Create NeXus files by python and validate them

1. The goal

1. Use python to create a NeXus file (.nxs) by hardcoding via the python package h5py
2. Use a second tool to validate this hardcoded file to a given set of NeXus definitions:
    1. FAIRmat https://fairmat-nfdi.github.io/nexus_definitions/index.html#
    2. NIAC https://manual.nexusformat.org/

2. Create NeXus by hardcoding with python

Install h5py via pip by `pip install h5py`

Then you can create a python by the python script called "h5py_nexus_file_creation.py".

```
# Import h5py, to write an hdf5 file
import h5py

# create a h5py file in writing mode with given name
# "NXopt_minimal_example", file extension "nxs"
f = h5py.File("NXopt_minimal_example.nxs", "w")

# there are only 3 fundamental objects: >group<, >attribute<
# and >datafield<.


# create a >group< called "entry"
f.create_group('/entry')

# assign the >group< called "entry" an >attribute<
# The attribute is "NX_class"(a NeXus class) with the value of
# this class is "NXentry"
f['/entry'].attrs['NX_class'] = 'NXentry'

# create >datafield< called "definition" inside the entry, and assign it
# the value "NXoptical_spectroscopy"
# This field is important, as it is used in validation process to
# identify the NeXus definition.
f['/entry/definition'] = 'NXoptical_spectroscopy'
```

This proves a starting point of the NeXus file. The comments indicated by "#" help to understand what these lines do. Well go through these functions in the following.

2.1. Fill the content of the .nxs file

Go to https://fairmat-nfdi.github.io/nexus_definitions/index.html#

Scroll down until you see the search box named "Quick search".

Type "NXoptical" and press start the search.

You see several search results, select the one with is named "NXoptical_spectroscopy".

Then you are (ideally) on this page: https://fairmat-nfdi.github.io/nexus_definitions/classes/contributed_definitions/NXoptical_spectroscopy.html

You see a tree like structure of the NeXus definition NXoptical_spectrosocopy with several tree nodes: Status, Description, Symbols, Groups_cited, Structure. For now, only the part in Structure is of interest. This contains the information, which has to be written in the python code to extend the NeXus file, which is created by the script "NXopt_minimal_example.nxs".

Use your browser search (CRTL+F) and search for "required". Ideally your browser highlights all entries which are required. You have to add those to the python script, to extend your created .nxs file.

In the following. It will be shown, how the python script has to be extendend for the three fundamental objects:

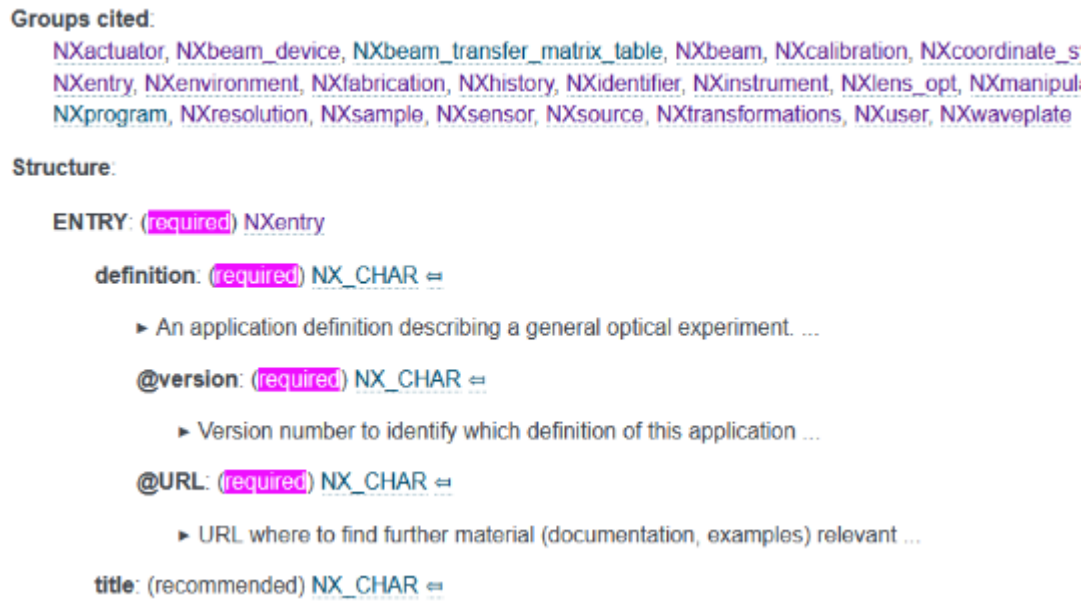
1. Attribute
2. Datafield
3. Group

2.2. Adding an attribute

Search for the first entry in the NeXus file, which is not created yet. For me it is:

@version: (required) NX_CHAR ⇦

1. It is located in the Tree at position: ENTRY/definition/
2. The "@" indicates, that this is an attribute of the concept "definition".
3. The name of the attribute is "version".
4. The "required" indicates, that this attribute has to be added to be in line with the NeXus definition "NXoptical_spectroscopy".
5. The "NX_CHAR" indicates the datatype. This is should be a string: "The preferred string representation is UTF-8" (more information see here: https://manual.nexusformat.org/nxdl-types.html)



Now the python script has to be extended in the following:

```
f['/entry/definition'].attrs['version'] = 'v2024.02'
```

This h5py command extends the entry "/entry/definition" by the attribute named "version" with the value "v2024.02". The same is done for the URL attribute:

```
f['/entry/definition'].attrs['URL'] = 'https://github.com/FAIRmat-
NFDI/nexus_definitions/blob/f75a29836431f35d68df6174e3868a0418523397/contributed_d
efinitions/NXoptical_spectroscopy.nxdl.xml'
```

Though, you have to use your versions which you want to refer to, as in a few years this NeXus definition might change a little bit. This is shown in the following.

