NeXus Files from Tango and Sardana

PANDATA WP5 / HDRI

Jan Kotański, Eugen Wintersberger

Deutsches Elektronen-Synchrotron

May 7, 2014

Outline

- NeXus/HDF5 files

  HDRI/Pandata projects
- NeXus Writer Configuration

  Components with DataSources and Strategy
- User Interfaces
  - Sardana + GUI

    NeXus Recorder
  - Sardana + Macros

    Beamline specific macros
  - Python scripts

NeXus/HDF5 files

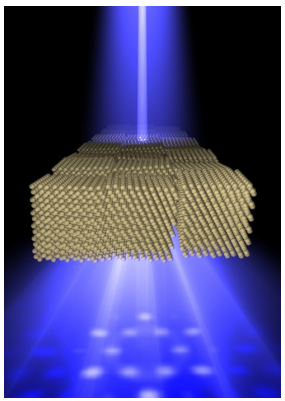
HDRI/Pandata projects

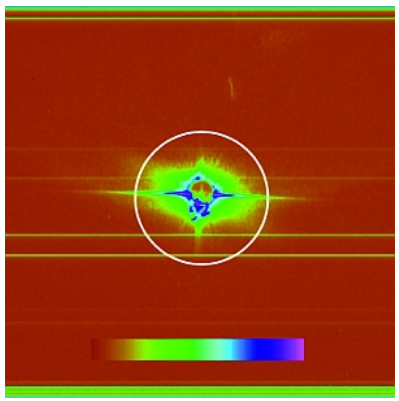
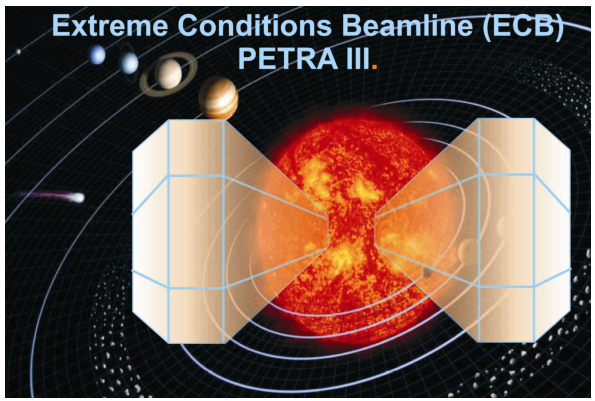
NeXus/HDF5 files

Why NeXus ?

- store all data in one container – NeXus file
- full description of experiment, metadata catalogs
  searching via keywords, i.e. γ-Portal
- data provenance – sufficient details to allow reproducibility
  managing, sharing, and reusing data
- application definitions for specific experiment types
  common data structure in photon-science facilities

This way data can be managed efficiently



NeXus/HDF5 files

NeXus Files as a file system

They contain four types of entity: groups, fields, attributes, and links

- group – folder with a defined NeXus type; building block of the NeXus structure
- field – data-set with a name and a NeXus path related to its physical meaning
- attribute – extra information required to describe a particular group or field
- link – used to reference the plottable data from NXdata

NeXus/HDF5 files

NeXus Files as a file system

They contain four types of entity: groups, fields, attributes, and links

- group – folder with a defined NeXus type; building block of the NeXus structure
- field – data-set with a name and a NeXus path related to its physical meaning
- attribute – extra information required to describe a particular group or field
- link – used to reference the plottable data from NXdata

NeXus/HDF5 files

NeXus Files as a file system

They contain four types of entity: groups, fields, attributes, and links

- group – folder with a defined NeXus type; building block of the NeXus structure
- field – data-set with a name and a NeXus path related to its physical meaning
- attribute – extra information required to describe a particular group or field
- link – used to reference the plottable data from NXdata

NeXus/HDF5 files

NeXus Files as a file system

They contain four types of entity: groups, fields, attributes, and links

- group – folder with a defined NeXus type; building block of the NeXus structure
- field – data-set with a name and a NeXus path related to its physical meaning
- attribute – extra information required to describe a particular group or field
- link – used to reference the plottable data from NXdata

PNI C++ libraries

Simply creation of NeXus files in C++ (HDRI project)

- `libpnicore` provides
  - types of well defined size
  - templates for buffers and arrays
  - reader code to import data from proprietary formats
- `libpniio` classes
  - write NeXus files using HDF5 as its storage back-end
  - make the development independent of the NeXus API
- `python-pniio` Python bindings via the Python package

All libraries are actually in use to develop the software required to establish NeXus as a data format at DESY

`http://code.google.com/p/pni-libraries/`

`http://www.pni-hdri.de`

NeXus Writer Configuration

Components, DataSources and Strategy

Configuration Components

To store the data we need to know:

- Where? – NeXus path with physical meaning
- When and How? – writing strategy: INIT, STEP, FINAL, POSTRUN
- What? – DataSources: CLIENT, TANGO, DB, Python scripts

Configuration Components:

- XML Strings in the NXDL format extended by strategy and datasources tags
- correspond to one or more devices and like devices can be switch on/off
- set positions of stored data in the NeXus tree

