definitions* against the file
4. Does the file define a default plot in each **NXdata** group? (recommended but no longer required)
5. Does the file define a path to the default plot? (recommended but no longer required)
6. Is the file a NeXus HDF5 data file?

Is file HDF5?

This is a simple test and is handled by the *h5py* package.

[1] "The rate limit allows you to make up to 60 requests per hour, associated with your IP address", https://developer.github.com/v3/#rate-limiting
[2] Status of GitHub API Rate Limit: https://developer.github.com/v3/rate_limit/
[3] *update: cannot download NXDL files from GitHub #64,* https://github.com/prjemian/punx/issues/64
[1] http://download.nexusformat.org/doc/html/classes/base_classes/NXentry.html

Test *NeXus definitions* against the data file

The *NeXus definitions* provide specifications for what should be found in a NeXus data file and where it should be found. Some itmes are optional and some items may be repeated.

In NeXus data files, the structure is defined by adding *NX_class* attributes to each of the groups. This structure must match what is defined in the NXDL file for that group.

Groups must be one of the defined base classes (or contributed definitions intended for use as a base class, but this is rare)

Test each NXentry group agains the *NeXus definitions*

In a NeXus data file, there are one more more **NXentry** groups. Validation proceeds by walking through each of the groups that define a *NX_class* attribute using the matching base class (or contributed definition).

NeXus application definitions are a special case of **NXentry** (or **NXsubentry**) group. If a group's *NX_class* attribute has the value *NXentry* or *NXsubentry*, that group must contain a *definition* field. The value of this *definition* field gives the name of the application definition to which this group (and all its subgroups) must comply. It is recommended to use *NXsubentry* to contain an application definition.

Base classes are the building blocks of the NeXus structure. Application definitions differ from **NXentry** and **NXsubentry** in one important aspect: content specified in an application definition is *required*, by default. In base classes, content is *optional* by default. Contributed definitions include propositions from the community for NeXus base classes or application definitions, as well as other NXDL files for long-term archival by NeXus. Consider the contributed definitions as either *candidates* for inclusion in the NeXus standard or a special case not for general use.

Details

–tba–

Parsing the XML Schema

The XML Schema defines the constructs of the NXDL language, the various enumerations, and the default values when the constructs are used in base classes or application definitions.

Parsing the NXDL files

–tba–

Application Definitions

–tba–

NXDL File Validation

NXDL files must adhere to the specifications of the NeXus XML Schema, as defined in *nxdl.xsd* and *nxdlTypes.xsd*.

Caution: TODO: citation needed

Any NXDL file may be validated using the Linux command line tool `xmllint`. Such as:

```
user@host ~ $  xmllint --noout --schema nxdl.xsd base_classes/NXentry.nxdl.xml
base_classes/NXentry.nxdl.xml validates
```

Validation is the process of comparing an object with a standard. An important aspect of validation is the report of each aspect tested and whether or not it complies with the standard. This is a useful and necessary step when composing NeXus HDF5 data files or software that will read NeXus data files and when building NeXus Definition Language (NXDL) files.

In NeXus, three basic types of object can be validated:

- *HDF5 data files* must comply with the specifications set forth in the applicable NeXus base classes, application definitions, and contributed definitions.
- *NeXus NXDL files* must comply with the XML Schema files *nxdl.xsd* and *nxdlTypes.xsd*.
- **XML Schema files** must comply with the rules defined by the WWW3 consortium. TODO: citation needed.

User interface: subcommand: validate

validate a NeXus file

command line help

```
usage: punx validate [-h] [--report REPORT] [-l [LOGFILE]] [-i INTEREST]
                     infile

positional arguments:
  infile                HDF5 or NXDL file name

optional arguments:
  -h, --help            show this help message and exit
  --report REPORT       select which validation findings to report, choices:
                        COMMENT,ERROR,NOTE,OK,TODO,UNUSED,WARN
  -l [LOGFILE], --logfile [LOGFILE]
                        log output to file (default: no log file)
  -i INTEREST, --interest INTEREST
                        logging interest level (1 - 50), default=1 (Level 1)
```

The **REPORT** findings are as presented in the table above for each validation step.

The logging **INTEREST** levels are for output from the program,

Examples

–tba–

punx uses a subcommand structure to provide several different modules under one identifiable program. These are invoked using commands of the form:

```
punx <subcommand> <other parameters>
```

where <*subcommand*> is chosen from this table:

subcommand	brief description
configuration	show internal punx configuration
demonstrate	demonstrate HDF5 file validation
hierarchy	show NeXus base class hierarchy (not implemented yet)
structure	(deprecated) use tree
tree	show tree structure of HDF5 or NXDL file
update	update the local cache of NeXus definitions
validate	validate a NeXus file

and the <*other parameters*> are desribed by the help for each subcommand:

```
punx <subcommand> -h
```

Example[1]

```
console> punx val -h
usage: punx validate [-h] [--report REPORT] infile

positional arguments:
  infile           HDF5 or NXDL file name

optional arguments:
  -h, --help       show this help message and exit
  --report REPORT  select which validation findings to report, choices:
                   COMMENT,ERROR,NOTE,OK,OPTIONAL,TODO,UNUSED,WARN
```

1.2 Installation

Released versions of punx are available on PyPI.

If you have `pip` installed, then you can install:

```
$ pip install punx
```

The latest development versions of punx can be downloaded from the GitHub repository listed above:

```
$ cd /some/directory
$ git clone http://github.com/prjemian/punx.git
```

To install in the standard Python location:

```
$ cd punx
$ pip install .
# -or-
$ python setup.py install
```

[1] tip: Subcommands may be shortened.
It is only necessary to use the first two (or more) characters of any subcommand, enough that the short version remains unique and could not be misinterpreted as another subcommand. The program imposes a minimum limit of at least 2-characters.
Such as: `demonstrate` can be abbreviated to `demo` or even `de`.

To install in user's home directory:

```
$ python setup.py install --user
```

To install in an alternate location:

```
$ python setup.py install --prefix=/path/to/installation/dir
```

1.2.1 Updating

pip If you have installed previously with *pip*:

```
$ pip install -U --no-deps punx
```

git assuming you have cloned as shown above:

```
$ cd /some/directory/punx
$ git pull
$ pip install -U --no-deps .
```

1.2.2 Required Packages

It may be necessary to install some prerequisite packages in your python installation. If you are using an Anaconda python distribution, it is advised to install these pre-requisites using *conda* rather than *pip*. The pre-requisites include:

- h5py
- lxml
- numpy
- Qt and PyQt (either v4 or v5)
- requests
- PyGithub

See your distribution's documentation for how to install these. With Anaconda, use:

```
conda install h5py lxml numpy Qt=5 PyQt=5 requests
pip install PyGitHub pyRestTable
```

Package	URL
h5py	http://www.h5py.org
lxml	http://lxml.de
numpy	http://numpy.scipy.org
PyGithub	https://github.com/PyGithub/PyGithub
PyQt4	https://riverbankcomputing.com/software/pyqt/intro
requests	http://docs.python-requests.org

1.2.3 Optional Packages

Package	URL
pyRestTable	http://pyresttable.readthedocs.io

The *pyRestTable* package is used for various reports in the punx application. If using the punx package as a library and developing your own custom reporting, this package is not required.