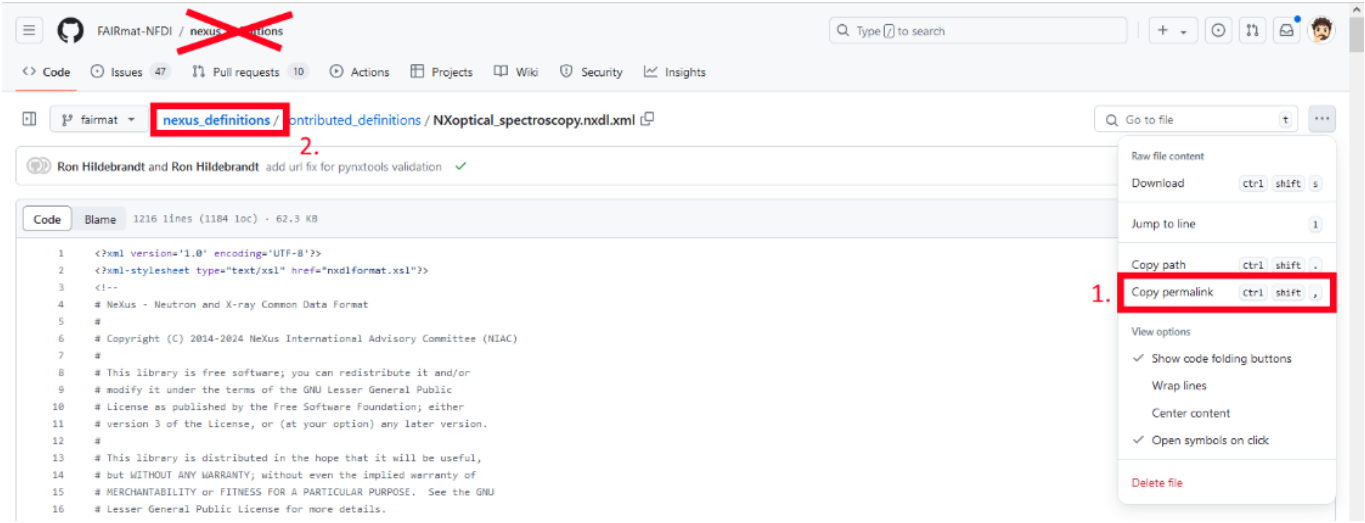
How to get the "version" and "URL" values

At the time, you create the NeXus definition. Go to the page of the respectively used NeXus concept, i.e. https://fairmat-nfdi.github.io/nexus_definitions/classes/contributed_definitions/NXoptical_spectroscopy.html

Scroll down until you find "**NXDL Source**:" and follow this link, i.e. https://github.com/FAIRmat-NFDI/nexus_definitions/blob/fairmat/contributed_definitions/NXoptical_spectroscopy.nxdl.xml

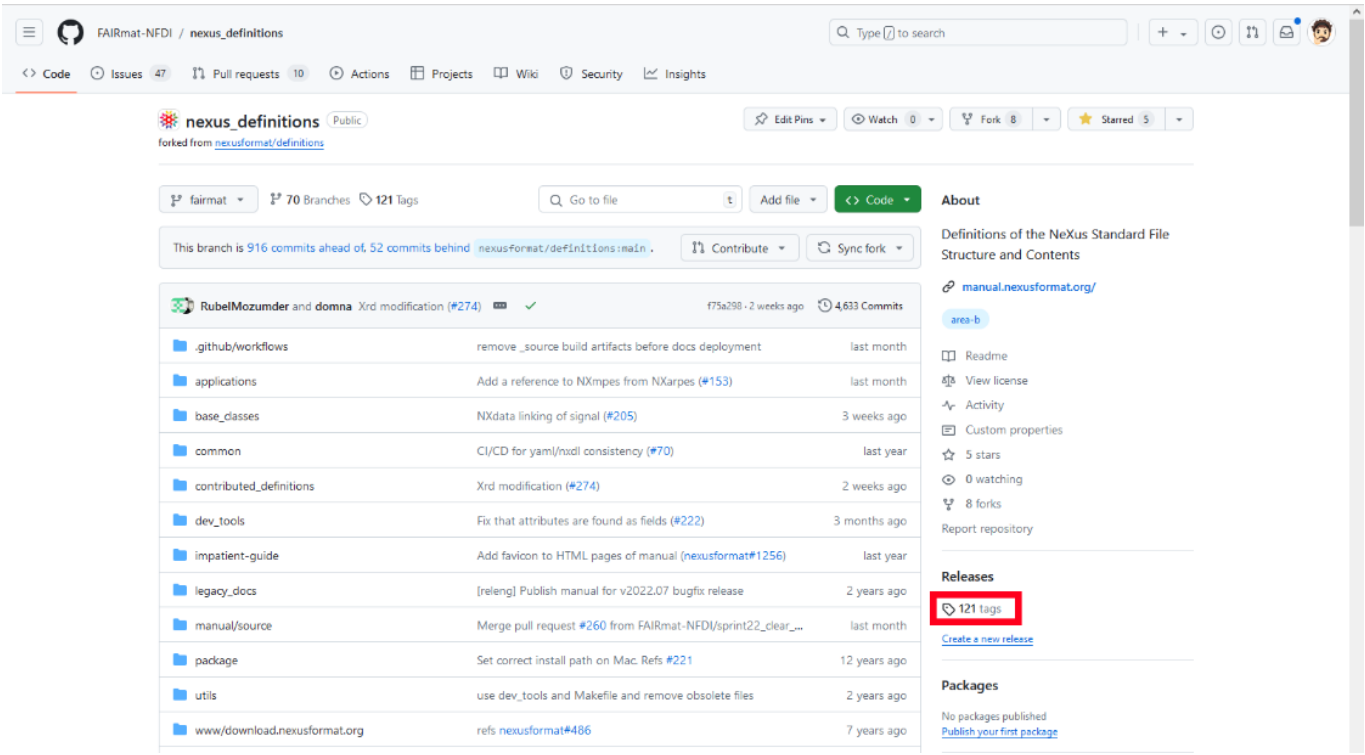
This is the github website, in which the latest NeXus definition of NXoptical_spectroscopy is stored in the NeXus definition language file (.nxdl). The information is structured in the xml format.