Configuration Components

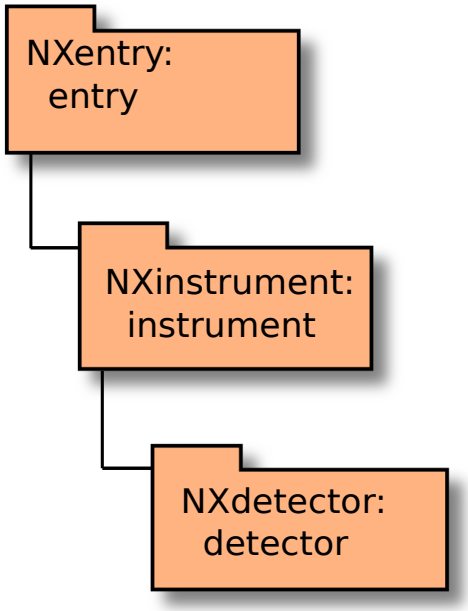
To store the data we need to know:

- Where? – NeXus path with physical meaning
- When and How? – writing strategy: INIT, STEP, FINAL, POSTRUN
- What? – DataSources: CLIENT, TANGO, DB, Python scripts

Configuration Components:

- XML Strings in the NXDL format extended by strategy and datasources tags
- correspond to one or more devices and like devices can be switch on/off
- set positions of stored data in the NeXus tree

