EUROPEAN SPALLATION SOURCE

An introduction to NICOS

PRESENTED BY MATT CLARKE

Background

- Developed at FRM II in Munich
  - Started around 2002, developed for specific beamlines
  - Modernisation project started in 2009
  - Decision made to use it on all beamlines (2013)
- ESS and SINQ joined the collaboration ~2016
  - Survey of existing control systems such as IBEX, SICS, Bluesky, Sardana, …
- Deployed at V20 for ESS tests
  - Choppers, SANS, …
- Deployed at Utgård for integrating SE
- YMIR
  - Motion, scanning, light tomo

SINQ: deployment at AMOR

Facilities

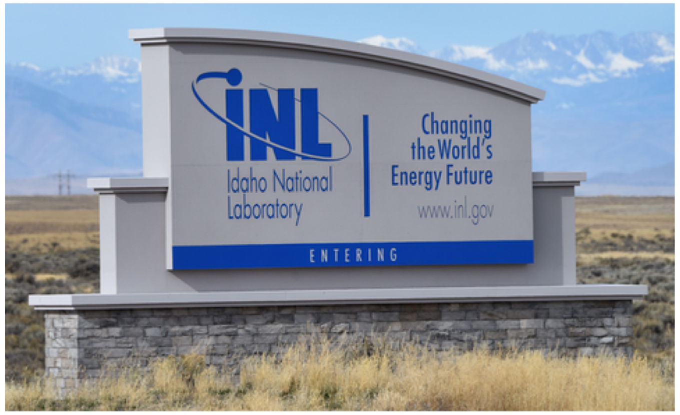
ess

PETERSBURG NUCLEAR PHYSICS INSTITUTE

batan National Nuclear Energy Agency of Indonesia

MGML

LAHN LABORATORIO ARGENTINO DE HACES DE NEUTRONES



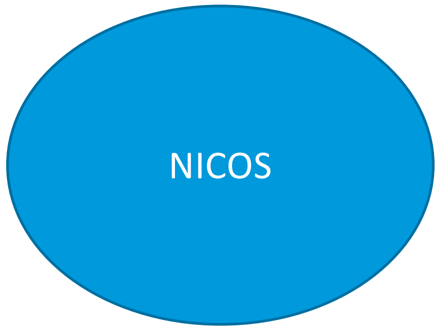
TU WIEN TECHNISCHE UNIVERSITÄT WIEN Vienna University of Technology

ISIS Neutron and Muon Source

Architecture

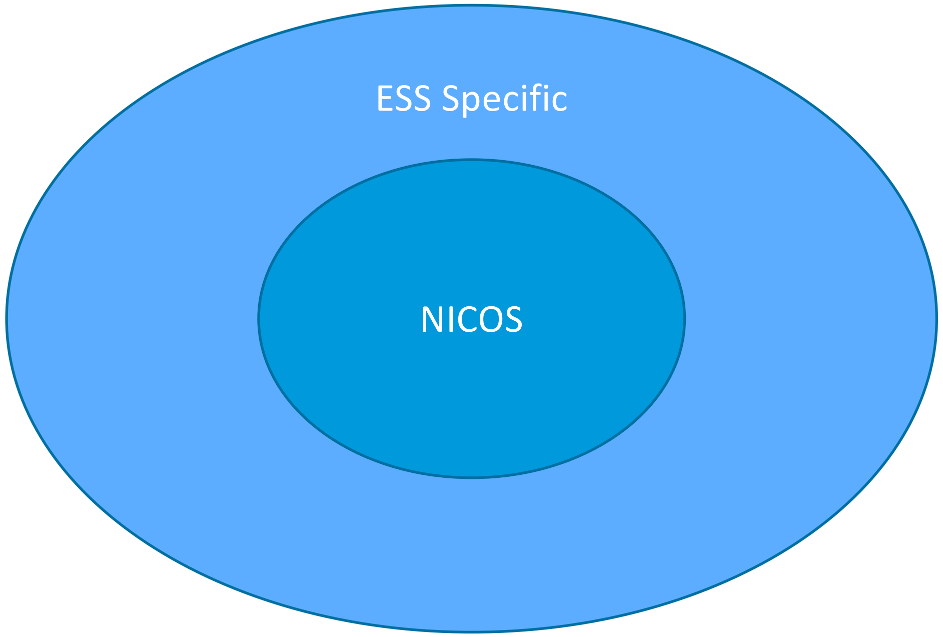
- Poller
  - Responsible for getting the latest values from devices
  - Puts the values in the cache
- Cache
  - Stores device values
  - Supplies latest values to clients
- Daemon
  - The brain
  - Runs commands and scripts
  - Determines what devices are loaded
  - Forwards requests to devices