Now you have to copy the permalink of this file. Go to the top right side of the website. Find the Menu made by 3 dots:



Copy the permalink and insert it as value for the "URL" attribute (Step 1, Red box in the image)

Go to "nexus_definitions" (Step 2, Red box in the image)



On the right side, you should see below "Releases" the "tags" (Red box in the image). Follow this link.

Copy the latest Tag, which should look similar to "v2024.02". Insert it as value for the "version" attribute.

2.3. Adding a datafield

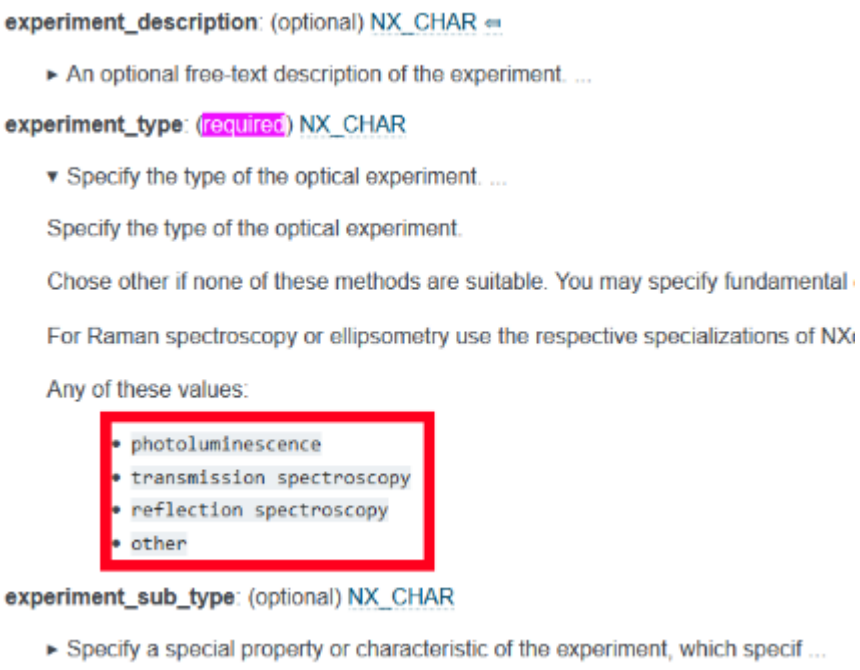
Two attributes were added two "ENTRY/definition". Both were required. By this, now this part of the NeXus file is in line with the NeXus Definition for NXoptical_spectroscopy.

The next required entry of this NeXus definition (use https://fairmat-nfdi.github.io/nexus_definitions/classes/contributed_definitions/NXoptical_spectroscopy.html with brwoser search for "required") is "**experiment_type**".

experiment_type: (required) NX_CHAR

1. It is located in the Tree at position: ENTRY/
2. There is no "@" in front of "**experiment_type"**. So, this may be a group or a datafield.
3. The name of this group/datafield is "**experiment_type**".
4. The "required" indicates, that this group/datafield has to be added to be in line with the NeXus definition "NXoptical_spectroscopy".
5. The "NX_CHAR" indicates the datatype. This is should be a string: "The preferred string representation is UTF-8" (more information see here: https://manual.nexusformat.org/nxdl-types.html).
6. The "NX_CHAR" indicates, that this is a datafield. It is NOT a group.
   A group would have a link to a NeXus class (i.e. for "**ENTRY**: (required) NXentry" to https://fairmat-nfdi.github.io/nexus_definitions/**classes**/base_classes/NXentry.html#nxentry).
   As it is a field, the link directs to a data type (i.e. https://fairmat-nfdi.github.io/nexus_definitions/**nxdl-types**.html#nx-char).

Read the documentation at "▶ Specify the type of the optical experiment. ..." by extending it via click on the triangle symbol. You should see something like this:



There value of the datafile has to be one of the list. e.g "transmission spectroscopy". This is case sensitive. Best is to just copy the string from the website and paste it into the python script.

Therefore, the python script has to be extended by:

```
f['/entry/experiment_type'] = 'transmission spectroscopy'
```

2.4. Adding a group

The first required group in NXoptical_spectroscopy on the "ENTRY/" level is "**INSTRUMENT**: (required) NXinstrument ⇦"

1. It is located in the Tree at position: ENTRY/
2. There is no "@" in front of "**INSTRUMENT"** and because the "NXinstrument" points to link of a NeXus class, this has to be implemented as group in the python script.
3. The "required" indicates, that this group has to be added to be in line with the NeXus definition "NXoptical_spectroscopy".
4. The "NXinstrument" indicates via the link, that it is a NeXus class (or group in python).
5. As this is a group, only attributes can be assigned to this. No value is assigned to the group.
6. As this is a group, it can contain many datafield or groups.
7. The uppercase notation of "**INSTRUMENT**" means:
    1. You can give INSTRUMENT any name, such as "abc" or "Raman_setup".
    2. You can create as many groups with the class NXinstrument as you want. Their names have to be different.
    3. For more information see: https://github.com/FAIRmat-NFDI/pynxtools/blob/master/docs/learn/nexus-rules.md

The respective python code to implement a NXinstrument class (or equivalently in python group) with the name "experiment_setup_1" is:

```
f.create_group('/entry/experiment_setup_1')
f['/entry/experiment_setup_1'].attrs['NX_class'] = 'NXinstrument'
```

The first line creates the group with the name "experiment_setup_1".