Components and their merging

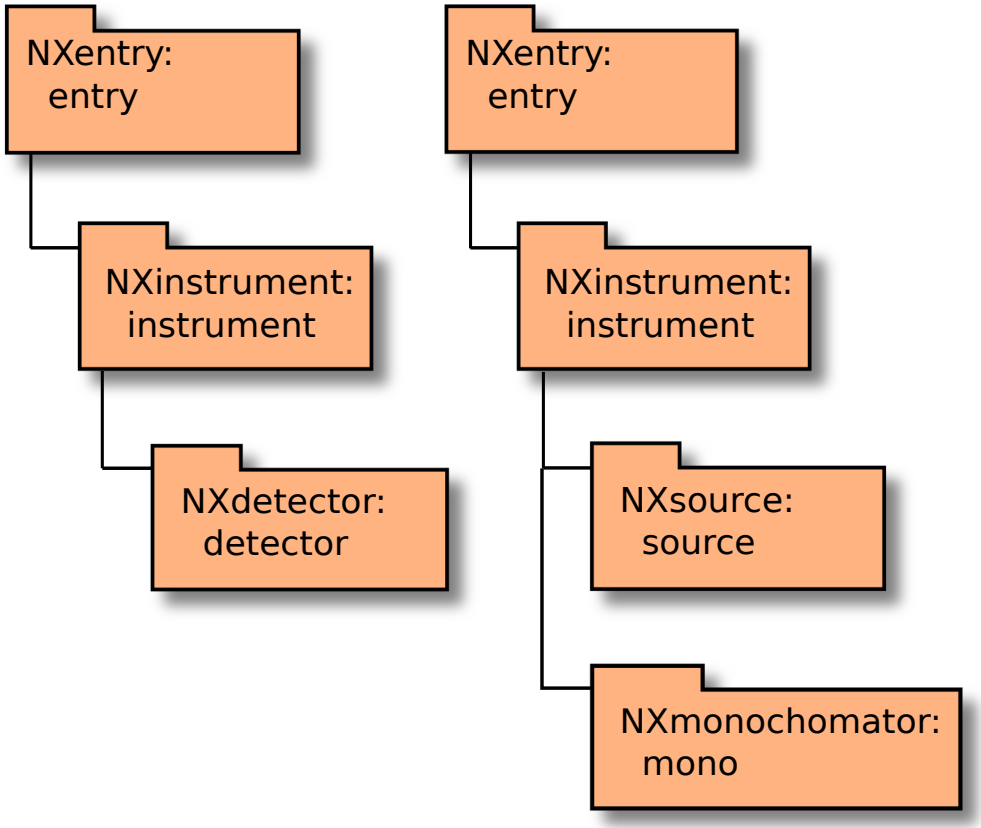


Detector Component

Default Component

Merged Components

Components and their merging

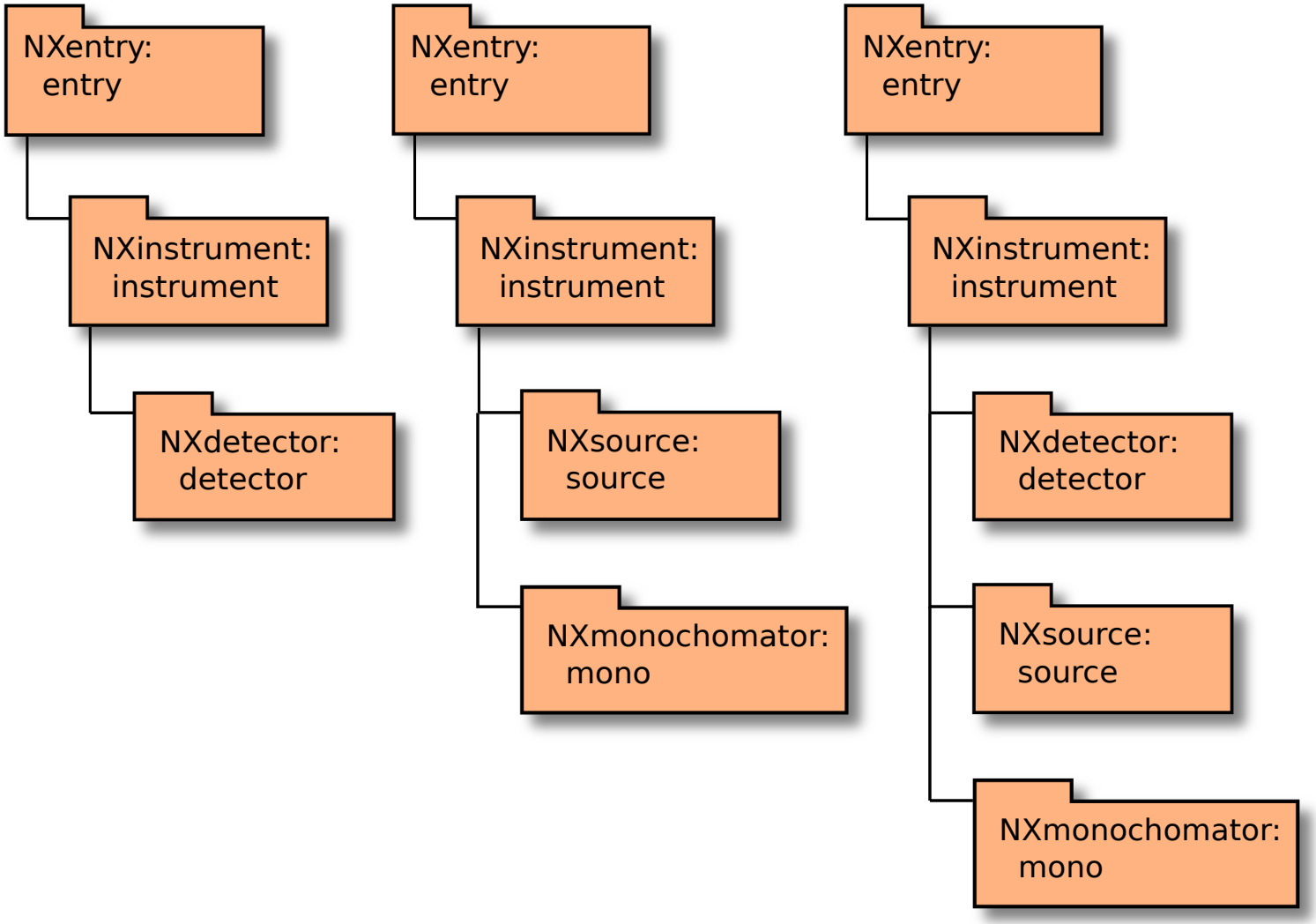


Detector Component

Default Component

Merged Components

Components and their merging

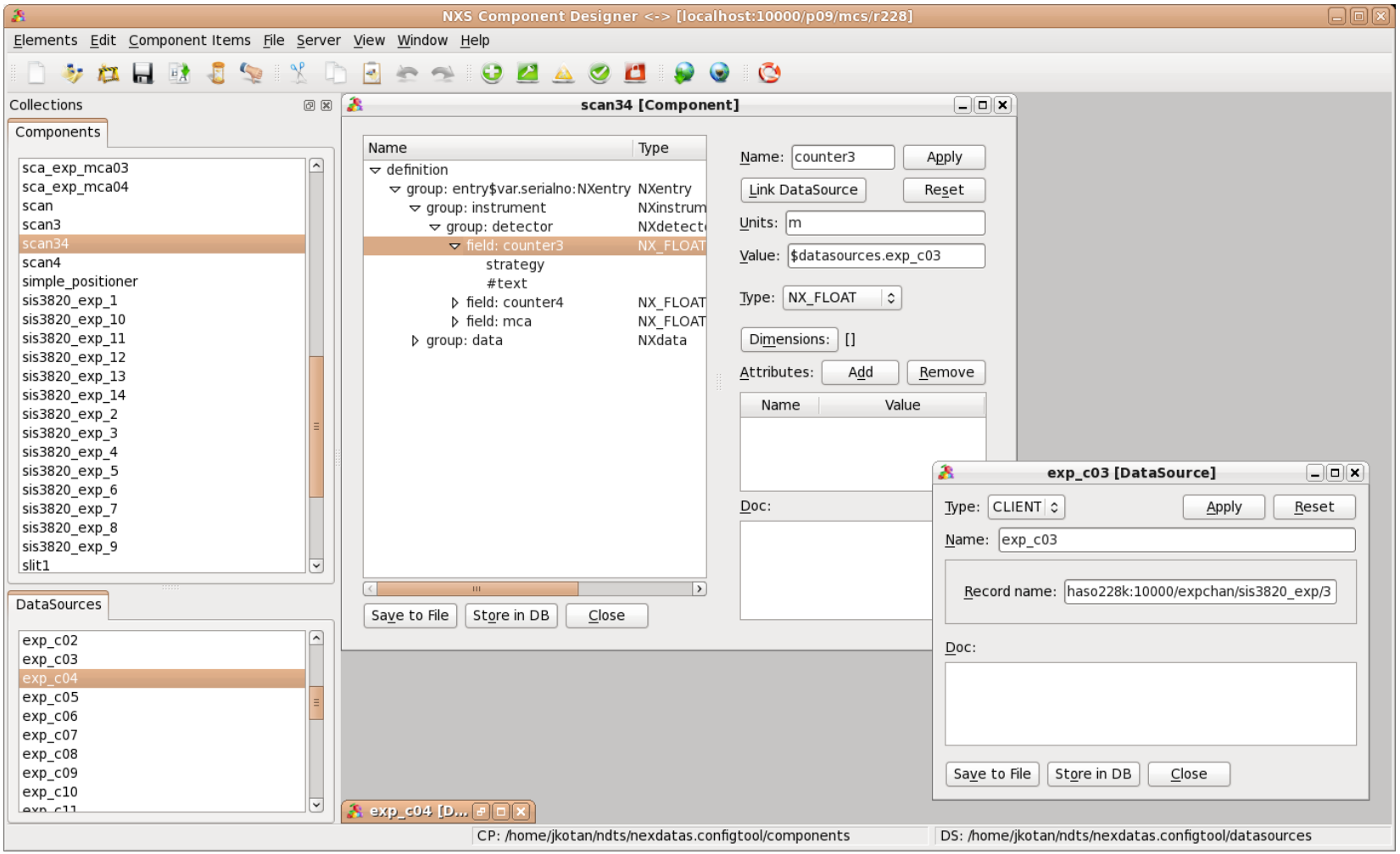


Detector Component

Default Component

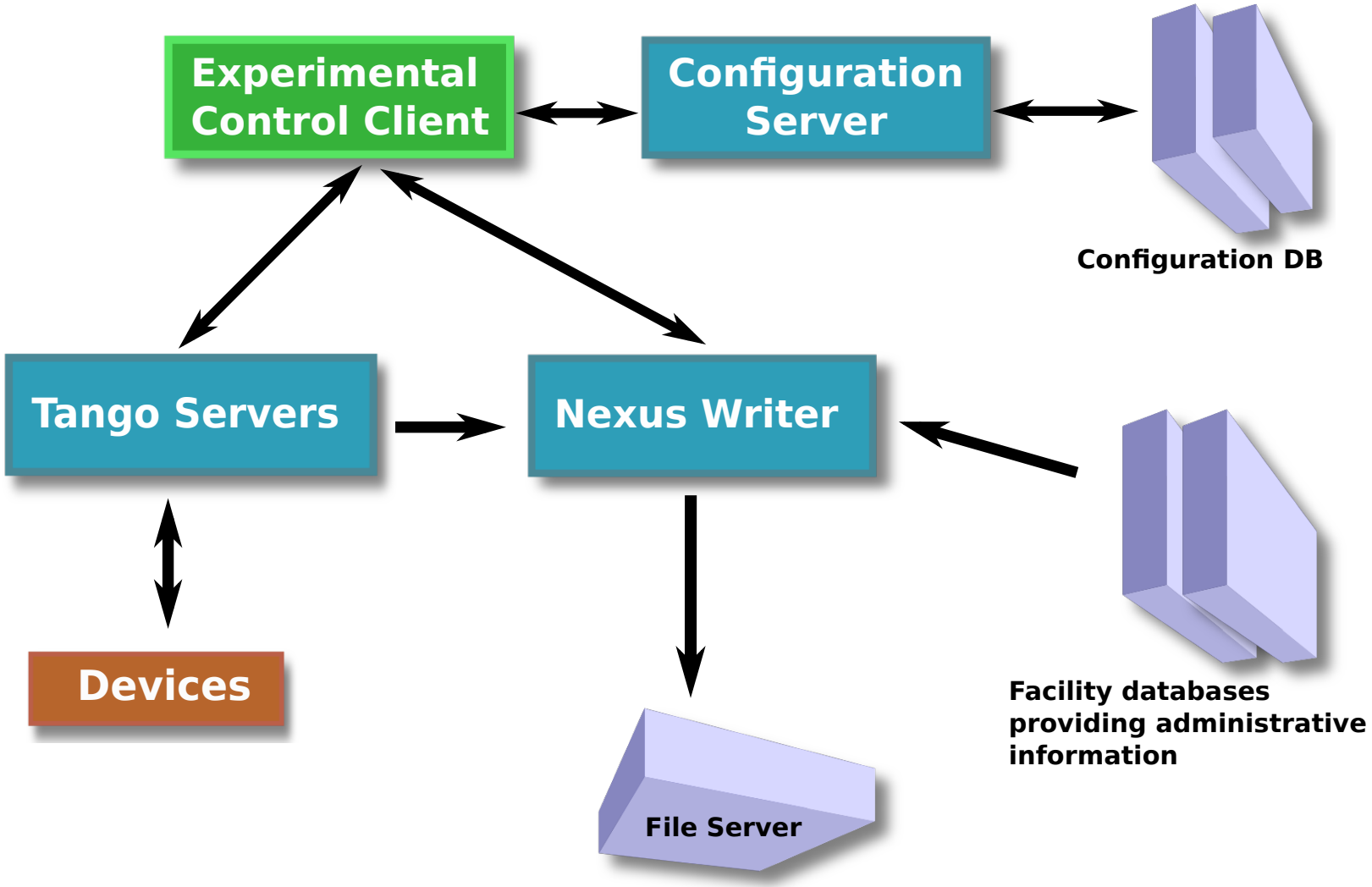
Merged Components

Component Designer



The Configuration Client Tool allows to create configuration components as well as datasources (for FS-EC use)

NeXus Writer

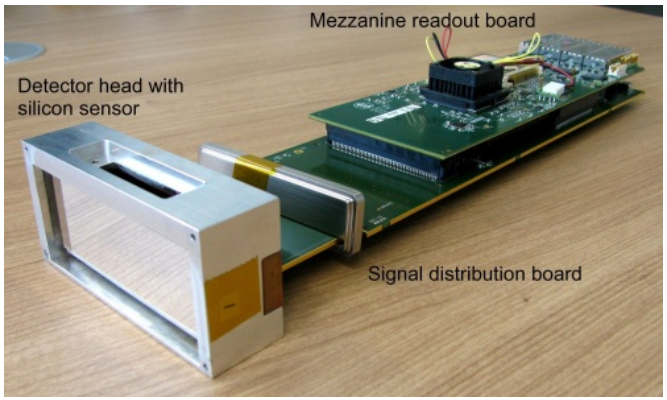


http://code.google.com/p/nexdatas/

Data Collector

For fast detector data is:

- saved locally by Detector Tango Server,
  e.g. lambda detector in NeXus files
- merged with data from NeXus Writer