1.3 Change History

1.3.1 Production

–none–

1.3.2 Development

0.2.0 2018-07-02 – first tag after major refactor (#72, #105)

0.1.9 2017-07-09 – last tag before major refactor (#72), no changes here since 2017-03-31

0.1.8 2017-03-12 – package .json file in the cache file sets

0.1.7 2017-03-11 – NeXus def 3.2 bundled into repo now

0.1.4 2016-12-09 – validation reports sorted by HDF5 address

0.1.3 2016-12-07 – Py2 & Py3: passes all unit tests

0.1.2 2016-11-21 – unit tests added for reports

0.0.9 2016-06-29 – retry failed https requests to GitHub and cleanup a QString

0.0.8 2016-06-29 – refactor update procedure

0.0.7 2016-06-27 – add “report” arguments to “demo” subcommand

0.0.6 2016-06-22 – resolved some UnicodeDecodeError exceptions

0.0.5 2016-06-20 – added subcommand shortcuts and logging

0.0.4 2016-06-17 – work-in-progress to test installation with remote user

0.0.3 2016-06-11 – basic UI established, **demo** command added

0.0.2 2016-06-11 – basic UI established

0.0.1 2016-06-10 – basic functions

started 2016-05-20 – initial project creation

1.4 License

```
Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound␣
↪by the terms and conditions of this Creative Commons Attribution 4.0 International␣
↪Public License ("Public License"). To the extent this Public License may be␣
↪interpreted as a contract, You are granted the Licensed Rights in consideration of␣
↪Your acceptance of these terms and conditions, and the Licensor grants You such␣
↪rights in consideration of benefits the Licensor receives from making the Licensed␣
↪Material available under these terms and conditions.
```

(continues on next page)

(continued from previous page)

```
Section 1 -- Definitions.

   Adapted Material means material subject to Copyright and Similar Rights that is␣
↪derived from or based upon the Licensed Material and in which the Licensed Material␣
↪is translated, altered, arranged, transformed, or otherwise modified in a manner␣
↪requiring permission under the Copyright and Similar Rights held by the Licensor.␣
↪For purposes of this Public License, where the Licensed Material is a musical work,␣
↪performance, or sound recording, Adapted Material is always produced where the␣
↪Licensed Material is synched in timed relation with a moving image.
   Adapter's License means the license You apply to Your Copyright and Similar␣
↪Rights in Your contributions to Adapted Material in accordance with the terms and␣
↪conditions of this Public License.
   Copyright and Similar Rights means copyright and/or similar rights closely␣
↪related to copyright including, without limitation, performance, broadcast, sound␣
↪recording, and Sui Generis Database Rights, without regard to how the rights are␣
↪labeled or categorized. For purposes of this Public License, the rights specified␣
↪in Section 2(b)(1)-(2) are not Copyright and Similar Rights.
   Effective Technological Measures means those measures that, in the absence of␣
↪proper authority, may not be circumvented under laws fulfilling obligations under␣
↪Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or␣
↪similar international agreements.
   Exceptions and Limitations means fair use, fair dealing, and/or any other␣
↪exception or limitation to Copyright and Similar Rights that applies to Your use of␣
↪the Licensed Material.
   Licensed Material means the artistic or literary work, database, or other␣
↪material to which the Licensor applied this Public License.
   Licensed Rights means the rights granted to You subject to the terms and␣
↪conditions of this Public License, which are limited to all Copyright and Similar␣
↪Rights that apply to Your use of the Licensed Material and that the Licensor has␣
↪authority to license.
   Licensor means the individual(s) or entity(ies) granting rights under this Public␣
↪License.
   Share means to provide material to the public by any means or process that␣
↪requires permission under the Licensed Rights, such as reproduction, public display,
↪ public performance, distribution, dissemination, communication, or importation,␣
↪and to make material available to the public including in ways that members of the␣
↪public may access the material from a place and at a time individually chosen by␣
↪them.
   Sui Generis Database Rights means rights other than copyright resulting from␣
↪Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on␣
↪the legal protection of databases, as amended and/or succeeded, as well as other␣
↪essentially equivalent rights anywhere in the world.
   You means the individual or entity exercising the Licensed Rights under this␣
↪Public License. Your has a corresponding meaning.

Section 2 -- Scope.

   License grant.
       Subject to the terms and conditions of this Public License, the Licensor␣
↪hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive,␣
↪irrevocable license to exercise the Licensed Rights in the Licensed Material to:
           reproduce and Share the Licensed Material, in whole or in part; and
           produce, reproduce, and Share Adapted Material.
       Exceptions and Limitations. For the avoidance of doubt, where Exceptions and␣
↪Limitations apply to Your use, this Public License does not apply, and You do not␣
↪need to comply with its terms and conditions.
       Term. The term of this Public License is specified in Section 6(a).
```

(continues on next page)

(continued from previous page)

```
        Media and formats; technical modifications allowed. The Licensor authorizes␣
↪You to exercise the Licensed Rights in all media and formats whether now known or␣
↪hereafter created, and to make technical modifications necessary to do so. The␣
↪Licensor waives and/or agrees not to assert any right or authority to forbid You␣
↪from making technical modifications necessary to exercise the Licensed Rights,␣
↪including technical modifications necessary to circumvent Effective Technological␣
↪Measures. For purposes of this Public License, simply making modifications␣
↪authorized by this Section 2(a)(4) never produces Adapted Material.
        Downstream recipients.
            Offer from the Licensor -- Licensed Material. Every recipient of the␣
↪Licensed Material automatically receives an offer from the Licensor to exercise the␣
↪Licensed Rights under the terms and conditions of this Public License.
            No downstream restrictions. You may not offer or impose any additional or␣
↪different terms or conditions on, or apply any Effective Technological Measures to,␣
↪the Licensed Material if doing so restricts exercise of the Licensed Rights by any␣
↪recipient of the Licensed Material.
        No endorsement. Nothing in this Public License constitutes or may be␣
↪construed as permission to assert or imply that You are, or that Your use of the␣
↪Licensed Material is, connected with, or sponsored, endorsed, or granted official␣
↪status by, the Licensor or others designated to receive attribution as provided in␣
↪Section 3(a)(1)(A)(i).

    Other rights.
        Moral rights, such as the right of integrity, are not licensed under this␣
↪Public License, nor are publicity, privacy, and/or other similar personality rights;
↪ however, to the extent possible, the Licensor waives and/or agrees not to assert␣
↪any such rights held by the Licensor to the limited extent necessary to allow You␣
↪to exercise the Licensed Rights, but not otherwise.
        Patent and trademark rights are not licensed under this Public License.
        To the extent possible, the Licensor waives any right to collect royalties␣
↪from You for the exercise of the Licensed Rights, whether directly or through a␣
↪collecting society under any voluntary or waivable statutory or compulsory␣
↪licensing scheme. In all other cases the Licensor expressly reserves any right to␣
↪collect such royalties.

Section 3 -- License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following␣
↪conditions.

    Attribution.

        If You Share the Licensed Material (including in modified form), You must:
            retain the following if it is supplied by the Licensor with the Licensed␣
↪Material:
                identification of the creator(s) of the Licensed Material and any␣
↪others designated to receive attribution, in any reasonable manner requested by the␣
↪Licensor (including by pseudonym if designated);
                a copyright notice;
                a notice that refers to this Public License;
                a notice that refers to the disclaimer of warranties;
                a URI or hyperlink to the Licensed Material to the extent reasonably␣
↪practicable;
            indicate if You modified the Licensed Material and retain an indication␣
↪of any previous modifications; and
            indicate the Licensed Material is licensed under this Public License, and␣
↪include the text of, or the URI or hyperlink to, this Public License.
```