Core Functionality



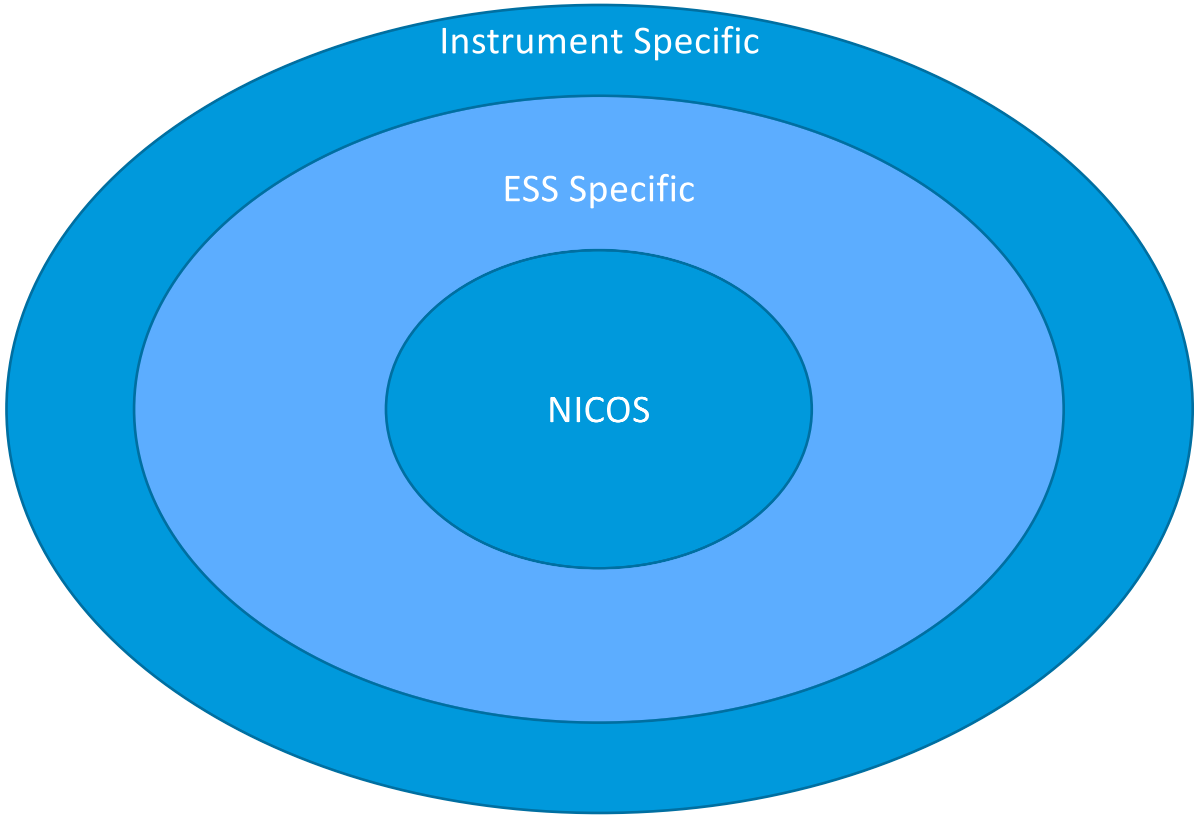
- Poller
- Cache
- Daemon
- Architecture
- Device hierarchy
- …