  after performed experiment
- merging will be done by Data Collector (under discussion)



User Interfaces

GUI, Macros and Python

User Interfaces

- Sardana + GUI
  NeXus Component Selector
  NeXus Recorder in Sardana
- Sardana + Macros
  Beamline specific macros
  - NeXus Writer and Configuration Server used directly
  - examples:
    - Continuous scans with external trigger (Zebra & XIA & MCS) [P21]
    - Sweep Scans [P02]
- Python scripts

...

User Interfaces

- Sardana + GUI
  NeXus Component Selector
  NeXus Recorder in Sardana
- Sardana + Macros
  Beamline specific macros
  - NeXus Writer and Configuration Server used directly
  - examples:
    - Continuous scans with external trigger (Zebra & XIA & MCS) (P06)
    - Sweep Scans (P02)
- Python scripts

...

User Interfaces

- Sardana + GUI
  NeXus Component Selector
  NeXus Recorder in Sardana
- Sardana + Macros
  Beamline specific macros
  - NeXus Writer and Configuration Server used directly
  - examples:
    - Continuous scans with external trigger (Zebra & XIA & MCS) (P06)
    - Sweep Scans (P02)
- Python scripts

...

Client code example

A simple control client in Python:

```
from PyTango import DeviceProxy
dpx = DeviceProxy("p09/nxsdatawriter/01")
dpx.Init()
dpx.FileName = "/tmp/my_scan.nxs"
dpx.OpenFile()

ncs = DeviceProxy("p09/nxsconfigserver/01")
# create configuration from components
ncs.CreateConfiguration( ["ten_channnel_detector",
                          "slits", "beamstop"])
# send XMLString for a new scan
dpx.XMLSettings = ncs.XMLString

dpx.JSONRecord = '{"data": {"parameterA":0.2}}'
dpx.OpenEntry()
# ...
```