(continues on next page)

(continued from previous page)

```
       You may satisfy the conditions in Section 3(a)(1) in any reasonable manner␣
↪based on the medium, means, and context in which You Share the Licensed Material.␣
↪For example, it may be reasonable to satisfy the conditions by providing a URI or␣
↪hyperlink to a resource that includes the required information.
       If requested by the Licensor, You must remove any of the information required␣
↪by Section 3(a)(1)(A) to the extent reasonably practicable.
       If You Share Adapted Material You produce, the Adapter's License You apply␣
↪must not prevent recipients of the Adapted Material from complying with this Public␣
↪License.

Section 4 -- Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use␣
↪of the Licensed Material:

   for the avoidance of doubt, Section 2(a)(1) grants You the right to extract,␣
↪reuse, reproduce, and Share all or a substantial portion of the contents of the␣
↪database;
   if You include all or a substantial portion of the database contents in a␣
↪database in which You have Sui Generis Database Rights, then the database in which␣
↪You have Sui Generis Database Rights (but not its individual contents) is Adapted␣
↪Material; and
   You must comply with the conditions in Section 3(a) if You Share all or a␣
↪substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your␣
↪obligations under this Public License where the Licensed Rights include other␣
↪Copyright and Similar Rights.

Section 5 -- Disclaimer of Warranties and Limitation of Liability.

   Unless otherwise separately undertaken by the Licensor, to the extent possible,␣
↪the Licensor offers the Licensed Material as-is and as-available, and makes no␣
↪representations or warranties of any kind concerning the Licensed Material, whether␣
↪express, implied, statutory, or other. This includes, without limitation,␣
↪warranties of title, merchantability, fitness for a particular purpose, non-
↪infringement, absence of latent or other defects, accuracy, or the presence or␣
↪absence of errors, whether or not known or discoverable. Where disclaimers of␣
↪warranties are not allowed in full or in part, this disclaimer may not apply to You.
   To the extent possible, in no event will the Licensor be liable to You on any␣
↪legal theory (including, without limitation, negligence) or otherwise for any␣
↪direct, special, indirect, incidental, consequential, punitive, exemplary, or other␣
↪losses, costs, expenses, or damages arising out of this Public License or use of␣
↪the Licensed Material, even if the Licensor has been advised of the possibility of␣
↪such losses, costs, expenses, or damages. Where a limitation of liability is not␣
↪allowed in full or in part, this limitation may not apply to You.

   The disclaimer of warranties and limitation of liability provided above shall be␣
↪interpreted in a manner that, to the extent possible, most closely approximates an␣
↪absolute disclaimer and waiver of all liability.

Section 6 -- Term and Termination.

   This Public License applies for the term of the Copyright and Similar Rights␣
↪licensed here. However, if You fail to comply with this Public License, then Your␣
↪rights under this Public License terminate automatically.
```

(continues on next page)

(continued from previous page)

```
    Where Your right to use the Licensed Material has terminated under Section 6(a),␣
↪it reinstates:
        automatically as of the date the violation is cured, provided it is cured␣
↪within 30 days of Your discovery of the violation; or
        upon express reinstatement by the Licensor.
    For the avoidance of doubt, this Section 6(b) does not affect any right the␣
↪Licensor may have to seek remedies for Your violations of this Public License.
    For the avoidance of doubt, the Licensor may also offer the Licensed Material␣
↪under separate terms or conditions or stop distributing the Licensed Material at␣
↪any time; however, doing so will not terminate this Public License.
    Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 -- Other Terms and Conditions.

    The Licensor shall not be bound by any additional or different terms or␣
↪conditions communicated by You unless expressly agreed.
    Any arrangements, understandings, or agreements regarding the Licensed Material␣
↪not stated herein are separate from and independent of the terms and conditions of␣
↪this Public License.

Section 8 -- Interpretation.

    For the avoidance of doubt, this Public License does not, and shall not be␣
↪interpreted to, reduce, limit, restrict, or impose conditions on any use of the␣
↪Licensed Material that could lawfully be made without permission under this Public␣
↪License.
    To the extent possible, if any provision of this Public License is deemed␣
↪unenforceable, it shall be automatically reformed to the minimum extent necessary␣
↪to make it enforceable. If the provision cannot be reformed, it shall be severed␣
↪from this Public License without affecting the enforceability of the remaining␣
↪terms and conditions.
    No term or condition of this Public License will be waived and no failure to␣
↪comply consented to unless expressly agreed to by the Licensor.
    Nothing in this Public License constitutes or may be interpreted as a limitation␣
↪upon, or waiver of, any privileges and immunities that apply to the Licensor or You,
↪ including from the legal processes of any jurisdiction or authority.
```

1.5 Cache : `cache_manager`

Hierarchy:

- *punx.nxdl_manager*
- *punx.schema_manager*
- *punx.cache_manager*
- *punx.github_handler*

1.5.1 source code documentation

manages the NXDL cache directories of this project

A key component necessary to validate both NeXus data files and NXDL class files is a current set of the NXDL definitions.

There are two cache directories:

- the source cache
- the user cache

Within each of these cache directories, there may be one or more subdirectories, each containing the NeXus definitions subdirectories and files (`*.xml`, `*.xsl`, & `*.xsd`) of a specific branch, release, tag, or commit hash from the NeXus definitions repository.

source cache contains default set of NeXus NXDL files

user cache contains additional set(s) of NeXus NXDL files, installed by user

The *cache_manager* calls the *github_handler* and is called by *schema_manager* and *nxdl_manager*.