The second line, assigns this group the attribute with the name "NX_class" and it's value "NXinstrument".

2.5. Finishing the .nxs file

This has to be done by using the respective NeXus definiton website:

https://fairmat-nfdi.github.io/nexus_definitions/classes/contributed_definitions/NXoptical_spectroscopy.html

And by searching for all "required" entries. The next required entries are located inside the NXinstrument class:

1. **beam_TYPE**: (required) NXbeam ⇦
2. **detector_TYPE**: (required) NXdetector ⇦

Both are groups. "**beam_TYPE"** could be named: "beam_abc" or "beam_Raman_setup". Use the knowledge above to extend the python script to create those NeXus file entries.

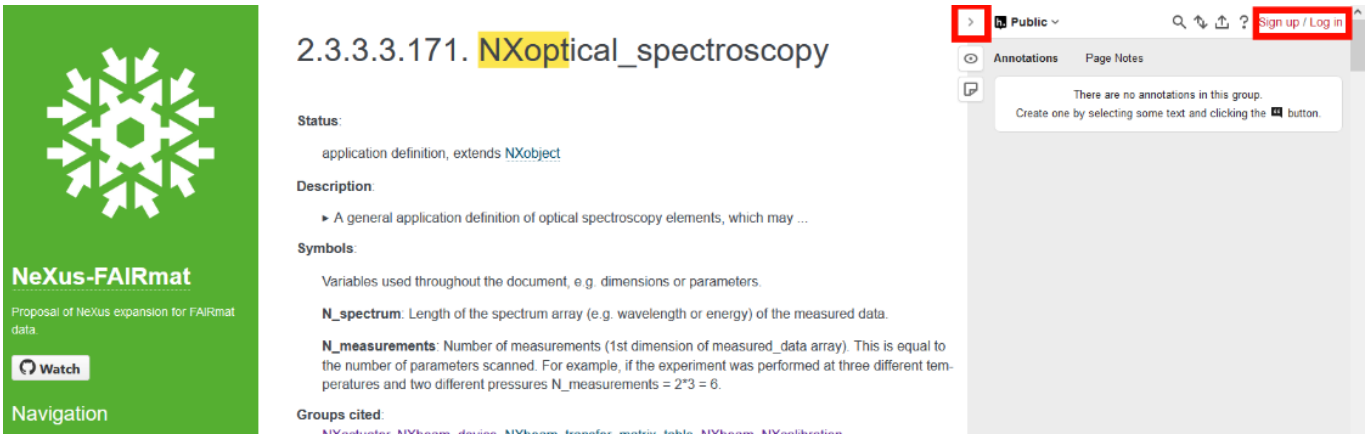
Note for required entries:

Above in the definition of NXoptical_spectroscopy, you as well may found a required entry "**depends_on**: (required) NX_CHAR ⇦". This is at the level of "ENTRY/reference_frames/beam_ref_frame". If you dont have the group "**beam_ref_frame"** because this is "optional", then you don't need to have this field.

3. Validation of a .nxs file

The validity of NeXus file is fundamental, to ensure FAIR data. Without specific requirements, it is not possible to understand the data. What type of experiment? What Laser Wavelength? Which voltage? What data is represented at all in the table? What is the unit of the value? Which ISO norm does this refer to? Where was this measured? Which year was this measured?

Therefore you have enter all required fields in the NeXus definition. The requirements are set by the community via Workshops, or at Conferences. You can as well comment the NeXus definitions, to initiate or propose changes/additions. Go to the NeXus definition, and sign-up/log-in and give us some feedback (Red boxes in the image. Expand this panel on the left by clicking on the arrow symbol).



Though, humans make errors: Typos, missing requirements, forget to add attributes, using the incorrect datatype or format (Matrix instead of List, Float instead of integer, etc.). Therefore a validation is required, to ensure, that you can share finally your FAIR data.

This is done by software.

3.1. Validation software

There are right now three methods, which can be used for validation of NeXus files. All are different and have individual advantages or disadvantages:

1. cnxvalidate
2. punx
3. pynxtools

Open software is usually shared on Github - There you find usually the most accurate information, as documentation sometimes lags behind. There you see a box with folders and files. Below is the content of the README.md file displayed. This usually shows instructions for installation and handling of the software.

Here are the GitHub links for the thee software packages:

cnxvalidate: https://github.com/nexusformat/cnxvalidate

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

pynxtools: https://github.com/FAIRmat-NFDI/pynxtools

In the following, each package and its capabilities is presented.

Operating systems

Almost all PC users are used to Windows as operating system.

A lot of Software development is done on Linux as operating system.

This is not a problem for big company, but for smaller open software, which are often develop without any payment, this is a problem.

If you are used to Windows, consider setting up a Linux operating system to eliminate problems in the installation process and ensure compatibility.