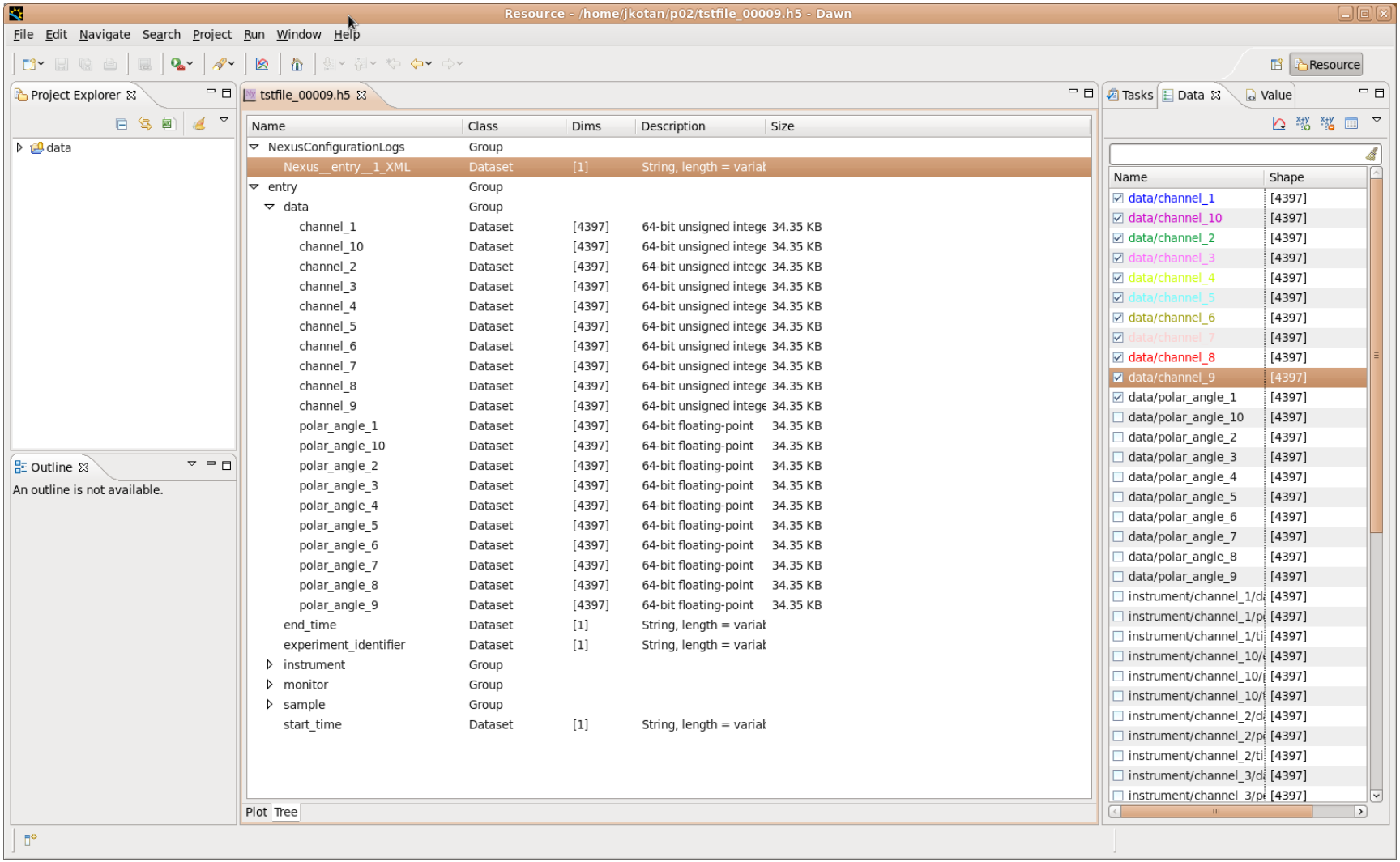
Client code example

```
# experiment main loop
# (write data with strategy STEP)
for i in range(100):
    dpx.Record(
        '{"data": {"exp_c01":%s,"exp_c02":%s}}' % \
            (i*0.1, i*1.2))

# write final data (strategy FINAL)
# - close the entry - close the file
dpx.JSONRecord = '{"data": {"parameterB":0.3}}'
dpx.CloseEntry()
dpx.CloseFile()
```

