

h5cpp and pninexus libraries

DESY HDF5 meeting 2023

Jan Kotański

Deutsches Elektronen-Synchrotron



Sep 20, 2023

PETRA III Beamlines



h5cpp: c++ library which wraps the hdf5 library



EUROPEAN
SPALLATION
SOURCE

- uses c++ class objects to wrap pointers and hid's,
we avoid `new`, `delete` and `close()`
- maps native c++ datatypes to hdf5 datatypes
making use of c++ traits, `create()` and `get()` methods
- maps native std c++ containers to hdf5 dataspace
making use of c++ traits, `create()` and `get()` methods

h5cpp: c++ library which wraps the hdf5 library



EUROPEAN
SPALLATION
SOURCE

- uses c++ class objects to wrap pointers and hid's, we avoid `new`, `delete` and `close()`
- maps native c++ datatypes to hdf5 datatypes making use of c++ traits, `create()` and `get()` methods
- maps native std c++ containers to hdf5 dataspace making use of c++ traits, `create()` and `get()` methods

pninexus: c++ library which adds DESY/NeXus functionality



- XML builder: `create_from_string(...)`, `create_from_file(...)`, ...
- NeXus specific: `create_file(...)`, `get_objects(...)`, ...
- maps non-std c++ dataspace and datatypes to hdf5 datatypes with c++ traits

Description

h5cpp: c++ library which wraps the hdf5 library



EUROPEAN
SPALLATION
SOURCE

- uses c++ class objects to wrap pointers and hid's, we avoid `new`, `delete` and `close()`
- maps native c++ datatypes to hdf5 datatypes making use of c++ traits, `create()` and `get()` methods
- maps native std c++ containers to hdf5 dataspace making use of c++ traits, `create()` and `get()` methods

pninexus: c++ library which adds DESY/NeXus functionality



- XML builder: `create_from_string(...)`, `create_from_file(...)`, ...
- NeXus specific: `create_file(...)`, `get_objects(...)`, ...
- maps non-std c++ dataspace and datatypes to hdf5 datatypes with c++ traits

python-pnexus: simple boost python bindings



- maps numpy objects to hdf5 datatypes and dataspace with c++ traits

NeXus format and HDF5

- NeXus is a set of rules how data must be organized

within a particular format in order to become a valid file.

NeXus International Advisory Committee at <https://www.nexusformat.org>

- Every NeXus file written by us is also a valid HDF5 file!

Hierarchical Data Format v5: file format that supports large, complex, heterogeneous data



NeXus

Layer 3: application definitions

Layer 2: base classes

Layer 1: nxgroup, nxfield, nxattribute, nxlink, types

- scalar and multidimensional data within a single file
- data within the file in a tree like manager
- additional attributes can be attached to objects in a file storing metadata which might be required for later analysis

Tango Servers:

Lambda, AGIPD, PCO,
FireBird, Varex,
Vimba, DALSA, Andor,
PiLC, Express3, ...



Yuelong, Yu, Teresa Nunez,
Johannes Blume, Jan Garrevoet, ...



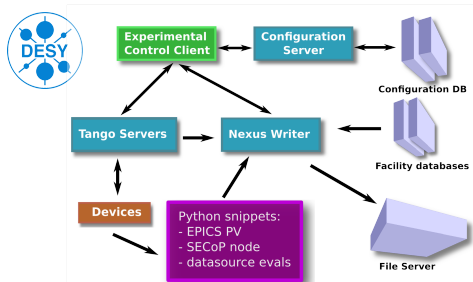
Tango Servers:

Lambda, AGIPD, PCO,
FireBird, Varex,
Vimba, DALSA, Andor,
PiLC, Express3, ...

Yuelong, Yu, Teresa Nunez,
Johannes Blume, Jan Garrevoet, ...



NeXus Metadata Framework:



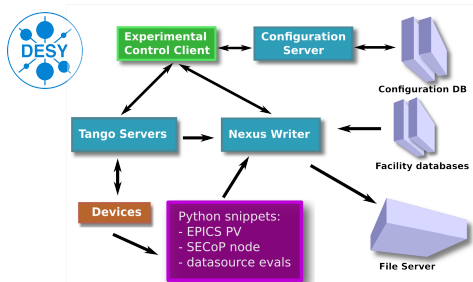
Tango Servers:

Lambda, AGIPD, PCO,
FireBird, Varex,
Vimba, DALSA, Andor,
PiLC, Express3, ...