Public interface

CacheManager()	manager both source and user caches

Internal interface

get_short_sha(full_sha)	return the first few unique characters of the git commit hash (SHA)
read_json_file(filename)	read a structured object from the JSON file `file_name`
write_json_file(filename, obj)	write the structured `obj` to the JSON file `file_name`
should_extract_this(item, ...)	decide if this item should be extracted from the ZIP download
should_avoid_download(grr, path)	decide if the download should be avoided (True: avoid, False: download)
extract_from_download(grr, path)	download & extract NXDL files from `grr` into a subdirectory of `path`
table_of_caches()	return a pyRestTable table describing all known file sets in both source and user caches
Base_Cache	provides comon methods to get the QSettings path and file name
SourceCache()	manage the source directory cache of NXDL files
UserCache()	manage the user directory cache of NXDL files
NXDL_File_Set	describe a single set of NXDL files

class punx.cache_manager.**Base_Cache**

provides comon methods to get the QSettings path and file name

find_all_file_sets()	index all NXDL file sets in this cache
fileName()	full path of the QSettings file
path()	directory containing the QSettings file
cleanup()	removes any temporary directories

cleanup()

removes any temporary directories

fileName()

full path of the QSettings file

find_all_file_sets()
index all NXDL file sets in this cache

path()
directory containing the QSettings file

class punx.cache_manager.**CacheManager**
manager both source and user caches

install_NXDL_file_set(grr[, user_cache, ...])	using *ref* as a name, get the se of NXDL files from the NeXus GitHub
select_NXDL_file_set([ref])	return the named self.default_file_set instance or raise KeyError exception if unknown
find_all_file_sets()	return dictionary of all NXDL file sets in both source & user caches
cleanup()	removes any temporary directories

cleanup()
removes any temporary directories

find_all_file_sets()
return dictionary of all NXDL file sets in both source & user caches

install_NXDL_file_set(*grr*, *user_cache=True*, *ref=None*, *force=False*)
using *ref* as a name, get the se of NXDL files from the NeXus GitHub

Parameters

- **grr** (*obj*) – instance of GitHub_Repository_Reference
- **user_cache** (*bool*) – True: use user cache, '' False'': use source cache (default)
- **ref** (*str*) – name to use when requesting from GitHub, (*master*, commit hash such as *abc1234*, branch name, release name such as *v3.2*, or tag name)
- **force** (*bool*) – update if installed is not the same SHA

select_NXDL_file_set(*ref=None*)
return the named self.default_file_set instance or raise KeyError exception if unknown

Return obj

table_of_caches()
return a pyRestTable table describing all known file sets in both source and user caches

Returns obj instance of pyRestTable.Table with all known file sets

Example:

```
============= ======= ====== =================== =======
 ↪====================================
NXDL file set type    cache  date & time         commit  path
============= ======= ====== =================== =======
 ↪====================================
v3.2          tag     source 2017-01-18 23:12:44 e888dac /home/user/punx/src/
 ↪punx/cache/v3.2
NXroot-1.0    tag     user   2016-10-24 14:58:10 e0ad63d /home/user/.config/
 ↪punx/NXroot-1.0
master        branch  user   2016-12-20 18:30:29 85d056f /home/user/.config/
 ↪punx/master
```

(continues on next page)

(continued from previous page)

```
Schema-3.3    release user   2017-05-02 12:33:19 4aa4215 /home/user/.config/
↪punx/Schema-3.3
a4fd52d       commit  user   2016-11-19 01:07:45 a4fd52d /home/user/.config/
↪punx/a4fd52d
============= ======= ====== ================== =======
↪===================================
```

class punx.cache_manager.**NXDL_File_Set**
describe a single set of NXDL files

class punx.cache_manager.**SourceCache**
manage the source directory cache of NXDL files

class punx.cache_manager.**UserCache**
manage the user directory cache of NXDL files

punx.cache_manager.**extract_from_download**(*grr*, *path*)
download & extract NXDL files from `grr` into a subdirectory of `path`

USAGE:

```
grr = github_handler.GitHub_Repository_Reference()
grr.connect_repo()
if grr.request_info() is not None:
    extract_from_download(grr, cache_directory)
```

punx.cache_manager.**get_short_sha**(*full_sha*)
return the first few unique characters of the git commit hash (SHA)

Parameters **full_sha** (*str*) – hash code from Github

punx.cache_manager.**read_json_file**(*filename*)
read a structured object from the JSON file `file_name`

See https://docs.python.org/3.5/library/json.html#json.loads

punx.cache_manager.**should_avoid_download**(*grr*, *path*)
decide if the download should be avoided (True: avoid, False: download)

Return bool

punx.cache_manager.**should_extract_this**(*item*, *NXDL_file_endings_list*, *allowed_parent_directories*)
decide if this item should be extracted from the ZIP download

Return bool

punx.cache_manager.**table_of_caches**()
return a pyRestTable table describing all known file sets in both source and user caches

Returns obj instance of pyRestTable.Table with all known file sets

Example:

```
============= ======= ====== ================== =======
↪===================================
NXDL file set type    cache  date & time         commit  path
============= ======= ====== ================== =======
↪===================================
v3.2          tag     source 2017-01-18 23:12:44 e888dac /home/user/punx/src/punx/
↪cache/v3.2
```

(continues on next page)

(continued from previous page)

```
NXroot-1.0    tag     user   2016-10-24 14:58:10 e0ad63d /home/user/.config/punx/
↪NXroot-1.0
master        branch  user   2016-12-20 18:30:29 85d056f /home/user/.config/punx/
↪master
Schema-3.3    release user   2017-05-02 12:33:19 4aa4215 /home/user/.config/punx/
↪Schema-3.3
a4fd52d       commit  user   2016-11-19 01:07:45 a4fd52d /home/user/.config/punx/
↪a4fd52d
============ ======= ====== ================== =======␣
↪====================================
```

punx.cache_manager.**write_json_file**(*filename*, *obj*)
write the structured `obj` to the JSON file `file_name`

See https://docs.python.org/3.5/library/json.html#json.dumps

1.6 Findings : `finding`

Each validation test of an object in the NeXus data file should produce a *finding*.

1.6.1 source code documentation

document each item during validation

Finding(h5_address, test_name, status, comment)	a single reported observation while validating
VALID_STATUS_DICT	dictionary (by names) of all available validations

class punx.finding.**Finding**(*h5_address*, *test_name*, *status*, *comment*)
a single reported observation while validating

Parameters

- **h5_address** (*str*) – address of h5py item
- **test_name** (*str*) – short description of the test
- **status** (*obj*) – one of: OK NOTE WARN ERROR TODO COMMENT OPTIONAL UNUSED
- **comment** (*str*) – description

make_md5()
make a unique hash for this finding

punx.finding.**VALID_STATUS_DICT = {'COMMENT': <punx.finding.ValidationResultStatus object>,**
dictionary (by names) of all available validations

class punx.finding.**ValidationResultStatus**(*key*, *value*, *color*, *description*)
summary result of a Finding

Parameters

- **key** (*str*) – short name
- **color** (*str*) – suggested color for GUI
- **description** (*str*) – one-line summary

1.7 GitHub : github_handler

The github_handler module handles all communications with the NeXus GitHub repository.

If the user has provided GitHub credentials (username and password) in a *__github_creds__.txt* file in the source code directory, then the access will be through the Github Basic Authentication interface. This is helpful since the GitHub API Rate Limit allows for only a few downloads through the API per hour if using unauthenticated access.