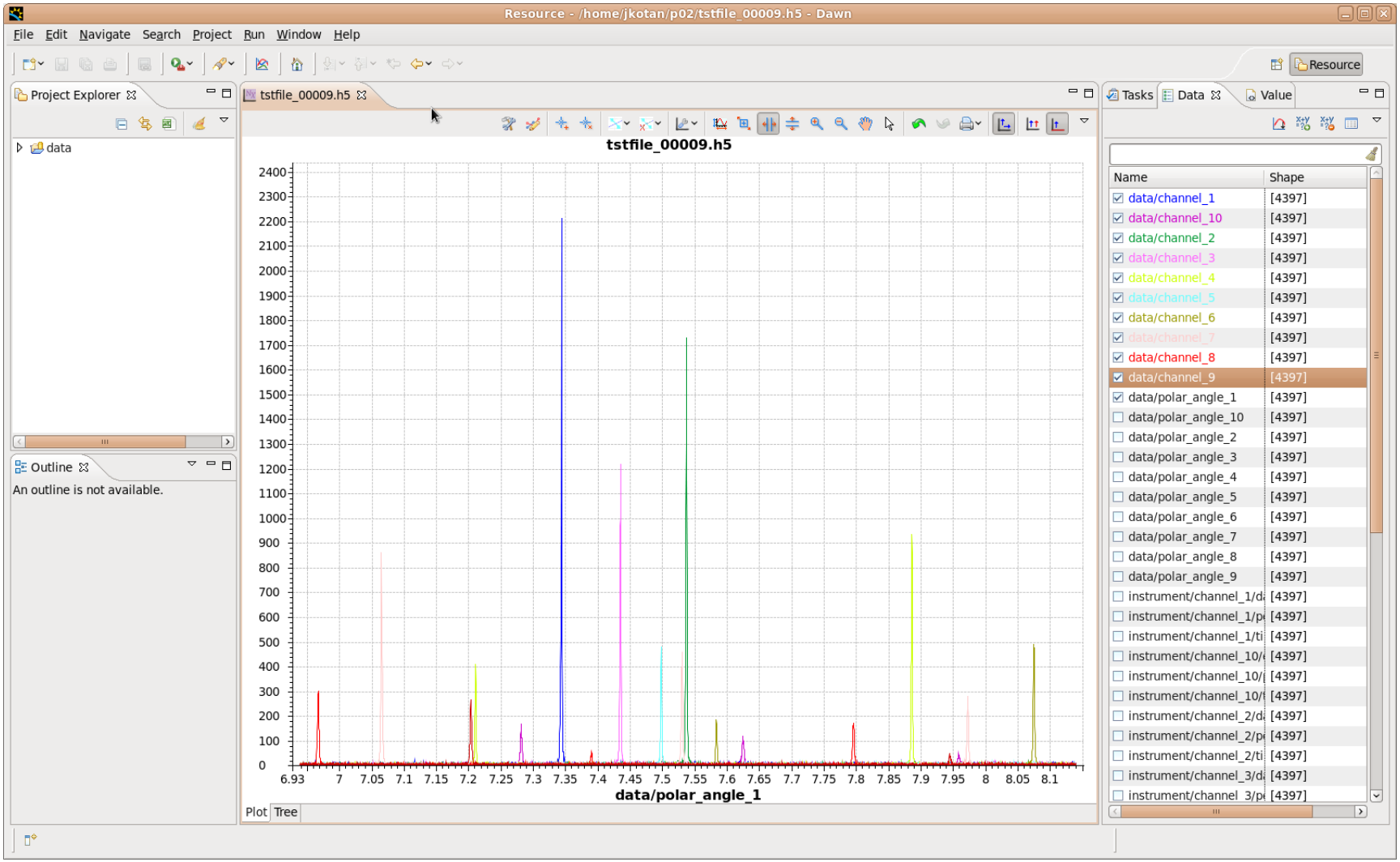
First data on P02.1 viewed in Dawn

High Resolution Powder Diffraction Beamline: `p02sweep.py` script



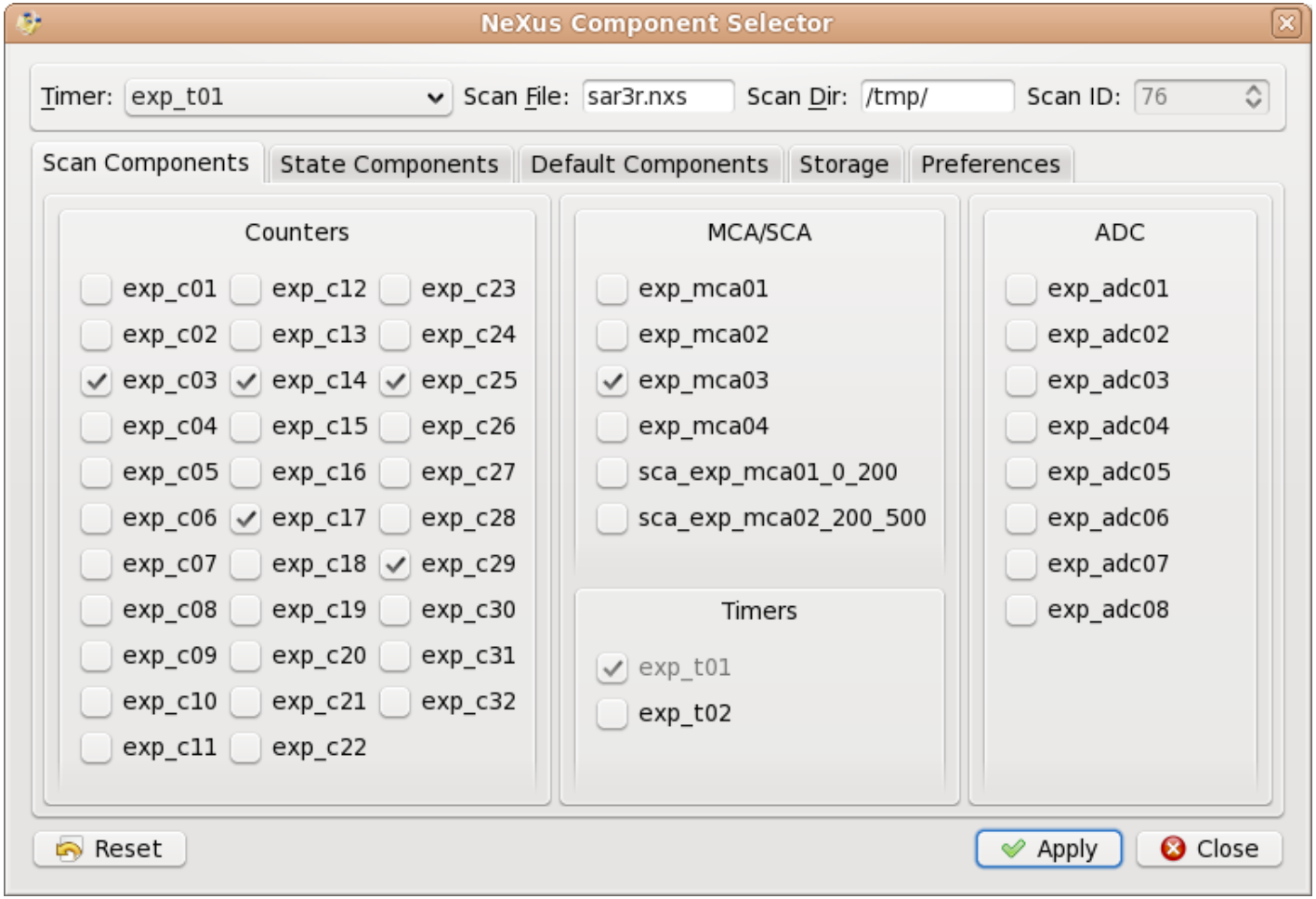
First data on P02.1 viewed in Dawn