Yuelong, Yu, Teresa Nunez,
Johannes Blume, Jan Garrevoet, ...



NeXus Metadata Framework:



PLUG2: Online XPCS Analysis

Vijay Kartik



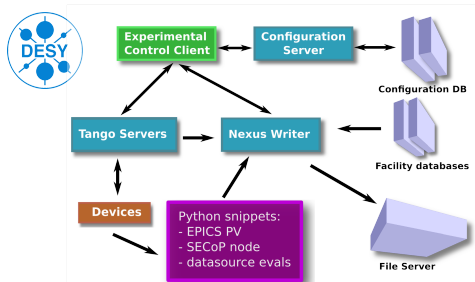
Tango Servers:

Lambda, AGIPD, PCO,
FireBird, Varex,
Vimba, DALSA, Andor,
PiLC, Express3, ...

Yuelong, Yu, Teresa Nunez,
Johannes Blume, Jan Garrevoet, ...



NeXus Metadata Framework:



PLUG2: Online XPCS Analysis

Vijay Kartik



European Spallation Source:

Kafka-to-NeXus, ...



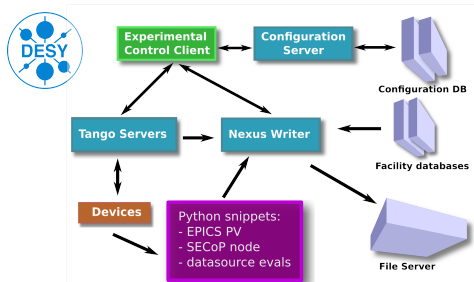
Tango Servers:

Lambda, AGIPD, PCO,
FireBird, Varex,
Vimba, DALSA, Andor,
PiLC, Express3, ...

Yuelong, Yu, Teresa Nunez,
Johannes Blume, Jan Garrevoet, ...



NeXus Metadata Framework:



PLUG2: Online XPCS Analysis

Vijay Kartik



European Spallation Source:

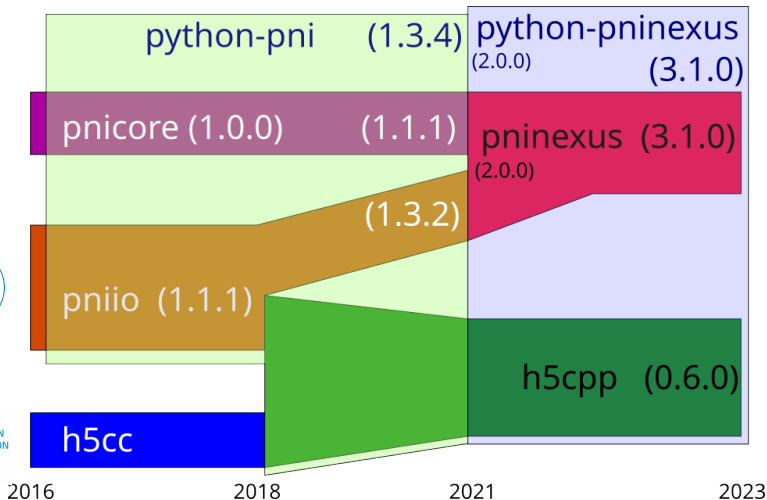
Kafka-to-NeXus, ...



Other facilities and also outside science

h5cc, pnii + pnicore $\Rightarrow$ h5cpp + pninexus

c++ wrapping libraries of hdf5 and their python bindings



Eugen Wintersberger (DESY), Martin Shetty (ESS), Jan Kotanski (DESY), Matthew D Jones (STFC/ESS), Alfonso Mukai (ESS), Jonas Nilsson (ESS), Sebastian Koenig (NC), Yuelong Yu (DESY), Andrew Marshall (UK), Shiv Upadhyay (WA), ...

