# punx Documentation

***Release 214.g23e40e2.dirty***

**Pete R. Jemian**

**Oct 09, 2017**

# 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 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** 0.1.9

**release** 214.g23e40e2.dirty

**published** Oct 09, 2017

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
punx -h
usage: punx [-h] [-v] {demo,hierarchy,structure,update,validate} ...

Python Utilities for NeXus HDF5 files URL: http://punx.readthedocs.io
v0+unknown

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

subcommands:
  valid subcommands

  {demonstrate,structure,update,validate}
    demonstrate         demonstrate HDF5 file validation
    structure           show structure of HDF5 or NXDL file
    update              update the local cache of NeXus definitions
    validate            validate a NeXus file

http://punx.readthedocs.io
```

## 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.

```
console> punx validate C:\Users\shoun\Apps\Anaconda\lib\site-
↪packages\punx\data\writer_1_3.hdf5
Validation findings
:file: writer_1_3.hdf5
:validation results shown:  COMMENT, ERROR, NOTE, OK, TODO, UNUSED, WARN
=================== ============================ ======␣
↪==================================================
address              validation                   status comment(s)
=================== ============================ ======␣
↪==================================================
/                    @NX_class assumed            OK     file root: NXroot
/Scan                validItemName                NOTE   relaxed re: [A-Za-z_][\w_]*
/Scan@NX_class       @NX_class known              OK     NXentry is known
/Scan/data           validItemName-strict         OK     strict re: [a-z_][a-z0-9_]*
/Scan/data@NX_class  @NX_class known              OK     NXdata is known
/Scan/data/counts    NXdata@ignoreExtraFields     OK     field ignored due to group␣
↪attribute setting
/Scan/data/two_theta NXdata@ignoreExtraFields     OK     field ignored due to group␣
↪attribute setting
/Scan/data           NXDL review: NXdata          TODO   validate with NXdata␣
↪specification (incomplete)
/Scan                NXDL review: NXentry         TODO   validate with NXentry␣
↪specification (incomplete)
/                    NXDL review: NXroot          TODO   validate with NXroot␣
↪specification (incomplete)
```

```
/Scan/data/counts    NXdata group default plot v3 OK     NXdata@signal = counts
/Scan/data/counts    NeXus default plot v3        OK     NeXus data file default plot: /NXentry/NXdata@signal
==================== ============================ ======================================================

summary statistics
======= ===== ============================================================
status  count description
======= ===== ============================================================
OK      8     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  0     optional NeXus item not used in data file
COMMENT 0     comment from the punx source code
--      --    --
TOTAL   12    --
======= ===== ============================================================

console> punx structure C:\Users\shoun\Apps\Anaconda\lib\site-packages\punx\data\writer_1_3.hdf5
C:\Users\shoun\Apps\Anaconda\lib\site-packages\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] = [17.926079999999999, 17.925909999999998, 17.925750000000001, '...', 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: structure

show structure of HDF5 or NXDL file

### command line help

```
console> punx structure -h
punx structure -h
usage: punx structure [-h] [-a] 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
```

### 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

  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. 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*).

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?
5. Does the file define a path to the default plot?
6. Is the file a NeXus HDF5 data file?

### Is file HDF5?

This is a simple test and is handled by a library routine *h5py*.

### 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.

TODO: groups must be either one of the defined base classes (or contributed definitions but this is rare)

[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 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 group 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*, then 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.

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 in incubation 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 |
|---|---|
| *demonstrate* | demonstrate HDF5 file validation |
| *hierarchy* | show NeXus base class hierarch |
| *structure* | show 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:

```
console> punx validate -h
punx validate -h
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=20 (INFO)
```

**Tip:** Subcommands may be abbreviated.

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`.

## 1.2 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
```

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

| 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 only used for various reports.** If using the 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

**0.1.9** 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

```
    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).
        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.
        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.

    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 |

Continued on next page

Table 1.2 – continued from previous page

| | |
|---|---|
| *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
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
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 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.

### 1.6.1 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'v3.3'*)
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.

## 1.7 NXDL Manager : `nxdl_manager`

### 1.7.1 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

## 1.8 Manage the XML Schema files : schema_manager

-tba-

### 1.8.1 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

## 1.9 Source Code

| | |
|---|---|
| *nxdl_manager* | Load and/or document the structure of a NeXus NXDL class specification |
| *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 |

## 1.10 Indices and tables

- genindex
- modindex
- search

# Python Module Index

## p

# Index

## P



## R



## S



## T



## U



## V



## W