ESS Specific



- EPICS support
  - Monitors
- Data files
  - Kafka file-writer
- ESS proposal system
- flowui
- Live display
- Graylog
- Common devices
- …

Instrument Specific

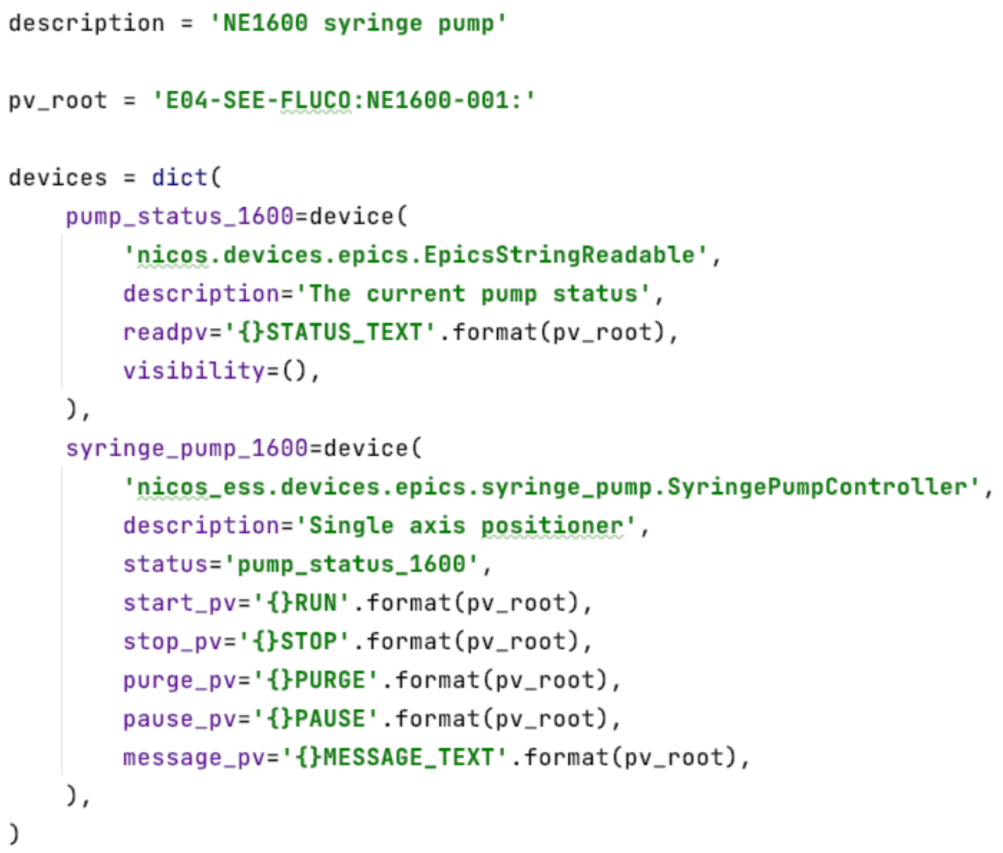


- Special devices
- Custom commands
- Scripts
- GUI widgets
- …

Devices

- NICOS aims to make devices easy to add
  - Rules to be followed
- Devices is a broad term
  - A device that reads a single PV
  - A device that reads and writes a PV
  - An umbrella device that combines multiple devices
  - A completely custom device