Installation from sources

- **libh5cpp** from github

```
git clone https://github.com/ess-dmnc/h5cpp.git
cd h5cpp && mkdir build && cd build
cmake -DCMAKE_INSTALL_PREFIX=/opt/h5cpp -DCMAKE_BUILD_TYPE=Release \
    -DH5CPP_CONAN=DISABLE -DH5CPP_DISABLE_TESTS=True \
    -DH5CPP_WITH_BOOST=False ../
make install -j4 && cd ../../
```

- **libpninexus** from github

```
git clone https://github.com/pni-libraries/libpninexus.git
cd libpninexus && mkdir build && cd build
cmake -DCMAKE_INSTALL_PREFIX=/opt/pninexus \
    -Dh5cpp_DIR=/opt/h5cpp/lib/cmake/h5cpp-0.6 \
    -DPNINEXUS_CONAN=DISABLE -DCMAKE_BUILD_TYPE=Release ../
make install -j4
```

- **python-pninexus** from github

```
git clone https://github.com/pni-libraries/python-pninexus.git
H5CPP_LOCAL_PATH=/opt/h5cpp PNINEXUS_LOCAL_PATH=/opt/pninexus \
    python3 setup.py install
```

details instructions on their [github](#) CI or [documentation](#)

Debian packages from our HDRI repository

- **add the HDRI repository**

```
curl -s http://repos.pni-hdri.de/debian_repo.pub.gpg | gpg \  
  --import --no-default-keyring \  
  --keyring gnupg-ring:/etc/apt/trusted.gpg.d/debian-hdri-repo.gpg  
chmod 644 /etc/apt/trusted.gpg.d/debian-hdri-repo.gpg  
cd /etc/apt/sources.list.d  
wget http://repos.pni-hdri.de/bookworm-pni-hdri.list  
apt-get update
```

- **install h5cpp**

```
apt-get install libh5cpp0.6.0 libh5cpp0.6.0-dev \  
  libh5cpp0.6.0-doc libh5cpp0.6.0-dbg
```

- **install pninexus**

```
apt-get install libpninexus3.1.0 libpninexus3.1.0-dev \  
  libpninexus3.1.0-doc libpninexus3.1.0-dbg
```

- **install python-pninexus**

```
apt-get install python3-pninexus
```

Also for **buster**, **bullseye** debian and **focal**, **jammy**, **lunar** ubuntu.

Using h5cpp with cmake

minimal CMakeLists.txt file

```
# project name
project(h5cpp_create_dataset)

# cmake required version
cmake_minimum_required(VERSION 3.13...3.20)

# we use c++17 standard
set(CMAKE_CXX_STANDARD 17)

# find h5cpp
find_package(H5CPP REQUIRED)

# define executable and its content
add_executable(h5cpp_create_dataset h5cpp_create_dataset.cpp)

# link h5cpp::h5cpp and std::filesystem
target_link_libraries(h5cpp_create_dataset PRIVATE h5cpp::h5cpp stdc++fs)
```

from https://gitlab.desy.de/h5cpp-pninxus-examples/h5cpp_create_dataset.git

Create a dataset with h5cpp: tree structure

```
#include <h5cpp/hdf5.hpp>

int main()
{
    using namespace hdf5;

    // create a file
    file::File f = file::create("myscan_00123.nxs", file::AccessFlags::Truncate);

    // create an entry group
    node::Group root_group = f.root();

    // create a root group attribute
    root_group.attributes.create<std::string>("default").write("entry");

    // create an entry group
    node::Group entry_group = root_group.create_group("entry");

    entry_group.create_dataset("title",
                              datatype::get<std::string>(),
                              dataspace::Scalar()).write("My scan");

    // create entry group attributes
    entry_group.attributes.create<std::string>("NX_class").write("NXentry");
    entry_group.attributes.create<std::string>("default").write("data");

    // create an data group
    node::Group data_group = entry_group.create_group("data");

    // create a data group attributes
    data_group.attributes.create<std::string>("NX_class").write("NXdata");
    data_group.attributes.create<std::string>("signal").write("counter");
    data_group.attributes.create<std::string>("axes").write("x");
```