3.2. cnxvalidate

This package is written in c. It is allows a command line evocation like:

```
nxvalidate -l appdefdir datafile
```

1. nxvalidate: calls the software
2. -l appdefdir: points to the location of the NeXus definitions you want to use. This is a path to a folder called "defintions".
3. datafile: This is the path to the .nxs file which should be checked.

This output shows warnings like:

```
definition=NXoptical_spectroscopy.nxdl.xml message="Required attribute URL
missing" nxdlPath=/NXentry/definition sev=error dataPath=/entry/definition
dataFile=NXopt_minimal_example.nxs
```

and indicates the entry of the .nxs file, which is incorrect and what the respective problem is. It also points to the NeXus definition (.nxdl.xml file), in which this conflict was found.

While the software itself is powerful, its installation is difficult.

3.2.1. Installation (Linux only)

This did not work for me on windows. (The problem was the software cmake couldn't fine the libxml2 library. Though, if you solve this, this maybe work on windows).

Therefore: Use linux.

The installation process has to be build from source. This is eased significantly by using another software called: cmake.

Install cmake, github, hdf5 & xml2 library, etc:

open the terminal and install all parts required to install cnxvalidate via cmake:

```
sudo apt-get update
sudo apt-get install git
sudo apt-get install build-essential
sudo add-apt-repository universe
sudo apt-get install libhdf5-serial-dev
sudo apt-get -y install pkg-config
sudo apt upgrade -y
sudo apt-get -y install cmake
sudo apt-get install libxml2-dev
```

Directoty location

create a folder named "nexusvalidation" via terminal or file manager.

The folder is located at `/home/USER/nexusvalidation`

"USER" is your user name. (You can get your username by the terminal command: `echo $USER`)

In the terminal, this is indicated by `~/nexusvalidation` (~ = /home/USER)

open the thermal and go into this directory by:

```
cd /home/USER/nexusvalidation
```

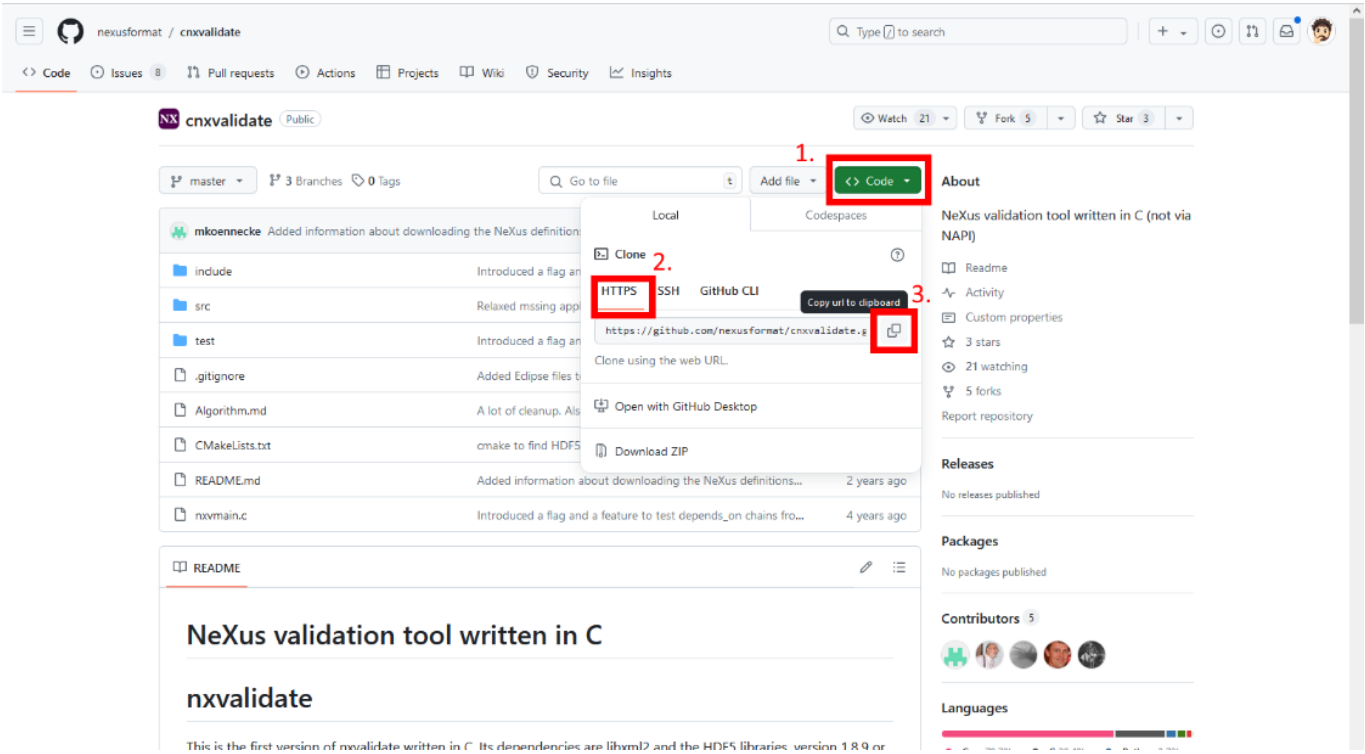
Using GitHub

Go to the Github Repository of cnxvalidate: https://github.com/nexusformat/cnxvalidate

Click on the green "<> Code" button.

Click on "HTTPS".

