PAUL SCHERRER INSTITUT

Sven Augustin :: SCD/AWI/EIDO (7901) :: Paul Scherrer Institut

Tools at SwissFEL: slic & sfdata

AWI Bi-Monthly Meeting – 3rd May 2022

SwissFEL **L**ibrary for **I**nstrument **C**ontrol

slic – Overview

- Python library → toolbox for creating
  - control environments,
  - automation scripts,
  - GUIs

  for experiments.

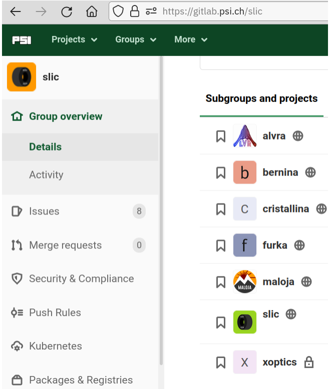
- Common experiment control system for all* SwissFEL instruments.
  → Used at:
  - Alvra
  - Cristallina
  - Furka
  - Maloja

- Needs & Goals:
  - CLI (ipython), scripting and GUI.
  - Maximum flexibility for rapidly changing endstations.
  - Extensible by BL scientists / external users with minimal training.

---

*OK, almost all...

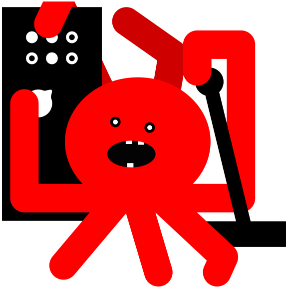
slic – Organization



- Clear separation between:
  - Experiment control library
  - Endstation codes
- Different parts may move with different speed.
- Clear border between working and in-development code:
  - New features can be build for an individual endstation,
  - when ready, they may be generalized and moved into the common library.

slic – Overview

- High-level layer combining the different services:
  - **epics devices**
  - custom / non-epics / user devices
  - **sf-daq**
  - bsread
  - epics monitors
  - DataAPI (epics archiver, image-/databuffer)
  - etc.
- Scan engine
- General-purpose devices
- Straight-forward building of complex devices from various components
  → Hardware abstraction layer



Press all the buttons!
Dial all the knobs!

slic – Hardware Abstraction

Hardware abstraction layer:

- `Adjustable`: single component / scannable axis
- `Device` assembled from `Adjustables` and other `Devices`

Bridges gap(s) between internal hardware implementation and user-facing coherent device representation.

Example:

- Change the FEL photon energy (== one `Adjustable`) ← *User*
- via *n* undulator gaps (== *n* epics PVs) ← *Controls*
- while maintaining the taper (== some math) ← *Beam Dynamics*

PAUL SCHERRER INSTITUT

slic – Hardware Abstraction

Built-in `Adjustable` types:

```
from slic.devices import Motor

mot = Motor("SPOES10-MANIP1:MOT1", name="Our favorite motor")
```

```
from slic.core import PVAdjustable

# with moving status PV
laser_delay = PVAdjustable(
    "SPOES10-LASER:SETVALUE",
    "SPOES10-LASER:READBACK",
    "SPOES10-LASER:MOVING",
    name="Laser Delay"
)

# without moving status PV
trigger_delay = PVAdjustable(
    "SPOES10-CVME-EVR0:Pul1-Delay-SP",
    "SPOES10-CVME-EVR0:Pul1-Delay-RB",
    accuracy=1,
    name="Trigger Delay"
)
```

etc. etc.

slic – Hardware Abstraction

New `Adjustable` type definition:

```
from slic.core import Adjustable

class MyNewCoolThing(Adjustable):

    pos = 0

    def get_current_value(self):
        return self.pos

    def set_target_value(self, value):
        self.pos = value

    def is_moving(self):
        return False # OK OK, this is probably cheating ;)

cool = MyNewCoolThing(name="My New Cool Thing")
```

- Useful built-in methods: `adj.tweak(delta)`, ...
- and shorthands: `adj.set(value)`, `adj.moving` property, ...
- **Appears automatically in the GUI**

slic – Hardware Abstraction

Device definition:

```
from slic.devices import Motor, SimpleDevice

mot_x = Motor("SPOES21-STAGE1:MOT_X", name="X")
mot_y = Motor("SPOES21-STAGE1:MOT_Y", name="Y")
mot_z = Motor("SPOES21-STAGE1:MOT_Z", name="Z")

stage3d = SimpleDevice("3D Stage", x=mot_x, y=mot_y, z=mot_z)

stuff = SimpleDevice("All our stuff",
    stages=SimpleDevice("Stages", stage3d=stage3d),
    some_other_thing=dummy
)
```