Create a dataset with h5cpp: write data

```
// create a counter dataset
using int_type = std::vector<int>;
int_type counter_data {0, 1, 2, 4, 6, 5, 1, 0};
node::Dataset cdataset = data_group.create_dataset("counter",
                                                    datatype::get<int_type>(),
                                                    dataspace::create(counter_data));

// set units attribute
cdataset.attributes.create<std::string>("units").write("counts");

// write to counter dataset
cdataset.write(counter_data);

// create a motor dataset
using float_type = std::vector<double>;
float_type x_data {2.01, 2.98, 4.01, 4.99, 6.00, 7.1, 8.2, 9.1};
node::Dataset xdataset = data_group.create_dataset("x",
                                                    datatype::get<float_type>(),
                                                    dataspace::create(x_data));

// set units attribute
xdataset.attributes.create<std::string>("units").write("mm");

// write to x dataset
xdataset.write(x_data);
}
```

Create a dataset with h5cpp: view in NeXpy

Compiling:

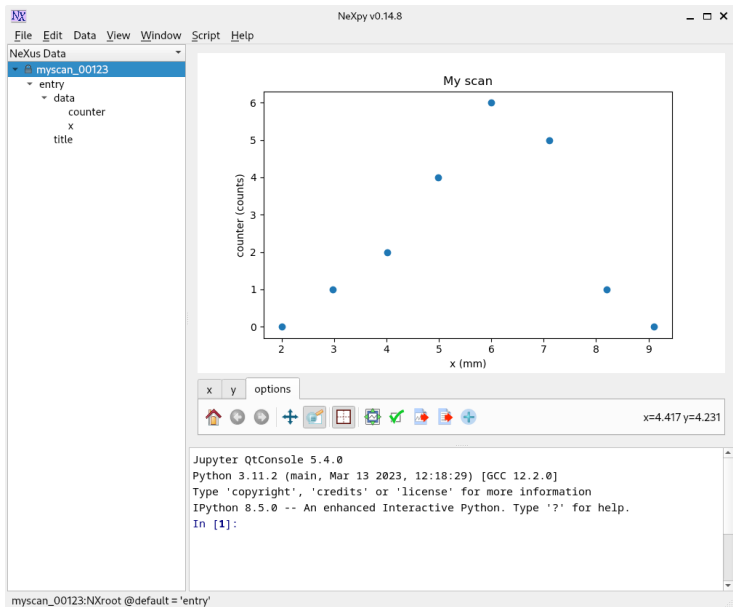
```
mkdir build
cd build
cmake ../
make
```

Executing:

```
./h5cpp_create_dataset
```

View in NeXpy:

```
default plot
units
title
```



Append data with h5cpp

```
const long long unsigned int xdim = 1024;
const long long unsigned int nframe = 133;

std::vector<unsigned short int> mca_data(xdim);

// define our chunk size
property::DatasetCreationList dcpl;
dcpl.layout(property::DatasetLayout::Chunked);
dcpl.chunk({1, xdim});

// compression filter
filter::Deflate filter(2u);
filter(dcpl);

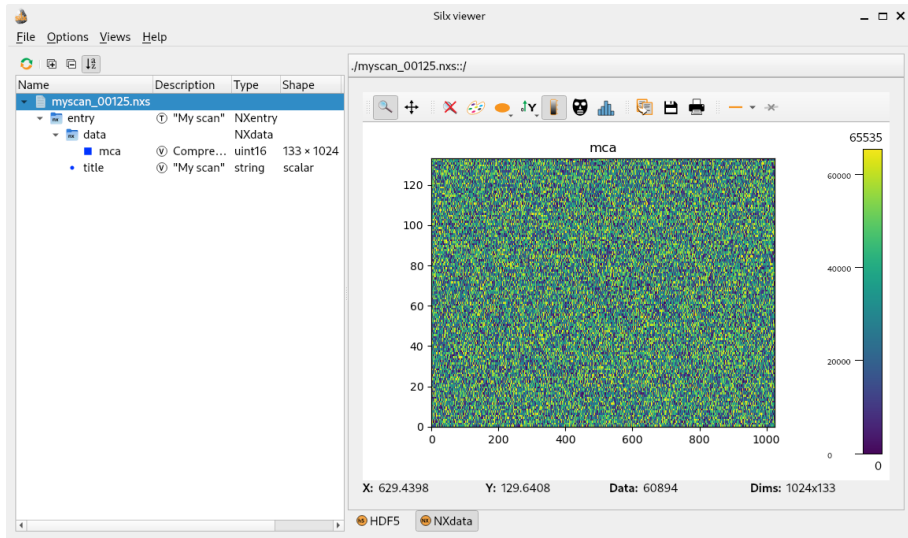
// create a dataset
using hdf5::dataspace::Simple;
node::Dataset dataset =
    data_group.create_dataset("mca", datatype::get<unsigned short int>(),
                             Simple({0, xdim}, {Simple::unlimited, Simple::unlimited}),
                             dcpl);

// define hyperslab of the chunk size
dataspace::Hyperslab framespace{{0, 0}, {1, xdim}};

for (long long unsigned int i = 0; i != nframe; i++) {
    // generate test data
    std::generate(mca_data.begin(), mca_data.end(), std::rand);

    // extend dataset, update hyperslab offset, write data
    dataset.extent(0, 1);
    framespace.offset({i, 0});
    dataset.write(mca_data, framespace);
}
// from https://gitlab.desy.de/h5cpp-pninxus-examples/h5cpp_append_dataset.git
```