Copy the https link.



open the terminal and ensure you are in the `nexusvalidation` folder.

clone the github repository (download the files of the software).

```
git clone https://github.com/nexusformat/cnxvalidate.git
```

now you have a new folder at `~/nexusvalidation/cnxvalidate`

go into this folder via the command

```
cd cnxvalidate
```

now you are in the source tree. This should be exactly the same files, which you find on the github repository (https://github.com/nexusformat/cnxvalidate)

make a new directory called "build":

```
mkdir build
```

go into this directory

```
cd build
```

use cmake, to compile/build the software - this puts together all pieces of software - and especially external parts such as xml2 and hdf5 library.

```
cmake ../
```

install cnxvalidate after it was sucessfully build

```
make
```

Now the above mentioned commands should be avaialble. The programm/executable is located at:

```
/home/USER/nexusvalidation/cnxvalidate/build/nxvalidate
```

3.2.2. Using cnxvalidate

Now you can start to validate your created NeXus file. You may also just use one of the provided [MAKE LINK] files. But before the validation, we need to get a set of NeXus definitions, which we want to use as reference. This is done by git:

Getting NeXus definitions

go to the folder nexusvalidation

```
cd /home/USER/nexusvalidation
```

Download a set of NeXus definitions. Choose only one:

For FAIRmat NeXus definitions, go to https://github.com/FAIRmat-NFDI/nexus_definitions and copy the github "Code" line to clone the repository. Then:

```
git clone https://github.com/FAIRmat-NFDI/nexus_definitions.git
```

For the NIAC NeXus definitions, go to https://github.com/nexusformat/definitions and copy the github "Code" line to clone the repository. Then:

```
git clone https://github.com/nexusformat/definitions.git
```

Now you have a folder called "definitions" in the "nexusvalidation" folder. The path to this definitions folder is used as option for cnxvalidate, to tell the program, which NeXus definitions shall be used.

The respective path would be:

```
/home/USER/nexusvalidation/definitions
```

Get your NeXus file

put your NeXus file created above ("NXopt_minimal_example.nxs") into the "nexusvalidation" folder (filemanager or change the output location in the python script).

The file should now be loacted at

```
/home/USER/nexusvalidation/NXopt_minimal_example.nxs
```

Validating the NeXus file

now you can use the cnxvalidate with the executable called "nxvalidate" to use the set of NeXus definitions called "appdefdir" to validate the NeXus file called "datafile". This is done from the terminal.

```
nxvalidate -l appdefdir datafile
```

All names are "paths" to the definition, application or file. Use absolute paths, if you are not experienced, but relative paths work as well.

For the provided example, the suitable command looks like:

```
/home/USER/nexusvalidation/cnxvalidate/build/nxvalidate -l
/home/USER/nexusvalidation/definitions
/home/USER/nexusvalidation/NXopt_minimal_example.nxs
```

The "-l" option tells the program, that it should look for the nexus definiton at the path after "-l".

For the proved file above, the output should look like this:

```
USER@XXX:/home/USER/nexusvalidation/cnxvalidate/build/nxvalidate -l
/home/USER/nexusvalidation/definitions
/home/USER/nexusvalidation/NXopt_minimal_example.nxs
definition=NXoptical_spectroscopy.nxdl.xml message="Required attribute version
missing" nxdlPath=/NXentry/definition sev=error dataPath=/entry/definition
dataFile=NXopt_minimal_example.nxs
definition=NXoptical_spectroscopy.nxdl.xml message="Required attribute URL
```

```
missing" nxdlPath=/NXentry/definition sev=error dataPath=/entry/definition
dataFile=NXopt_minimal_example.nxs
definition=NXoptical_spectroscopy.nxdl.xml message="Required field missing"
nxdlPath=/NXentry/experiment_type sev=error dataPath=/entry/experiment_type
dataFile=NXopt_minimal_example.nxs
definition=NXoptical_spectroscopy.nxdl.xml message="Required group missing"
nxdlPath=/NXentry/NXinstrument sev=error dataPath=/entry
dataFile=NXopt_minimal_example.nxs
definition=NXoptical_spectroscopy.nxdl.xml message="Required group missing"
nxdlPath=/NXentry/NXsample sev=error dataPath=/entry
dataFile=NXopt_minimal_example.nxs
definition=NXoptical_spectroscopy.nxdl.xml message="Required group missing"
nxdlPath=/NXentry/NXdata sev=error dataPath=/entry
dataFile=NXopt_minimal_example.nxs
9 errors and 11 warnings found when validating NXopt_minimal_example.nxs
```

The errors tell you now, which things are missing (message="Required group missing"), if there is a field missing (message="Required field missing") or if an attribute is missing (message="Required attribute URL missing" - here for example the attribute named URL)

Now go to the hardcoded files, and add the respective fields to make your NeXus file compliant with the NeXus definitions. This way, you can ensure that your data is FAIR, which is then ready for sharing and publication.

3.3. Punx - Python Utilities for NeXus HDF5 files

This is python package, and can therefore be used on Linux and Windows systems.

The package can be installed via pip. Therefore you need to have installed:

1. python
2. pip

You can then evoke a command like this:

```
punx validate [-h] [--report REPORT] infile
```

"validate" is tells the program, that we want to validate a file

"[-h]" tells the program to show the help message

"[--report REPORT]" tells the program, what findings should be reported.
This is done by replacing REPORT with ={COMMENT,ERROR,NOTE,OK,TODO,UNUSED,WARN}

Official docs: https://punx.readthedocs.io/en/latest/validate.html#validate

3.3.1 installation

Open the terminal and install punx via pip:

```
pip install punx
```

This software is based on other powerful software packages or libraries, therefore as well other packages have to be installed:

```
pip install h5py
pip install lxml
pip install numpy
pip install PyQt5
pip install requests
pip install pyRestTable
```