1.7.1 source code documentation

manages the communications with GitHub

GitHub_Repository_Reference()	all information necessary to describe and download a repository branch, release, tag, or SHA hash

USAGE:

```
grr = punx.github_handler.GitHub_Repository_Reference()
grr.connect_repo()
if grr.request_info(u'v3.2') is not None:
    d = grr.download()
```

class punx.github_handler.**GitHub_Repository_Reference**

all information necessary to describe and download a repository branch, release, tag, or SHA hash

ROUTINES

connect_repo([repo_name])	connect with the GitHub repository
request_info([ref])	request download information about ref
download()	download the NXDL definitions described by ref

See https://github.com/PyGithub/PyGithub/tree/master/github

connect_repo(*repo_name=None*)

connect with the GitHub repository

Parameters **repo_name** (*str*) – name of repository in https://github.com/nexusformat (default: *definitions*)

Returns bool True if using GitHub credentials

download()

download the NXDL definitions described by ref

get_branch(*ref=u'master'*)

learn the download information about the named branch

Parameters **ref** (*str*) – name of branch in repository

get_commit(*ref=u'a4fd52d'*)

learn the download information about the referenced commit

Parameters **ref** (*str*) – name of SHA hash, first unique characters are sufficient, usually 7 or less

get_release(*ref=u'v2018.5'*)

learn the download information about the named release

Parameters **ref** (*str*) – name of release in repository

get_tag (*ref=u'Schema-3.3'*)
learn the download information about the named tag

Parameters **ref** (*str*) – name of tag in repository

request_info (*ref=None*)
request download information about `ref`

Parameters **ref** (*str*) – name of branch, release, tag, or SHA hash (default: *v3.2*)

download URLs

- base: https://github.com
- master: https://github.com/nexusformat/definitions/archive/master.zip
- branch (www_page_486): https://github.com/nexusformat/definitions/archive/www_page_486.zip
- hash (83ce630): https://github.com/nexusformat/definitions/archive/83ce630.zip
- release (v3.2): see hash c0b9500
- tag (NXcanSAS-1.0): see hash 83ce630

`punx.github_handler.`**get_BasicAuth_credentials** (*creds_file_name=None*)
get the Github Basic Authentication credentials from a local file

GitHub requests can use *Basic Authentication* if the credentials (username and password) are provided in the local file `__github_creds__.txt` which is placed in the same directory as this file. The credentials file is not placed under version control since it has GitHub credentials. If found, the file is parsed for `username password` as shown below. Be sure to make the file readable only by the user and not others.

1.8 HDF5 Data File Tree Structure : `h5tree`

Print the tree structure of any HDF5 file.

Note: The *tree* subcommand replaces the now-legacy *structure* subcommand and also [replaces](https://github.com/prjemian/spec2nexus/issues/70) the *h5toText* program from the [*spec2nexus*](https://github.com/prjemian/spec2nexus) project.

1.8.1 How to use h5tree

Print the HDF5 tree of a file:

```
$ punx tree  path/to/file/hdf5/file.hdf5
```

the help message:

```
  [linux,512]$ punx tree -h
  usage: punx tree [-h] [-a] [-m MAX_ARRAY_ITEMS] infile

  positional arguments:
    infile                HDF5 or NXDL file name

  optional arguments:
    -h, --help            show this help message and exit
    -a                    Do not print attributes of HDF5 file structure
```

(continues on next page)

(continued from previous page)

```
    -m MAX_ARRAY_ITEMS, --max_array_items MAX_ARRAY_ITEMS
                          maximum number of array items to be shown
```

1.8.2 Example

Here's an example from a test data file (**writer_1_3.h5** from the NeXus documentation[1]):

```
[linux,512]$ punx tree data/writer_1_3.hdf5
data/writer_1_3.hdf5 : NeXus data file
  Scan:NXentry
    @NX_class = NXentry
    data:NXdata
      @NX_class = NXdata
      @signal = counts
      @axes = two_theta
      @two_theta_indices = [0]
      counts:NX_INT32[31] = [1037, 1318, 1704, '...', 1321]
        @units = counts
      two_theta:NX_FLOAT64[31] = [17.926079999999999, 17.925909999999998, 17.
↪925750000000001, '...', 17.92108]
        @units = degrees
```

1.8.3 source code documentation

Describe the tree structure of any HDF5 file

Hdf5TreeView(filename)	Describe the tree structure of any HDF5 file

class punx.h5tree.**Hdf5TreeView**(*filename*)

Describe the tree structure of any HDF5 file

Example usage showing default display:

```
mc = Hdf5TreeView(filename)
mc.array_items_shown = 5
show_attributes = False
txt = mc.report(show_attributes)
```

report(*show_attributes=True*)

return the structure of the HDF5 file in a list of strings

The work of parsing the datafile is done in this method.

1.9 User interface : `main`

Provides the user interface(s) to the punx program.

[1] writer_1_3 from NeXus: http://download.nexusformat.org/doc/html/examples/h5py/writer_1_3.html

1.9.1 source code settings

Python Utilities for NeXus HDF5 files

main user interface file

Usage

```
console> punx -h
usage: punx [-h] [-v]
            {configuration,demonstrate,structure,tree,update,validate} ...

Python Utilities for NeXus HDF5 files version: 0.2.0+9.g31fd4b4.dirty URL:
http://punx.readthedocs.io

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

subcommand:
  valid subcommands

  {configuration,demonstrate,structure,tree,update,validate}
    configuration       show configuration details of punx
    demonstrate         demonstrate HDF5 file validation
    structure           structure command deprecated. Use ``tree`` instead
    tree                show tree structure of HDF5 or NXDL file
    update              update the local cache of NeXus definitions
    validate            validate a NeXus file

Note: It is only necessary to use the first two (or more) characters of any
subcommand, enough that the abbreviation is unique. Such as: ``demonstrate``
can be abbreviated to ``demo`` or even ``de``.
```

main()	
MyArgumentParser([prog, usage, description, ...])	override standard ArgumentParser to enable shortcut feature
parse_command_line_arguments()	process command line
func_demo(args)	show what **punx** can do
func_validate(args)	validate the content of a NeXus HDF5 data file of NXDL XML file
func_hierarchy(args)	not implemented yet
func_configuration(args)	show internal configuration of punx
func_tree(args)	print the tree structure of a NeXus HDF5 data file of NXDL XML file
func_update(args)	update or install versions of the NeXus definitions

class punx.main.**MyArgumentParser**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *version=None*, *parents=[]*, *formatter_class=<class 'argparse.HelpFormatter'>*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=True*)

override standard ArgumentParser to enable shortcut feature

stretch goal: permit the first two char (or more) of each subcommand to be accepted # ?? http://stackoverflow.

com/questions/4114996/python-argparse-nargs-or-depending-on-prior-argument?rq=1

parse_args(*args=None*, *namespace=None*)
permit the first two char (or more) of each subcommand to be accepted

punx.main.**exit_message**(*msg*, *status=None*, *exit_code=1*)
exit this code with a message and a status

Parameters

- **msg** (*str*) – text to be reported
- **status** (*int*) – 0 - 50 (default: ERROR = 40)
- **exit_code** (*int*) – 0: no error, 1: error (default)

punx.main.**func_configuration**(*args*)
show internal configuration of punx

punx.main.**func_demo**(*args*)
show what **punx** can do

Internally, runs these commands:

```
punx validate <source_directory>/data/writer_1_3.hdf5
punx tree <source_directory>/data/writer_1_3.hdf5
```