Append data with h5cpp: default plot



Append data with h5cpp: chunked dataset

The screenshot displays the Silx viewer application. On the left, a tree view shows the file structure of `myscan_00125.nxs`, with the `mca` dataset selected. The right pane provides detailed information about this dataset.

HDF5 Dataset

Path info

Basename	mca
Name	/entry/data/mca
Local	./myscan_00125.nxs::/entry/data/mca
Physical	./myscan_00125.nxs::/entry/data/mca

Data info

HDF5 type	INTEGER
dtype	Native uint16
shape	$133 \times 1024 = 136192$
chunks	$1 \times 1024 = 1024$

Compression info

Position	HDF5 ID	Name	Options
0	1	"deflate"	2

At the bottom of the right pane, there are view mode buttons: **HDF5** (selected), **Curve**, **Image**, and **Raw**.

Using pnnexus with cmake

minimal CMakeLists.txt file

```
# project name
project(pnnexus_from_xml)

# cmake required version
cmake_minimum_required(VERSION 3.13...3.20)

# we use c++17 standard
set(CMAKE_CXX_STANDARD 17)

# find h5cpp and pnnexus
find_package(H5CPP REQUIRED)
find_package(PNINEXUS REQUIRED)

# define executable and its content
add_executable(pnnexus_from_xml pnnexus_from_xml.cpp)

# link h5cpp::h5cpp pnnexus and std::filesystem
target_link_libraries(pnnexus_from_xml PRIVATE h5cpp::h5cpp pnnexus stdc++fs)
```

from https://gitlab.desy.de/h5cpp-pnnexus-examples/pnnexus_from_xml.git

NeXus from xml: xml configuration

```
#include <h5cpp/hdf5.hpp>
#include <pni/nexus.hpp>

int main()
{
    using namespace hdf5;
    using namespace pni;

    const long long unsigned int xdim = 256;
    const long long unsigned int ydim = 512;
    const long long unsigned int nframe = 133;

    // NeXus configuration
    std::ostringstream xmlcnf;
    xmlcnf << "<attribute name=\"default\" type=\"string\">entry</attribute>"
        << "<group name=\"entry\" type=\"NXentry\">"
        << "  <attribute name=\"default\" type=\"string\">data</attribute>"
        << "  <field name=\"title\" type=\"string\">My scan</field>"
        << "  <group name=\"instrument\" type=\"NXinstrument\">"
        << "    <group name=\"lambda\" type=\"NXdetector\">"
        << "      <field name=\"data\" type=\"uint32\">"
        << "        <dimensions rank=\"3\">"
        << "          <dim value=\"0\" index=\"1\" />"
        << "          <dim value=\"\" << xdim << \"\" index=\"2\" />"
        << "          <dim value=\"\" << ydim << \"\" index=\"3\" />"
        << "        </dimensions>"
        << "        <strategy compression=\"true\" rate=\"2\" shuffle=\"false\" />"
        << "      </field>"
        << "    </group>"
        << "  </group>"
        << "  <group name=\"data\" type=\"NXdata\">"
        << "    <attribute name=\"signal\" type=\"string\">lambda</attribute>"
        << "    <link name=\"lambda\" target=\"/entry/instrument/lambda/data\" />"
        << "  </group>"
        << "</group>";
```