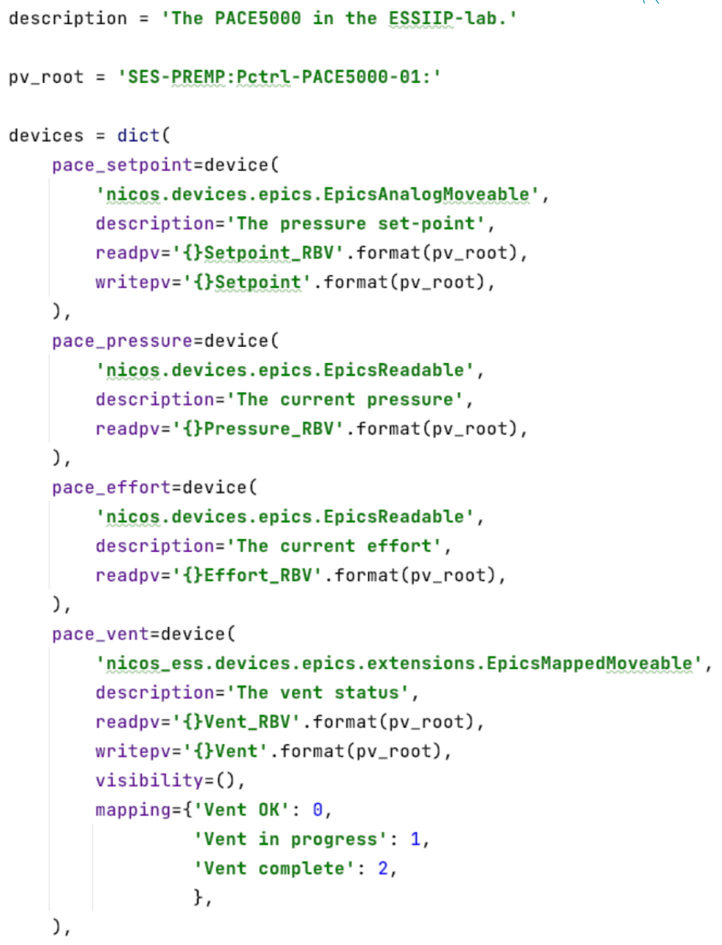
Setup files



```
description = 'NE1600 syringe pump'

pv_root = 'E04-SEE-FLUCO:NE1600-001:'

devices = dict(
    pump_status_1600=device(
        'nicos.devices.epics.EpicsStringReadable',
        description='The current pump status',
        readpv='{}STATUS_TEXT'.format(pv_root),
        visibility=(),
    ),
    syringe_pump_1600=device(
        'nicos_ess.devices.epics.syringe_pump.SyringePumpController',
        description='Single axis positioner',
        status='pump_status_1600',
        start_pv='{}RUN'.format(pv_root),
        stop_pv='{}STOP'.format(pv_root),
        purge_pv='{}PURGE'.format(pv_root),
        pause_pv='{}PAUSE'.format(pv_root),
        message_pv='{}MESSAGE_TEXT'.format(pv_root),
    ),
)
```



```
description = 'The PACE5000 in the ESSIIP-lab.'

pv_root = 'SES-PREMP:Pctrl-PACE5000-01:'

devices = dict(
    pace_setpoint=device(
        'nicos.devices.epics.EpicsAnalogMoveable',
        description='The pressure set-point',
        readpv='{}Setpoint_RBV'.format(pv_root),
        writepv='{}Setpoint'.format(pv_root),
    ),
    pace_pressure=device(
        'nicos.devices.epics.EpicsReadable',
        description='The current pressure',
        readpv='{}Pressure_RBV'.format(pv_root),
    ),
    pace_effort=device(
        'nicos.devices.epics.EpicsReadable',
        description='The current effort',
        readpv='{}Effort_RBV'.format(pv_root),
    ),
    pace_vent=device(
        'nicos_ess.devices.epics.extensions.EpicsMappedMoveable',
        description='The vent status',
        readpv='{}Vent_RBV'.format(pv_root),
        writepv='{}Vent'.format(pv_root),
        visibility=(),
        mapping={'Vent OK': 0,
                 'Vent in progress': 1,
                 'Vent complete': 2,
                 },
    ),
```

Standard Commands

- Lots of commands
  - Not all relevant to us (will be removed/hidden)
  - Can add our own or modify existing ones
- Examples
  - move => change a device’s value
  - maw => move and wait
  - waitfor => wait for condition
  - scan => scan a device
  - help()