If you get an error message that looks like this one (line breaks added here for clarity):

```
punx.cache.FileNotFound: file does not exist:
/Users/<username>/.config/punx/definitions-master/nxdl.xsd
AND not found in source cache either!  Report this problem to the developer.
```

then you will need to update your local cache of the NeXus definitions. Use this command to update the local cache:

```
punx update
```

punx.main.**func_hierarchy**(*args*)
not implemented yet

punx.main.**func_structure**(*args*)
deprecated subcommand

punx.main.**func_tree**(*args*)
print the tree structure of a NeXus HDF5 data file of NXDL XML file

punx.main.**func_update**(*args*)
update or install versions of the NeXus definitions

punx.main.**func_validate**(*args*)
validate the content of a NeXus HDF5 data file of NXDL XML file

punx.main.**parse_command_line_arguments**()
process command line

1.10 NXDL Manager : `nxdl_manager`

1.10.1 source code documentation

Load and/or document the structure of a NeXus NXDL class specification

The *nxdl_manager* calls the *schema_manager* and is called by *____tba_____*.

class punx.nxdl_manager.**NXDL_Manager**(*file_set=None*)
the NXDL classes found in `nxdl_dir`

get_nxdl_defaults()

class punx.nxdl_manager.**NXDL__Mixin**(*nxdl_definition*, **args*, ***kwds*)
base class for each NXDL structure

assign_defaults()
set default values for required components now

ensure_unique_name(*obj*)

parse_attributes(*xml_node*)

parse_fields(*xml_node*)

parse_groups(*xml_node*)

parse_links(*xml_node*)

parse_nxdl_xml(**args*, ***kwargs*)
parse the XML node and assemble NXDL structure

parse_symbols(*xml_node*)

parse_xml_attributes(*defaults*)

class punx.nxdl_manager.**NXDL__attribute**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *attribute* structure (XML element) in a NXDL XML file

~parse_nxdl_xml

parse_nxdl_xml(*xml_node*)
parse the XML content

class punx.nxdl_manager.**NXDL__definition**(*nxdl_manager=None*, **args*, ***kwds*)
contents of a *definition* element in a NXDL XML file

Parameters **path** (*str*) – absolute path to NXDL definitions directory (has nxdl.xsd)

parse_nxdl_xml()
parse the XML content

set_file(*fname*)
self.category: base_classes | applications | contributed_definitions

determine the category of this NXDL

class punx.nxdl_manager.**NXDL__dim**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *dim* structure (XML element) in a NXDL XML file

parse_nxdl_xml(*xml_node*)
parse the XML content

class punx.nxdl_manager.**NXDL__dimensions**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *dimensions* structure (XML element) in a NXDL XML file

parse_nxdl_xml(*xml_node*)
parse the XML content

class punx.nxdl_manager.**NXDL__field**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *field* structure (XML element) in a NXDL XML file

parse_nxdl_xml(*xml_node*)
parse the XML content

class punx.nxdl_manager.**NXDL__group**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *group* structure (XML element) in a NXDL XML file

parse_nxdl_xml(*xml_node*)
parse the XML content

class punx.nxdl_manager.**NXDL__link**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *link* structure (XML element) in a NXDL XML file

example from NXmonopd:

```
<link name="polar_angle" target="/NXentry/NXinstrument/NXdetector/polar_angle">
    <doc>Link to polar angle in /NXentry/NXinstrument/NXdetector</doc>
</link>
<link name="data" target="/NXentry/NXinstrument/NXdetector/data">
    <doc>Link to data in /NXentry/NXinstrument/NXdetector</doc>
</link>
```

parse_nxdl_xml(*xml_node*)
parse the XML content

class punx.nxdl_manager.**NXDL__symbols**(*nxdl_definition*, *nxdl_defaults=None*, **args*, ***kwds*)
contents of a *symbols* structure (XML element) in a NXDL XML file

example from NXcrystal:

```
<symbols>
  <doc>These symbols will be used below to coordinate dimensions with the same␣
↪lengths.</doc>
  <symbol name="n_comp"><doc>number of different unit cells to be described</doc>
↪</symbol>
  <symbol name="i"><doc>number of wavelengths</doc></symbol>
</symbols>
```

parse_nxdl_xml(*symbols_node*)
parse the XML content

punx.nxdl_manager.**get_NXDL_file_list**(*nxdl_dir*)
return a list of all NXDL files in the `nxdl_dir`

The list is sorted by NXDL category (base_classes, applications, contributed_definitions) and then alphabetically within each category.

punx.nxdl_manager.**validate_xml_tree**(*xml_tree*)
validate an NXDL XML file against the NeXus NXDL XML Schema file

Parameters **xml_file_name** (*str*) – name of XML file

1.11 NXDL Rules: The XML Schema files : `nxdl_schema`

Read the NeXus XML Schema

1.11.1 source code documentation

Read the NeXus XML Schema

NXDL_Summary(nxdl_xsd_file_name)	provide an easy interface for the nxdl_manager
render_class_str(obj)	useful optimization for classes
get_reference_keys(xml_node)	reference an xml_node in the catalog: `catalog[section][line]`
get_named_parent_node(xml_node)	return closest XML ancestor node with a `name` attribute or the schema node
get_xml_namespace_dictionary()	return the NeXus XML namespace dictionary

The `NXDL_item_catalog.definition_element` will provide the defaults for the definition, group, field, link, and symbols NXDL structures. These internal structures are used:

NXDL_item_catalog(nxdl_file_name)	content from the NeXus XML Schema (`nxdl.xsd`)
NXDL_schema__attribute()	node matches XPath query: `//xs:attribute`
NXDL_schema__attributeGroup()	node matches XPath query: `/xs:schema/xs:attributeGroup`
NXDL_schema__complexType()	node matches XPath query: `/xs:schema/xs:complexType`
NXDL_schema__element()	a complete description of a specific NXDL xs:element node
NXDL_schema__group()	node matches XPath query: `//xs:group`
NXDL_schema_named_simpleType()	node matches XPath query: `/xs:schema/xs:simpleType`

Note there is a recursion within *NXDL_schema__group* since a *group* may contain a child *group*.

class punx.nxdl_schema.**NXDL_Summary**(*nxdl_xsd_file_name*)

provide an easy interface for the nxdl_manager

USAGE:

```
summary = NXDL_Summary(nxdl_xsd_file_name)
...
summary.simpleType['validItemName'].patterns
```

class punx.nxdl_schema.**NXDL_item_catalog**(*nxdl_file_name*)

content from the NeXus XML Schema (`nxdl.xsd`)

EXAMPLE:

nxdl_xsd_file_name = os.path.join('cache', 'v3.2','nxdl.xsd') catalog = NXDL_item_catalog(nxdl_xsd_file_name) definition = catalog.definition_element

add_to_catalog(*node*, *obj*, *key=None*)

class punx.nxdl_schema.**NXDL_schema__Mixin**

class punx.nxdl_schema.**NXDL_schema__attribute**

node matches XPath query: `//xs:attribute`

xml_node is `xs:attribute`

a complete description of a specific NXDL attribute element

NOTES ON ATTRIBUTES

In nxdl.xsd, "attributeType" is used by fieldType and groupGroup to define the NXDL "attribute" element used in fields and groups, respectively. It is not necessary for this code to parse "attributeType" from the rules.