Then you should be able to test the package by:

```
punx demo
```

The output should look like this:

```
C:\>punx demo

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


console> punx validate
C:\Users\USER\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kf
ra8p0\LocalCache\local-packages\Python310\site-packages\punx\data\writer_1_3.hdf5
data file:
C:\Users\USER\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kf
ra8p0\LocalCache\local-packages\Python310\site-packages\punx\data\writer_1_3.hdf5
NeXus definitions: main, dated 2024-01-02 03:04:05,
sha=xxxx21fxcef02xfbaa6x04e182e3d67dace7ef1b

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
/                           OPTIONAL NXDL group in data file               not
found:  in //entry
/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-Z0-9_]([a-zA-Z0-9_.]*[a-zA-Z0-9_])?
/Scan                       OPTIONAL NXDL field in data file               not
found: /Scan/collection_description
/Scan                       OPTIONAL NXDL field in data file               not
found: /Scan/collection_identifier
/Scan                       OPTIONAL NXDL field in data file               not
found: /Scan/collection_time
/Scan                       OPTIONAL NXDL field in data file               not
found: /Scan/definition
/Scan                       OPTIONAL NXDL field in data file               not
found: /Scan/definition_local
...
...
...
/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
/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       35    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 40    allowed by NeXus specification, not identified               99
         --
TOTAL    83
======== ===== ============================================================
=========

<finding>=99.144737 of 76 items reviewed
NeXus definitions version: main

console> punx tree
C:\Users\rh83hixu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5
n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\punx\data\writer_1_3.hdf5
C:\Users\rh83hixu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5
n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\punx\data\writer_1_3.hdf5 : NeXus data file
  Scan:NXentry
    @NX_class = "NXentry"
    data:NXdata
      @NX_class = "NXdata"
      @axes = "two_theta"
      @signal = "counts"
      @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"
```

Then you should be able to use this package.

Official docs for installation: https://punx.readthedocs.io/en/latest/install.html

3.3.2. Using punx

Open your terminal. Assuming there is a folder at:

For Linux:

```
/home/USER/nexusvalidation
```

For Windows:

```
C:\nexusvalidation
```

Put a NeXus file into this folder. For example, the file: "SiO2onSi.ellips.nxs" (INSERT LINK).

then the command is (for Windows):

```
punx validate C:\nexusvalidation\SiO2onSi.ellips.nxs
```

For Linux:

```
punx validate C:\nexusvalidation\SiO2onSi.ellips.nxs
```

The output tables "findings" and "summary statistics" can be used to find error present in the NeXus file.

3.3.3. Example

Which NeXus definition?

The program selects the NeXus definitions (set of nxdl.xml files) by itself. It can in principle also be modified with different repositories. The functionality to add a new repository is right now not possible, as it was removed due to incompatibility in a major update of punx.

Therefore, only the official repository is functional.

You may update the repositttory for the lastest version via:

```
punx install
```

The NeXus respective definitions are found here:

https://manual.nexusformat.org/classes/index.html

Search on the right side under "quick search" for "NXopt":

https://manual.nexusformat.org/classes/contributed_definitions/NXopt.html#index-0

This python code creates the respective python file with all required fields:

NXopt_minimal_example_NIAC_NeXus_Def.nxs (INSERT LINK)

Here is the python code:

h5py_nexus_file_creation_NIAC_NeXus_Def.py (INSERT LINK)

The command:

```
punx validate --report ERROR
C:\nexusvalidation\NXopt_minimal_example_NIAC_NeXus_Def.nxs
```

then gives this output:

```
findings
======= ====== ========== ======================================
address status test       comments
======= ====== ========== ======================================
/entry  ERROR  known NXDL NXopt: unrecognized NXDL specification
======= ====== ========== ======================================


summary statistics
======== ===== ============================================================
=========
status   count description                                                 (value)
======== ===== ============================================================
=========
OK       148   meets NeXus specification                                   100
NOTE     0     does not meet NeXus specification, but acceptable           75
WARN     0     does not meet NeXus specification, not generally acceptable 25
ERROR    1     violates NeXus specification
-10000000
TODO     16    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 213   allowed by NeXus specification, not identified              99
         --
TOTAL    378
======== ===== ============================================================
=========
```

The last error message:

```
======= ====== ========== ====================================
/entry  ERROR  known NXDL NXopt: unrecognized NXDL specification
======= ====== ========== ====================================
```

can be ignored and is a bug right now. If this is the only Error message, then your NeXus file is compliant with the NeXus definitions and you can share and publish your data.

3.3.4. Further Notes

1. Punx only uses the NeXus definiton from the NIAC (https://manual.nexusformat.org/) - The FAIRmat NeXus definition is not possible right now (https://fairmat-nfdi.github.io/nexus_definitions/index.html#)
2. Other punx commands are avaialble: https://punx.readthedocs.io/en/latest/overview.html#
3. More details for installation: https://punx.readthedocs.io/en/latest/install.html
4. Github project: https://github.com/prjemian/punx

3.4 pynxtools - Python Nexus Tools

This is python package which is developed by the FAIRmat consortium.

As python package, this can be used on Linux and Windows systems.

The package can be installed via pip. Therefore you need to have installed:

1. python
2. pip

This tool has 3 command line functions:

1. dataconverter (https://github.com/FAIRmat-NFDI/pynxtools/blob/master/src/pynxtools/dataconverter/README.md)
2. read_nexus (https://github.com/FAIRmat-NFDI/pynxtools/blob/master/src/pynxtools/nexus/README.md)
3. generate_eln (https://github.com/FAIRmat-NFDI/pynxtools/blob/master/src/pynxtools/eln_mapper/README.md)

For validation purposes, we will use the "read_nexus" function.