NeXus from xml: create a file and write data

```
// create a file
file::File f =
    nexus::create_file("myscan_00245.nxs", file::AccessFlags::Truncate);

// create metadata from the xml string
nexus::xml::create_from_string(f.root(), xmlcnf.str());
node::Dataset dataset =
    node::get_dataset(f.root(), "/entry/instrument/lambda/data");

// detector image buffer
std::vector<unsigned short int> det_data(xdim * ydim);

// hyperslab of the chunk size
dataspace::Hyperslab framespace{{0, 0, 0}, {1, xdim, ydim}};

for (long long unsigned int i = 0; i != nframe; i++) {
    // generate test data
    std::generate(det_data.begin(), det_data.end(), std::rand);

    // extend dataset, update hyperslab offset, write data
    dataset.extent(0, 1);
    framespace.offset({i, 0, 0});
    dataset.write(det_data, framespace);
}
}
```


NeXus from xml: root attributes

The screenshot displays the Silx viewer application. The left pane shows a tree view of the NeXus data structure for the file `./myscan_00245.nxs`. The tree structure is as follows:

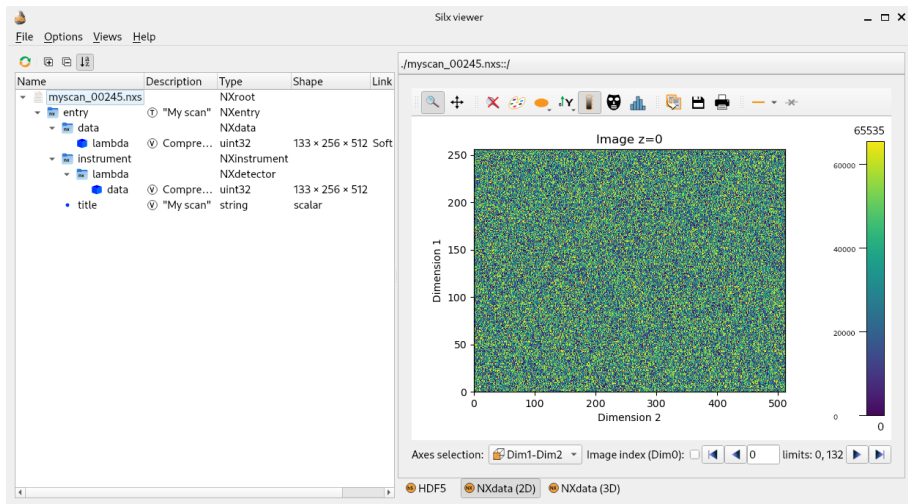
- `myscan_00245.nxs` (NXroot)
 - `entry` ("My scan") (NXentry)
 - `data` (NXdata)
 - `lambda` (uint32, 133 x 256 x 512, Soft)
 - `instrument` (NXinstrument)
 - `lambda` (NXdetector)
 - `data` (uint32, 133 x 256 x 512)
 - `title` ("My scan") (string, scalar)

The right pane shows the root attributes for the file `./myscan_00245.nxs:/`. The attributes are:

HDF5 File	
Path info	
Basename	
Name	/
Local	./myscan_00245.nxs:/
Physical	./myscan_00245.nxs:/
Attributes	
HDF5_Version	"1.10.8"
NX_class	"NXroot"
default	"entry"
file_name	"myscan_00245.nxs"
file_time	"2023-09-14T11:51:06.151887+0200"
file_update_time	"2023-09-14T11:51:06.151978+0200"

At the bottom of the right pane, there are three buttons: `HDF5`, `NXdata (2D)`, and `NXdata (3D)`.