Each of these XML *complexType* elements defines its own set of attributes and defaults for use in corresponding NXDL components:

- attributeType
- basicComponent
- definitionType
- enumerationType
- fieldType
- groupType
- linkType

There is also an "xs:attributeGroup" which may appear as a sibling to any `xs:attribute` element. The `xs:attributeGroup` provides a list of additional `xs:attribute` elements to add to the list. This is the only one known at this time (2017-01-08):

- `deprecatedAttributeGroup`

When the content under `xs:complexType` is described within an `xs:complexContent/xs:extension` element, the `xs:extension` element has a `base` attribute which names a `xs:complexType` element to use as a starting point (like a superclass) for the additional content described within the `xs:extension` element.

The content may be found at any of these nodes under the parent XML element. Parse them in the order shown:

- `xs:complexContent/xs:extension/xs:attribute`
- `xs:attribute`
- (`xs:attributeGroup/`)''xs:attribute''

This will get picked up when parsing the `xs:sequence/xs:element`.

- `xs:sequence/xs:element/xs:complexType/xs:attribute` (

The XPath query for `//xs:attribute` from the root node will pick up all of these. It will be necessary to walk through the parent nodes to determine where each should be applied.

parse(*xml_node*)

read the attribute node content from the XML Schema

xml_node is xs:attribute

class punx.nxdl_schema.**NXDL_schema__attributeGroup**

node matches XPath query: `/xs:schema/xs:attributeGroup`

xml_node is `xs:attributeGroup`

parse(*xml_node*)
read the attributeGroup node content from the XML Schema

xml_node is xs:attributeGroup

class punx.nxdl_schema.**NXDL_schema__complexType**
node matches XPath query: `/xs:schema/xs:complexType`

xml_node is `xs:complexType`

parse(*xml_node*, *catalog*)
read the element node content from the XML Schema

class punx.nxdl_schema.**NXDL_schema__element**
a complete description of a specific NXDL xs:element node

parse(*xml_node*)
read the element node content from the XML Schema

class punx.nxdl_schema.**NXDL_schema__group**
node matches XPath query: `//xs:group`

xml_node is `xs:group`

parse(*xml_node*)
read the element node content from the XML Schema

class punx.nxdl_schema.**NXDL_schema_named_simpleType**
node matches XPath query: `/xs:schema/xs:simpleType`

xml_node is `xs:simpleType`

parse(*xml_node*)
read the attribute node content from the XML Schema

punx.nxdl_schema.**get_named_parent_node**(*xml_node*)
return closest XML ancestor node with a `name` attribute or the schema node

punx.nxdl_schema.**get_reference_keys**(*xml_node*)
reference an xml_node in the catalog: `catalog[section][line]`

punx.nxdl_schema.**get_xml_namespace_dictionary**()
return the NeXus XML namespace dictionary

punx.nxdl_schema.**print_tree**(*obj*, *level=0*)

punx.nxdl_schema.**render_class_str**(*obj*)
useful optimization for classes

USAGE:

```
def __str__(self):
    return render_class_str(self)
```

1.12 NXDL Definition File Tree Structure : `nxdltree`

Describe the tree structure of a NeXus Definition Language NXDL XML file.

Note: The *tree* subcommand replaces the now-legacy *structure* subcommand.

1.12.1 source code documentation

Describe the tree structure of a NXDL XML file

NxdlTreeView(nxdl_file)	Describe the tree structure of a NXDL XML file

class punx.nxdltree.**NxdlTreeView**(*nxdl_file*)
Describe the tree structure of a NXDL XML file

Example usage showing default display:

```
mc = NxdlTreeView(nxdl_file_name)
mc.array_items_shown = 5
show_attributes = False
txt = mc.report(show_attributes)
```

report(*show_attributes=True*)
return the structure of the NXDL file in a list of strings

The work of parsing the data file is done in this method.

punx.nxdltree.**xslt_transformation**(*xslt_file*, *src_xml_file*)
return the transform of an XML file using an XSLT

Parameters

- **xslt_file** (*str*) – name of XSLT file
- **src_xml_file** (*str*) – name of XML file

1.13 Manage the XML Schema files : schema_manager

-tba-

1.13.1 source code documentation

manages the XML Schema of this project

The *schema_manager* calls the *cache_manager* and is called by *nxdl_manager*.

Public

SchemaManager([path])	describes the XML Schema for the NeXus NXDL definitions files
Schema_Root(element_node[, obj_name, ...])	root element of the nxdl.xsd file
Schema_Attribute(xml_obj[, obj_name, ...])	xs:attribute element
Schema_Element(xml_obj[, obj_name, ns_dict, ...])	xs:element
Schema_Type(ref[, tag, schema_root])	a named NXDL structure type (such as groupGroup)
get_default_schema_manager()	internal: convenience function
raise_error(node, text, obj)	standard *ValueError* exception handling
strip_ns(ref)	strip the namespace prefix from ref

Internal

_Mixin(xml_obj[, obj_name, ns_dict, schema_root])	common code for NXDL Rules classes below
_GroupParsing	internal: avoid a known recursion of group in a group
_Recursion(obj_name)	internal: an element used in recursion, such as child group of group

class punx.schema_manager.**SchemaManager**(*path=None*)

describes the XML Schema for the NeXus NXDL definitions files

parse_nxdlTypes()

get the allowed data types and unit types from nxdlTypes.xsd

parse_nxdl_patterns()

get regexp patterns for validItemName, validNXClassName, & validTargetName from nxdl.xsd

class punx.schema_manager.**Schema_Attribute**(*xml_obj*, *obj_name=None*, *ns_dict=None*, *schema_root=None*)

xs:attribute element

Parameters

- **xml_obj** (*lxml.etree.Element*) – XML element
- **obj_name** (*str*) – optional, default taken from xml_obj
- **ns_dict** (*dict*) – optional, default taken from NAMESPACE_DICT
- **schema_root** (*obj*) – optional, instance of lxml.etree._Element

class punx.schema_manager.**Schema_Element**(*xml_obj*, *obj_name=None*, *ns_dict=None*, *schema_root=None*)

xs:element

Parameters

- **xml_obj** (*lxml.etree.Element*) – XML element
- **obj_name** (*str*) – optional, default taken from xml_obj
- **ns_dict** (*dict*) – optional, default taken from NAMESPACE_DICT
- **schema_root** (*obj*) – optional, instance of lxml.etree._Element

See http://download.nexusformat.org/doc/html/nxdl.html

See http://download.nexusformat.org/doc/html/nxdl_desc.html#nxdl-elements

class punx.schema_manager.**Schema_Root**(*element_node*, *obj_name=None*, *ns_dict=None*, *schema_root=None*, *schema_manager=None*)

root element of the nxdl.xsd file

Parameters

- **xml_obj** (*lxml.etree.Element*) – XML element
- **obj_name** (*str*) – optional, default taken from xml_obj
- **ns_dict** (*dict*) – optional, default taken from NAMESPACE_DICT
- **schema_root** (*obj*) – optional, instance of lxml.etree._Element

parse_sequence(*seq_node*)

parse the sequence used in the root element

class punx.schema_manager.**Schema_Type**(*ref*, *tag='*'*, *schema_root=None*)
a named NXDL structure type (such as groupGroup)