Custom Commands

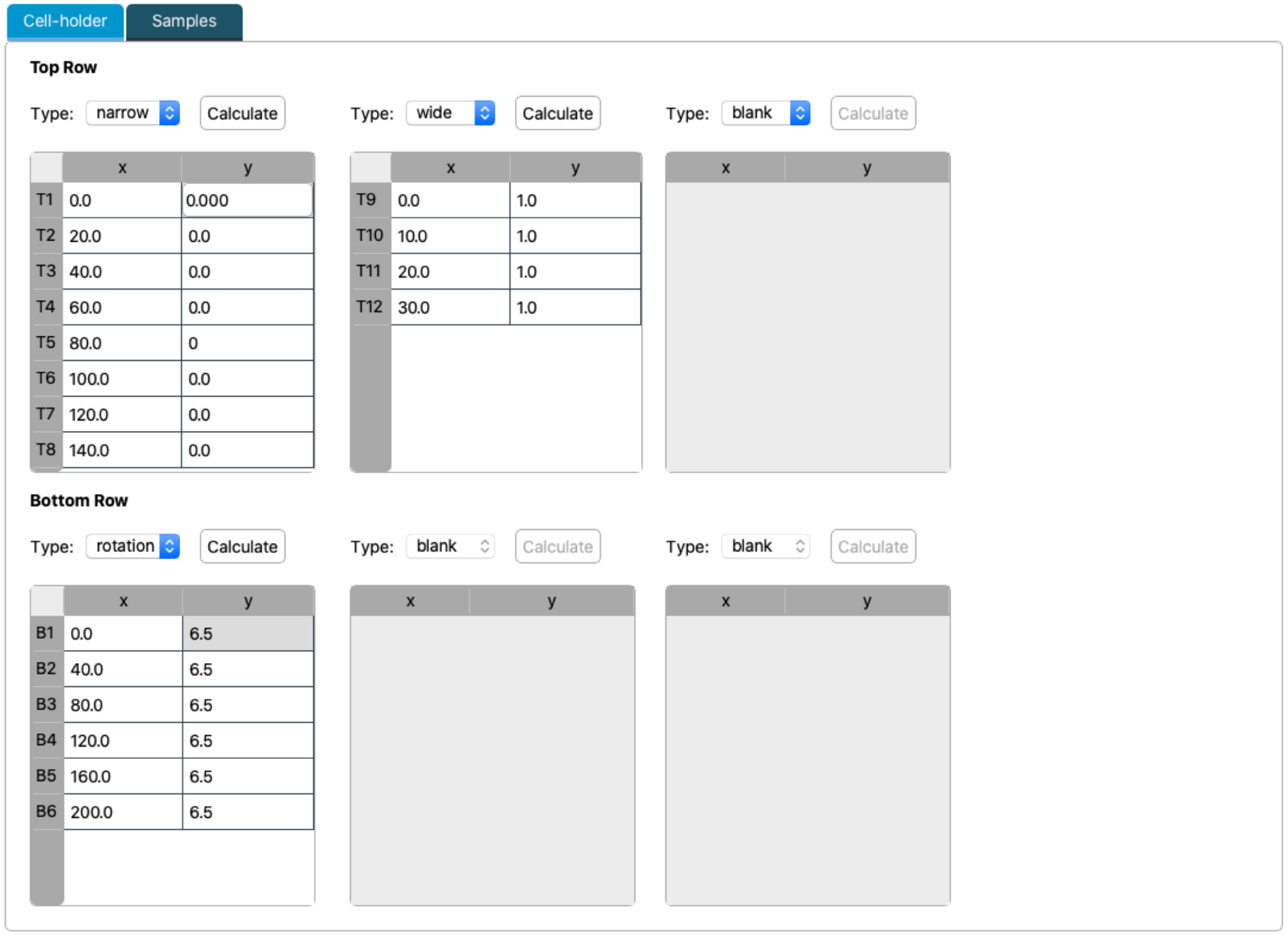
```python
import random

from nicos import session
from nicos.commands import usercommand
from nicos.commands.standard import move


@usercommand
def set_wavelength(value):
    """Set the chopper to the specified wavelength"""
    session.log.warn(f'setting chopper to {value}nm')
    #  TODO: insert proper calculation here
    move('ch1_phase', random.randint(0, 180))
    move('ch1_speed', min(value // 2, 300))
```

It’s just Python!