NeXus from xml: default plot



NeXus from xml: chunked dataset

The screenshot displays the Silx viewer application. On the left, a tree view shows the hierarchy of the 'myscan_00245.nxs' file. The 'lambda' dataset is selected, showing its properties: Type 'uint32', Shape '133 x 256 x 512', and Soft. On the right, the 'HDF5 Dataset' details for 'myscan_00245.nxs::/entry/data/lambda' are shown. The details include Path info, Data info, and Compression info.

HDF5 Dataset			
Path info			
Basename	lambda		
Name	/entry/data/lambda		
Local	myscan_00245.nxs::/entry/data/lambda		
Physical	myscan_00245.nxs::/entry/instrument/lambda/data		
Data info			
HDF5 type	INTEGER		
dtype	Native uint32		
shape	133 x 256 x 512 = 17432576		
chunks	1 x 256 x 512 = 131072		
Compression info			
Position	HDF5 ID	Name	Options
0	1	"deflate"	2

At the bottom of the right panel, there are buttons for different data representations: HDF5, Curve, Image, Cube, Raw, and Image stack.

python bindings: xml configuration

```
import numpy

from pninexus import h5cpp
from pninexus import nexus

xml_config = """
<attribute name="default" type="string">entry</attribute>
<group name="entry" type="NXentry">
  <attribute name="default" type="string">data</attribute>
  <field name="title" type="string">My scan</field>
  <group name="instrument" type="NXinstrument">
    <group name="mythen" type="NXdetector">
      <field name="data" type="uint16">
        <dimensions rank="2">
          <dim value="0" index="1" />
          <dim value="17" index="2" />
        </dimensions>
        <strategy compression="true" rate="2" shuffle="false" />
      </field>
    </group>
  </group>
  <group name="data" type="NXdata">
    <attribute name="signal" type="string">mythen</attribute>
    <link name="mythen" target="/entry/instrument/mythen/data" />
  </group>
</group>
"""
```

python bindings: white chunks directly

```
# create nexus file
mfile = nexus.create_file("myscan_00678.nxs", h5cpp.file.AccessFlags.TRUNCATE)
root = mfile.root()

# create nexus file structure
nexus.create_from_string(root, xml_config)

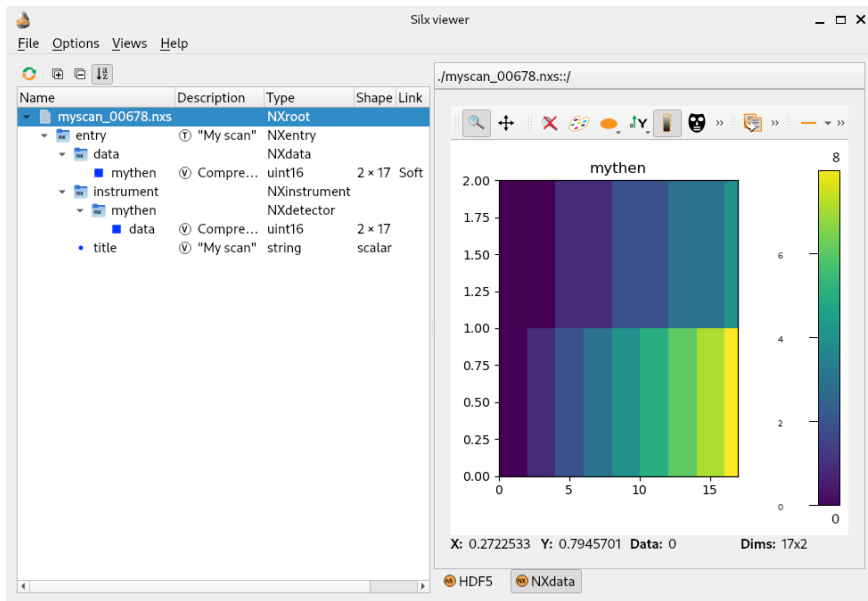
# get dataset directly
dataset = root.get_dataset(h5cpp.Path("/entry/instrument/mythen/data"))

# get datasets by group types (optional)
datasets = nexus.get_objects(root,
    nexus.Path.from_string(":NXentry/:NXinstrument/:NXdetector/data"))
if datasets:
    dataset = datasets[0]

# compressed detector frames
frame1 = numpy.array(
    [24184, 49677, 4609, 12288, 49156, 6832, 65478, 44127,
     151, 40580, 42322, 55254, 14696, 2563, 16640], dtype="uint16")
frame2 = numpy.array(
    [24184, 24675, 128, 1606, 25608, 32866, 26176, 2054, 24932, 0,
     20993, 7424], dtype="uint16")
frames = [frame1, frame2]

# write first direct
for i in range(len(frames)):
    dataset.extent(0, 1)
    dataset.write_chunk(frames[i], [i, 0])
```