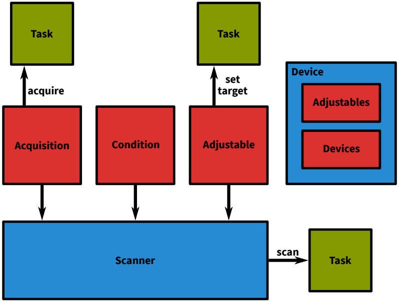
Interactive usage:

```
>>> stage3d
3D Stage:
---------
x: 10.2 mm
y: 0.1 mm
z: 123.4 mm

>>> stage3d.x
Motor "X" at 10.2 mm
```

```
>>> stuff
All our stuff:
--------------
some_other_thing: 1000 au
stages.stage3d.x: 10.2 mm
stages.stage3d.y: 0.1 mm
stages.stage3d.z: 123.4 mm

>>> stuff.stages
Stages:
-------
stage3d.x: 10.2 mm
stage3d.y: 0.1 mm
stage3d.z: 123.4 mm
```

slic – Components

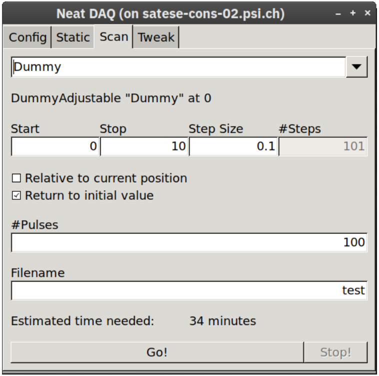
```
daq = SFAcquisition(
    instrument, pgroup,
    default_channels=channels
)

check_intensity = PVCondition(
    "SARFE10-PBPG050:INTENSITY",
    vmin=0, vmax=1500,
    wait_time=3, required_fraction=0.8
)

scan = Scanner(
    default_acquisitions=[daq],
    condition=check_intensity
)

gui = GUI(scan)
```




```
scan.scan1D(
    adjustable, start_pos, end_pos, step_size,
    n_pulses, filename,
    relative=False, return_to_initial_values=True, repeat=1, ...
)
```