Custom GUIs



Cell-holder | Samples

Top Row

Type: narrow | Calculate

	x	y
T1	0.0	0.000
T2	20.0	0.0
T3	40.0	0.0
T4	60.0	0.0
T5	80.0	0
T6	100.0	0.0
T7	120.0	0.0
T8	140.0	0.0

Type: wide | Calculate

	x	y
T9	0.0	1.0
T10	10.0	1.0
T11	20.0	1.0
T12	30.0	1.0

Type: blank | Calculate

x	y

Bottom Row

Type: rotation | Calculate

	x	y
B1	0.0	6.5
B2	40.0	6.5
B3	80.0	6.5
B4	120.0	6.5
B5	160.0	6.5
B6	200.0	6.5

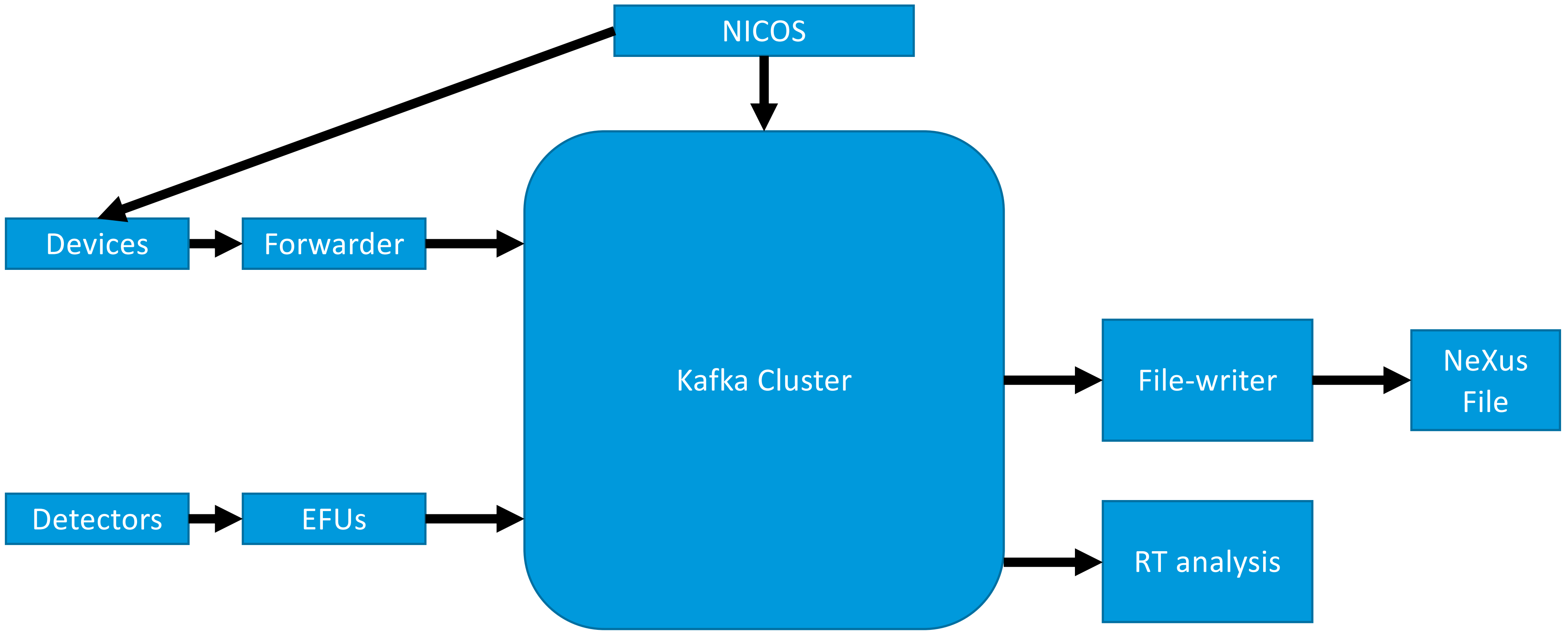
Type: blank | Calculate

x	y

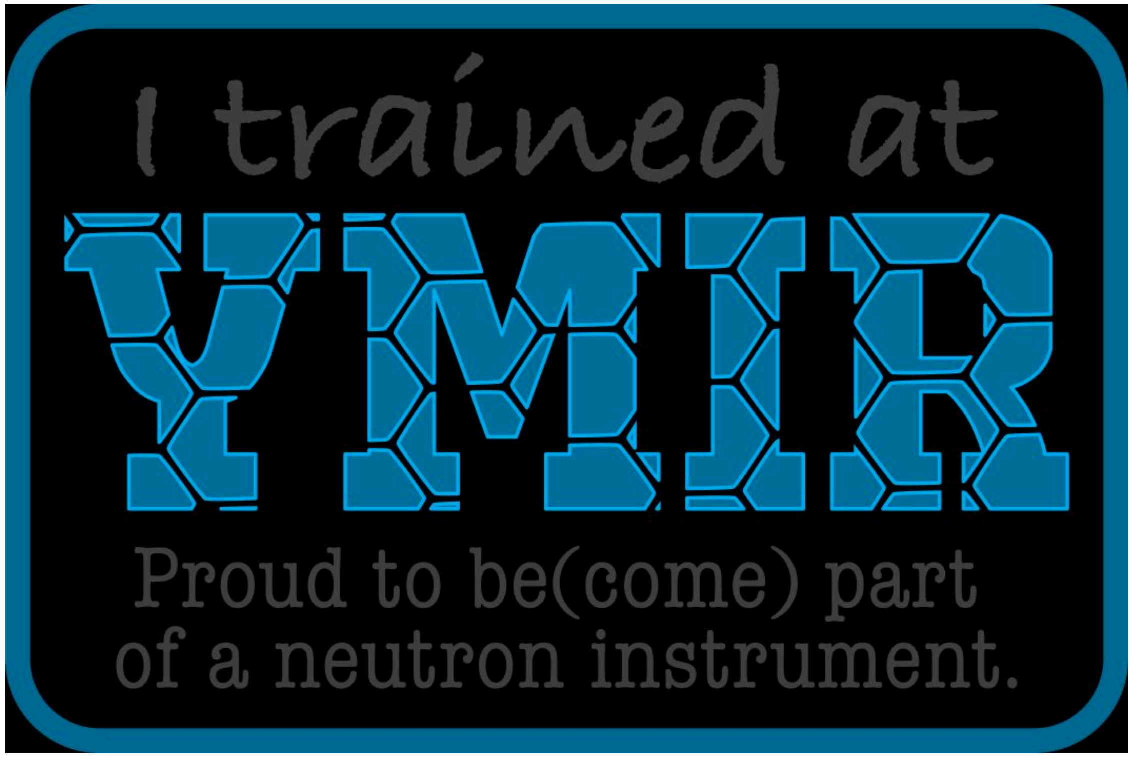
Type: blank | Calculate

x	y

Experiment Control



Demo


I trained at
YMIR
Proud to be(come) part
of a neutron instrument.

Tasks

- Use the help command
- Load the chopper, detector and motion stages
- Move some motors
- Use the count command to count for 30 seconds
- Use the count command to count for 20,000 events on det_image1
- Scan mx coarsely to find rough centre (hint: help(scan))
- Scan mx more finely around centre to find a more precise value
- Break out your Python skills! Write a script that manually does a 2D scan:
  - for my = [10, 20, 30] and mx = [20, 30, 50, 70, 80]
  - and at each point:
    - prints out “moved to mx=? my=?” (replace ? with real values)
    - counts for 5 seconds