python bindings: default plot



python bindings: root attributes

Silx viewer

File Options Views Help

myscan_00678.nxs

Name	Description	Type	Shape	Link
myscan_00678.nxs		NXroot		
entry	"My scan"	NXentry		
data		NXdata		
mythen	Compre...	uint16	2 x 17	Soft
instrument		NXinstrument		
mythen		NXdetector		
data	Compre...	uint16	2 x 17	
title	"My scan"	string	scalar	

./myscan_00678.nxs:/

HDF5 File

Path info

Basename	
Name	/
Local	./myscan_00678.nxs:/
Physical	./myscan_00678.nxs:/

Attributes

HDF5_Version	"1.10.8"
NX_class	"NXroot"
default	"entry"
file_name	"myscan_00678.nxs"
file_time	"2023-09-14T14:30:41.080296+0200"
file_update_tir	"2023-09-14T14:30:41.080326+0200"

HDF5 NXdata

python bindings: dataset details

The screenshot shows the Silx viewer application. On the left is a tree view of the dataset structure. The main panel on the right displays the details for the selected dataset, `./myscan_00678.nxs::entry/data/mythen`.

Tree View:

Name	Description	Type
myscan_00678.nxs		NXroot
entry	"My scan"	NXentry
data		NXdata
mythen	Compressed	uint16
instrument		NXinstrument
mythen		NXdetector
data	Compressed	uint16
title	"My scan"	string

Dataset Details:

HDF5 Dataset

Path info

Basename	mythen
Name	/entry/data/mythen
Local	./myscan_00678.nxs::entry/data/mythen
Physical	./myscan_00678.nxs::entry/instrument/mythen/data

Data info

HDF5 type	INTEGER
dtype	Native uint16
shape	2 × 17 = 34
chunks	1 × 17 = 17

Compression info

Position	HDF5 ID	Name	Options	
0	1	"deflate"	2	Available

Viewer Controls:

Buttons: HDF5, Curve, Image, Raw

Summary

- The **h5cpp** and **pninexus** libraries simplify our work with the **HDF5** library: **SWMR**, external filters, direct chunk access, **VDS**, user datatypes/dataspaces with c++ traits, ...
- They are widely used also outside the science (**h5cpp**)
- We test them on: **Linux** (**h5cpp**, **pninexus**), **Windows** (**h5cpp**, **pninexus**) and **MacOS** (**h5cpp**)
- Open Source Projects:
<https://github.com/ess-dmsc/h5cpp.git>
<https://github.com/pni-libraries/libpninexus.git>
<https://github.com/pni-libraries/python-pninexus.git>
- **Debian/Ubuntu packages** are available in our **HDRI repository**

...

Summary

- The **h5cpp** and **pninexus** libraries simplify our work with the **HDF5** library: **SWMR**, external filters, direct chunk access, **VDS**, user datatypes/dataspaces with c++ traits, ...
- They are widely used also outside the science (**h5cpp**)
- We test them on: **Linux** (**h5cpp**, **pninexus**), **Windows** (**h5cpp**, **pninexus**) and **MacOS** (**h5cpp**)
- Open Source Projects:
<https://github.com/ess-dmnc/h5cpp.git>
<https://github.com/pni-libraries/libpninexus.git>
<https://github.com/pni-libraries/python-pninexus.git>
- **Debian/Ubuntu packages** are available in our **HDRI repository**

Thank You !