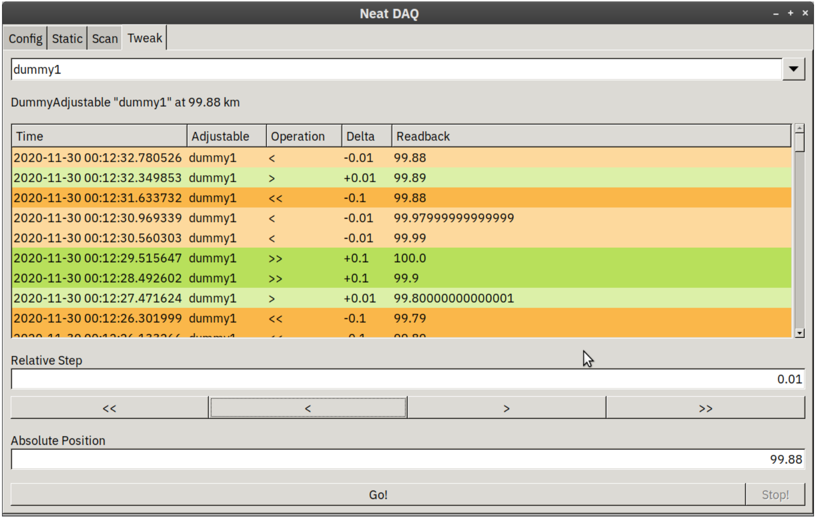
PAUL SCHERRER INSTITUT

slic – GUI

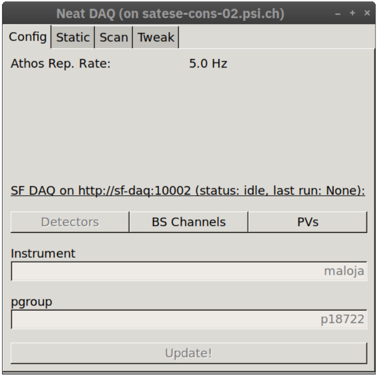


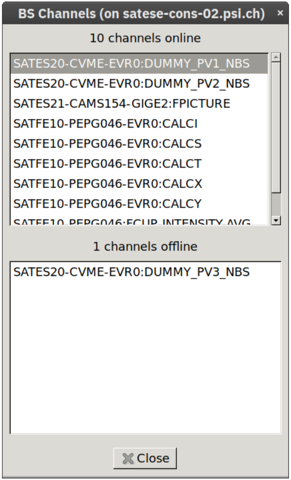

```
scan.scan1D(
    adjustable, start_pos, end_pos, step_size,
    n_pulses, filename,
    relative=False, return_to_initial_values=True, repeat=1, ...
)
```

slic – GUI


Neat DAQ
Config
Static
Scan
Tweak
dummy1
DummyAdjustable "dummy1" at 99.88 km
Time
Adjustable
Operation
Delta
Readback
2020-11-30 00:12:32.780526 dummy1 < -0.01 99.88
2020-11-30 00:12:32.349853 dummy1 > +0.01 99.89
2020-11-30 00:12:31.633732 dummy1 << -0.1 99.88
2020-11-30 00:12:30.969339 dummy1 < -0.01 99.97999999999999
2020-11-30 00:12:30.560303 dummy1 < -0.01 99.99
2020-11-30 00:12:29.515647 dummy1 >> +0.1 100.0
2020-11-30 00:12:28.492602 dummy1 >> +0.1 99.9
2020-11-30 00:12:27.471624 dummy1 > +0.01 99.80000000000001
2020-11-30 00:12:26.301999 dummy1 << -0.1 99.79
Relative Step
0.01
<<
<
>
>>
Absolute Position
99.88
Go!
Stop!

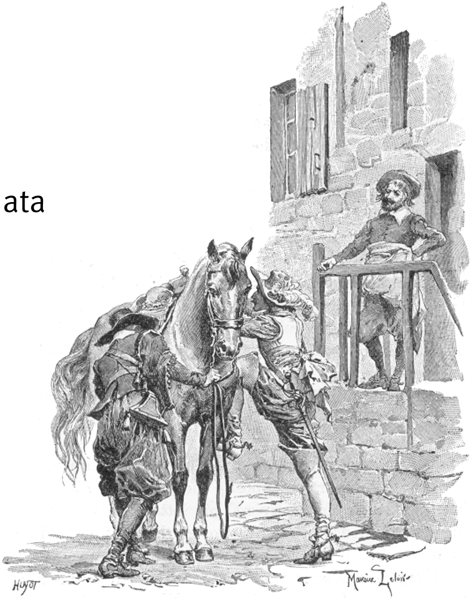
slic – GUI


Neat DAQ (on satese-cons-02.psi.ch)
Config
Static
Scan
Tweak
Athos Rep. Rate:
5.0 Hz
SF DAQ on http://sf-daq:10002 (status: idle, last run: None):
Detectors
BS Channels
PVs
Instrument
maloja
pgroup
p18722
Update!


BS Channels (on satese-cons-02.psi.ch)
10 channels online
SATES20-CVME-EVR0:DUMMY_PV1_NBS
SATES20-CVME-EVR0:DUMMY_PV2_NBS
SATES21-CAMS154-GIGE2:FPICTURE
SATFE10-PEPG046-EVR0:CALCI
SATFE10-PEPG046-EVR0:CALCS
SATFE10-PEPG046-EVR0:CALCT
SATFE10-PEPG046-EVR0:CALCX
SATFE10-PEPG046-EVR0:CALCY
1 channels offline
SATES20-CVME-EVR0:DUMMY_PV3_NBS
Close

Questions?

sfdata



FEL Data Analysis

- Needs for data analysis at FELs changing from experiment to experiment.
- Instead of providing ready-made solutions for a bespoke type of experiment, it is common practice to provide tools (for BL staff and users) to build an analysis quickly.
- Common setup used at all† SwissFEL instruments:
  - Jupyter (on Ra)
  - sfdata (internally using jungfrau_utils for JF data)

†OK, almost all...

PAUL SCHERRER INSTITUT PSI

sfdata – Motivation

Lower the bar as much as possible for users to analyze arbitrary data from rapidly changing endstations.