The command used is:

```
read_nexus -f NXopt_minimal_example.nxs
```

The output looks like this, if the respective entry is found:

```
DEBUG: ===== FIELD (//entry/experiment_type): <HDF5 dataset "experiment_type":
shape (), type "|O">
DEBUG: value: b'transmission spectroscopy'
DEBUG: classpath: ['NXentry', 'NX_CHAR']
DEBUG: classes:
NXoptical_spectroscopy.nxdl.xml:/ENTRY/experiment_type
DEBUG: <<REQUIRED>>
DEBUG: enumeration (NXoptical_spectroscopy.nxdl.xml:/ENTRY/experiment_type):
DEBUG: -> photoluminescence
DEBUG: -> transmission spectroscopy
DEBUG: -> reflection spectroscopy
DEBUG: -> other
DEBUG: documentation (NXoptical_spectroscopy.nxdl.xml:/ENTRY/experiment_type):
DEBUG:
                Specify the type of the optical experiment.

                Chose other if none of these methods are suitable. You may
specify
                fundamental characteristics or properties in the experimental
sub-type.

                For Raman spectroscopy or ellipsometry use the respective
specializations
                of NXoptical_spectroscopy.
```

or like this, if the respective entry is not found in the defintion:

```
DEBUG: ===== ATTRS (//entry/instrument/software_RC2/program@url)
DEBUG: value: https://www.jawoollam.com/ellipsometry-software/completeease
DEBUG: classpath: ['NXentry', 'NXinstrument']
DEBUG: NOT IN SCHEMA
DEBUG:
```

The first example was for for "experiment_type" entry in the "NXoptical_spectroscopy" definition.

The second example was for the "software_TYPE" attribute @URL entry in the "NXoptical_spectroscopy" definition. Here the problem was that "url" was used instead of "URL".

3.4.1 Installation

This is installed with pip:

```
pip install pynxtools
```

3.4.2 Using the read_nexus function

Open your terminal. Assuming there is a folder at:

For Linux:

```
/home/USER/nexusvalidation
```

For Windows:

```
C:\nexusvalidation
```

Put into this folder your NeXus file, for example the Raman.nxs file (INSERT LINK).

Then use:

```
read_nexus -f C:\nexusvalidation\Raman.nxs
```

shows the output like this:

```
===== FIELD (//entry/data/spectrum_data_y): <HDF5 dataset "spectrum_data_y": shape
(1600,), type "<f8">
DEBUG: ===== FIELD (//entry/data/spectrum_data_y): <HDF5 dataset
"spectrum_data_y": shape (1600,), type "<f8">
value: [ 288.5499878  289.         288.4500122 ... 1875.        1889.349976 ...
DEBUG: value: [ 288.5499878  289.         288.4500122 ... 1875.        1889.349976
...
Dataset referenced as NXdata SIGNAL
DEBUG: Dataset referenced as NXdata SIGNAL
===== ATTRS (//entry/data/spectrum_data_y@long_name)
DEBUG: ===== ATTRS (//entry/data/spectrum_data_y@long_name)
value: Raman Intensity
DEBUG: value: Raman Intensity
Dataset referenced as NXdata SIGNAL
DEBUG: Dataset referenced as NXdata SIGNAL
===== ATTRS (//entry/data/spectrum_data_y@units)
DEBUG: ===== ATTRS (//entry/data/spectrum_data_y@units)
value: counts
DEBUG: value: counts

DEBUG:
For Axis #0, 1 axes have been identified: [<HDF5 dataset "spectrum_data_x_Raman":
shape (1600,), type "<f8">]
DEBUG: For Axis #0, 1 axes have been identified: [<HDF5 dataset
"spectrum_data_x_Raman": shape (1600,), type "<f8">]
```

Search for filed which are not found in the NeXus definiton by searching for the line: "DEBUG: NOT IN SCHEMA". Recheck the used NeXus definition to eliminate the problem. Be careful with upper and lower case notation and correct spelling.

Keep in mind, that the output provides quite some information. This is useful for software development, but may be a bit too much for validation purposes.

Similar features as the tables and messages provided from punx and cnxvalidate for pynxtools are planned to be implemented in the future (https://github.com/FAIRmat-NFDI/pynxtools/pull/333).

4. Summary

This tutorial showed:

1. How to create a NeXus file with python.
2. How to check if the NeXus file is valid.

This provides the basics and fundamentals to create FAIR data, based on NeXus definitions. If your experimental setup provides enough meta data, you can extend the NeXus file creation script, to automaticalally include this information (e.g. measured spectra, sensor temperature, stage positon).

Pynxtools Parsers:

For a specifically structured set of data, a parser can be written, which uses the meta data and a pre-structured meta data file, to create a NeXus file. Tough, the parser depends on: Experimental Technique and Setup and has therefore to be written invidiually. This is another functionallity of pynxtools with plugins for the techniques:

electron microscopy (EM): https://github.com/FAIRmat-NFDI/pynxtools-em

x-ray photoelectron spectroscopy (XPS): https://github.com/FAIRmat-NFDI/pynxtools-xps

scanning tunneling spectroscopy/microscopy and atomic force microscopy (STS / STM /AFM) : https://github.com/FAIRmat-NFDI/pynxtools-stm

x-ray diffraction (XRD): https://github.com/FAIRmat-NFDI/pynxtools-xrd

ellipsometry (ellips): https://github.com/FAIRmat-NFDI/pynxtools-ellips

Raman spectroscopy (raman): https://github.com/FAIRmat-NFDI/pynxtools-raman

atom probe microscopy (APM): https://github.com/FAIRmat-NFDI/pynxtools-apm

Feedback and contact:

????