Parameters

- **ref** (*str*) – name of NXDL structure type (such as groupGroup)
- **tag** (*str*) – XML Schema element tag, such as complexType (default=‘‘*‘‘)
- **schema_root** (*obj*) – optional, instance of lxml.etree._Element

See http://download.nexusformat.org/doc/html/nxdl.html

See http://download.nexusformat.org/doc/html/nxdl_desc.html#nxdl-data-types-internal

parse_sequence(*node*)

class punx.schema_manager.**Schema_nxdlType**(*xml_obj*, *ns_dict=None*, *schema_root=None*)
one of the types defined in the file *nxdlTypes.xsd*

class punx.schema_manager.**Schema_pattern**
describe the regular expression patterns ofr names of NeXus things

punx.schema_manager.**get_default_schema_manager**()
internal: convenience function

punx.schema_manager.**raise_error**(*node*, *text*, *obj*)
standard *ValueError* exception handling

Parameters

- **node** (*obj*) – instance of
- **text** (*str*) – label for obj
- **obj** (*str*) – value

punx.schema_manager.**strip_ns**(*ref*)
strip the namespace prefix from ref

Parameters **ref** (*str*) – one word, colon delimited string, such as *nx:groupGroup*

Returns str the part to the right of the last colon

1.14 Validation : validate

The process of validation compares each item in an HDF5 data file and compares it with the NeXus standards to check that the item is valid within that standard. Each test is assigned a *finding* result, a Severity object, with values and meanings as shown in the table below.

value	color	meaning
OK	green	meets NeXus specification
NOTE	palegreen	does not meet NeXus specification, but acceptable
WARN	yellow	does not meet NeXus specification, not generally acceptable
ERROR	red	violates NeXus specification
TODO	blue	validation not implemented yet
UNUSED	grey	optional NeXus item not used in data file
COMMENT	grey	comment from the *punx* source code

Items marked with the WARN *severity* status are as noted in either the NeXus manual[1], the NXDL language specification[2], or the NeXus Definition Language (NXDL) files[3].

The *color* is a suggestion for use in a GUI.

Numerical values are associated with each finding value. The sum of these values is averaged to produce a numerical indication of the validation of the file against the NeXus standard. An average of 100 indicates that the file meets the NeXus specification for every validation test applied. An average that is less than zero indicates that the file contains content that is not valid with the NeXus standard.

1.14.1 NeXus HDF5 Data Files

NeXus data files are HDF5[4] and are validated against the suite of NXDL files using tools provided by this package. The strategy is to compare the structure of the HDF file with the structure of the NXDL file(s) as specified by the `NX_class` attributes of the various HDF groups in the data file.

1.14.2 NeXus NXDL Definition Language Files

NXDL files are XML and are validated against the XML Schema file: `nxdl.xsd`. See the GitHub repository[5] for this file.

---

source code documentation

validate files against the NeXus/HDF5 standard

PUBLIC

Data_File_Validator([ref])	manage the validation of a NeXus HDF5 data file

INTERNAL

ValidationItem(parent, obj[, attribute_name])	HDF5 data file object for validation

class punx.validate.**Data_File_Validator**(*ref=None*)

manage the validation of a NeXus HDF5 data file

USAGE

1. make a validator with a certain schema:

```
validator = punx.validate.Data_File_Validator()     # default
```

You may have downloaded additional NeXus Schema (NXDL file sets). If so, pick any of these by name as follows:

[1] NeXus manual: http://download.nexusformat.org/doc/html/user_manual.html
[2] NXDL Language: http://download.nexusformat.org/doc/html/nxdl.html
[3] NeXus Class Definitions (NXDL files): http://download.nexusformat.org/doc/html/classes/index.html
[4] HDF5: https://support.hdfgroup.org/HDF5/
[5] NeXus GitHub Definitions repository: https://github.com/nexusformat/definitions

```
validator = punx.validate.Data_File_Validator("v3.2")
validator = punx.validate.Data_File_Validator("master")
```

2. use to validate a file or files:

```
result = validator.validate(hdf5_file_name)
result = validator.validate(another_file)
```

3. close the HDF5 file when done with validation:

```
validator.close()
```

PUBLIC METHODS

close()	close the HDF5 file (if it is open)
validate(fname)	start the validation process from the file root
print_report()	print a validation report

INTERNAL METHODS

build_address_catalog()	find all HDF5 addresses and NeXus class paths in the data file
_group_address_catalog_(parent, group)	catalog this group's address and all its contents
validate_item_name(v_item[, key])	

build_address_catalog()
find all HDF5 addresses and NeXus class paths in the data file

close()
close the HDF5 file (if it is open)

finding_score()
return a numerical score for the set of findings

count: number of findings total: sum of status values for all findings score: total / count – average status / finding

finding_summary(*report_statuses=None*)
return a summary dictionary of the count of findings by status

summary statistics ======= ===== == status count description ======= ===== == OK 10 meets NeXus specification NOTE 1 does not meet NeXus specification, but acceptable WARN 0 does not meet NeXus specification, not generally acceptable ERROR 0 violates NeXus specification TODO 3 validation not implemented yet UNUSED 2 optional NeXus item not used in data file COMMENT 0 comment from the punx source code – – – TOTAL 16 – ======= ===== ==

print_report()
print a validation report

record_finding(*v_item*, *key*, *status*, *comment*)
prepare the finding object and record it

usedAsBaseClass(*nx_class*)
returns bool: is the nx_class a base class?

NXDL specifications in the contributed definitions directory could be intended as either a base class or an application definition. NeXus provides no easy identifier for this difference. The most obvious distinction between them is the presence of the *definition* field in the *NXentry* group of an application definition. This field is not present in base classes.

validate(*fname*)
start the validation process from the file root

validate_application_definition(*v_item*)
validate group as a NeXus application definition

validate_group(*v_item*)
validate the NeXus content of a HDF5 data file group

class punx.validate.**ValidationItem**(*parent*, *obj*, *attribute_name=None*)
HDF5 data file object for validation

determine_NeXus_classpath()
determine the NeXus class path

See http://download.nexusformat.org/sphinx/preface.html#class-path-specification

EXAMPLE

Given this NeXus data file structure:

```
/
    entry: NXentry
        data: NXdata
            @signal = data
            data: NX_NUMBER
```

For the "signal" attribute of this HDF5 address: `/entry/data`, its NeXus class path is: `/NXentry/NXdata@signal`

The `@signal` attribute has the value of `data` which means that the local field named `data` is the plottable data.

The HDF5 address of the plottable data is: `/entry/data/data`, its NeXus class path is: `/NXentry/NXdata/data`

1.15 Source Code

main	Python Utilities for NeXus HDF5 files
validate	validate files against the NeXus/HDF5 standard
h5tree	Describe the tree structure of any HDF5 file
nxdltree	Describe the tree structure of a NXDL XML file
nxdl_manager	Load and/or document the structure of a NeXus NXDL class specification
nxdl_schema	Read the NeXus XML Schema
schema_manager	manages the XML Schema of this project
cache_manager	manages the NXDL cache directories of this project
github_handler	manages the communications with GitHub

1.16 Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

R

S

T

U

V

W

X