SwissFEL data is (historically and currently) written to several independent and slightly inconsistent hdf5 files per acquisition (`acq0123.*.h5`):

- BS scalars and waveforms → `*.BSDATA.h5`
- BS camera images → `*.CAMERAS.h5`
- for each Jungfrau detector → `*.JF*.h5`
- epics scalars and waveforms → `*.PVCHANNELS.h5`

plus a `json` file with scan metadata.

Missing Pulses!
FEL data analysis usually has to be shot-by-shot (due to inherent fluctuations)
→ Pulses missing from arbitrary sources have to dealt with correctly.

sfdata – Single Acquisition

sfdata instead of plain h5py → Hide complexity from the user:

```
from matplotlib import pyplot as plt
from sfdata import SFDataFiles

fns = "/sf/instrument/data/p12345/raw/run0001/data/acq0001.*.h5"
with SFDataFiles(fns) as data:
    subset = data["SIGNAL", "BACKGROUND"] # select channels
    subset.drop_missing()                 # make channels consistent
    pids = subset["SIGNAL"].pids          # read pulse IDs
    sig = subset["SIGNAL"].data           # read data
    bkg = subset["BACKGROUND"].data

norm = sig - bkg

plt.plot(pids, norm)
plt.show()
```

- Open **all** files from one acquisition,
- merge channels into **one** dict-like object.

sfdata – Scan

Similarly for scans:

```
from matplotlib import pyplot as plt
from sfdata import SFScanInfo

fn = "/sf/instrument/data/p12345/raw/run0001/meta/scan.json"
scan = SFScanInfo(fn)

xs = scan.readbacks
ys = np.empty_like(xs)

for i, step in enumerate(scan):
    # step is a SFDataFiles object

    subset = step["SIGNAL", "BACKGROUND"]
    subset.drop_missing()
    pids = subset["SIGNAL"].pids
    sig = subset["SIGNAL"].data
    bkg = subset["BACKGROUND"].data

    ys[i] = sig - bkg

plt.plot(xs, ys)
plt.show()
```

sfdata – Further Nice Things

```
f = SFDataFiles(fns)
ch = f["SIGNAL"]
```

Reading slices:

```
# read only a 100x100 ROI of the first 10 images
rois = ch[:10, 200:300, 400:500]
```

Reading in batches:

```
for indices, batch in ch.in_batches(n=3, size=100):
    for image in batch:
        do_something_with(image)
```

```
intensity = np.empty(ch.nvalid)
for indices, batch in ch.in_batches():
    intensity[indices] = batch.sum(axis=(1, 2))
```

```
def proc(batch):
    return batch.sum(axis=(1, 2))

intensity = ch.apply_in_batches(proc)
```

sfdata – Further Nice Things

- For **valid** data: `len(ch)`, `ch.shape`, `ch.ndim`, `ch.size`
- Timing offsets:

```
subset = data["SIGNAL", "BACKGROUND"]
ch_sig = subset["SIGNAL"]
ch_bkg = subset["BACKGROUND"]

ch_bkg.offset = 1      # channel is delayed by one pid
subset.drop_missing() # takes offset into account
```

- Built-in conversion to (e.g., for imputation):
  - Pandas DataFrames
  - xarrays
- Statistics (also as command-line tool):

```
--------------------------------------------------------------------------------

SAR-CVME-TIFALL5:EvtSet                      102 / 102 ->  0% loss
SARES11-SPEC125-M1.roi_background_x_profile   45 / 102 -> 56% loss
SLAAR11-LTIM01-EVR0:DUMMY_PV1_NBS            101 / 102 ->  1% loss
SLAAR11-LTIM01-EVR0:DUMMY_PV2_NBS            102 / 102 ->  0% loss

over the whole data set: 45 / 102 -> 56% loss

--------------------------------------------------------------------------------
```

PAUL SCHERRER INSTITUT

sfdata – Outlook

Currently investigated idea: **Move away from full file names.**

```
load = make_loader(instrument="alvra", pgroup="p12345")
```

Open a single run:

```
run = load(run=10)
ch = run["SIGNAL"]
...
```

Loop over several runs:

```
runs = load(run=range(10))
for run in runs:
    ch = run["SIGNAL"]
    ...
```

Overwrite default parameters:

```
run = load(pgroup="p23456", run=10)
```

Allowing wildcards: `pgroup="p12*"`, etc.
and alternative spellings: `run=1`, `run="01"`, `run="run1"`, etc.

Thank you for your attention!