High Resolution Powder Diffraction Beamline: `p02sweep.py` script



NeXus Component Selector

Device Selection Editor/View

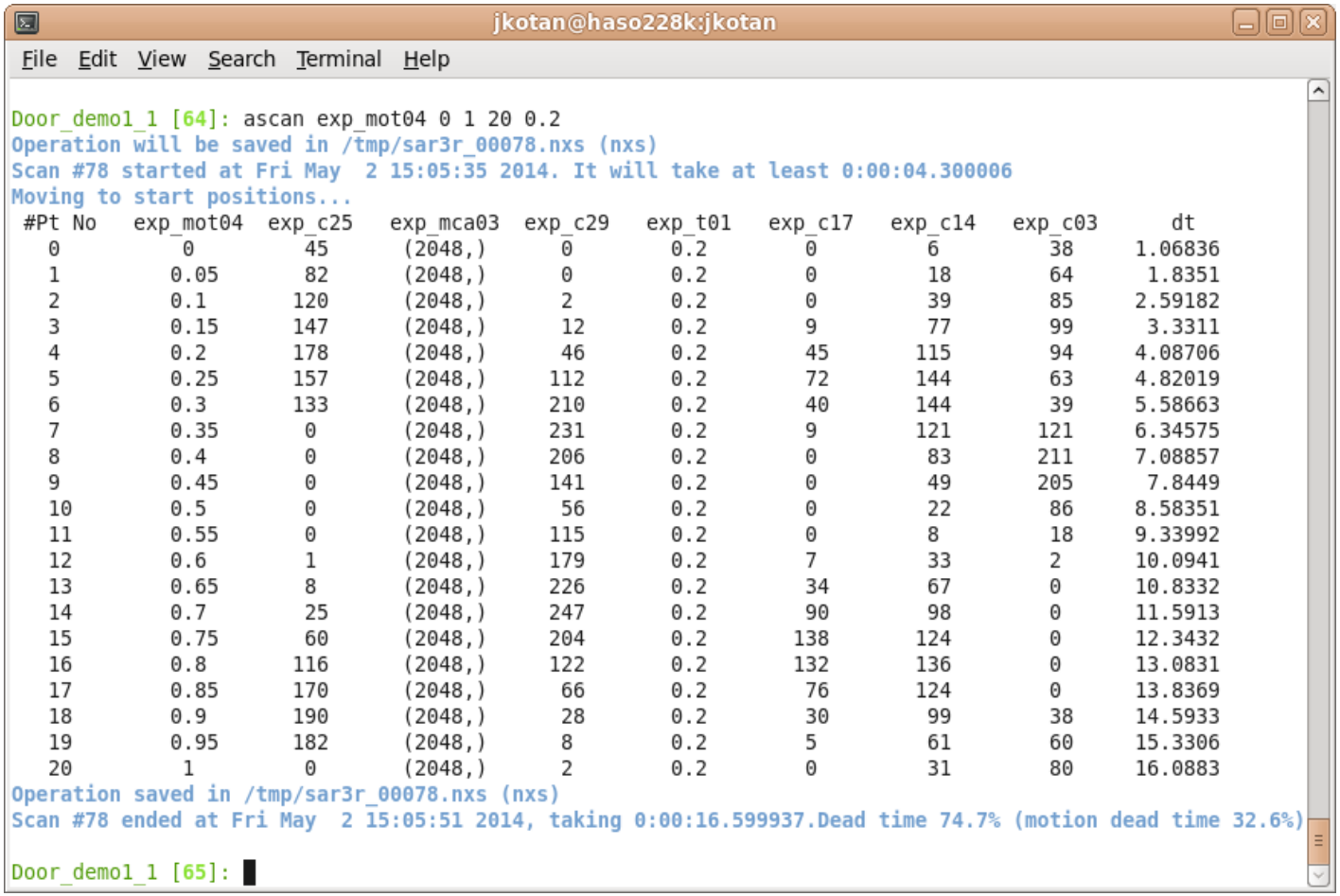


Now everybody can use Sardana!

Experimental Channels from Sardana Pool

Spock

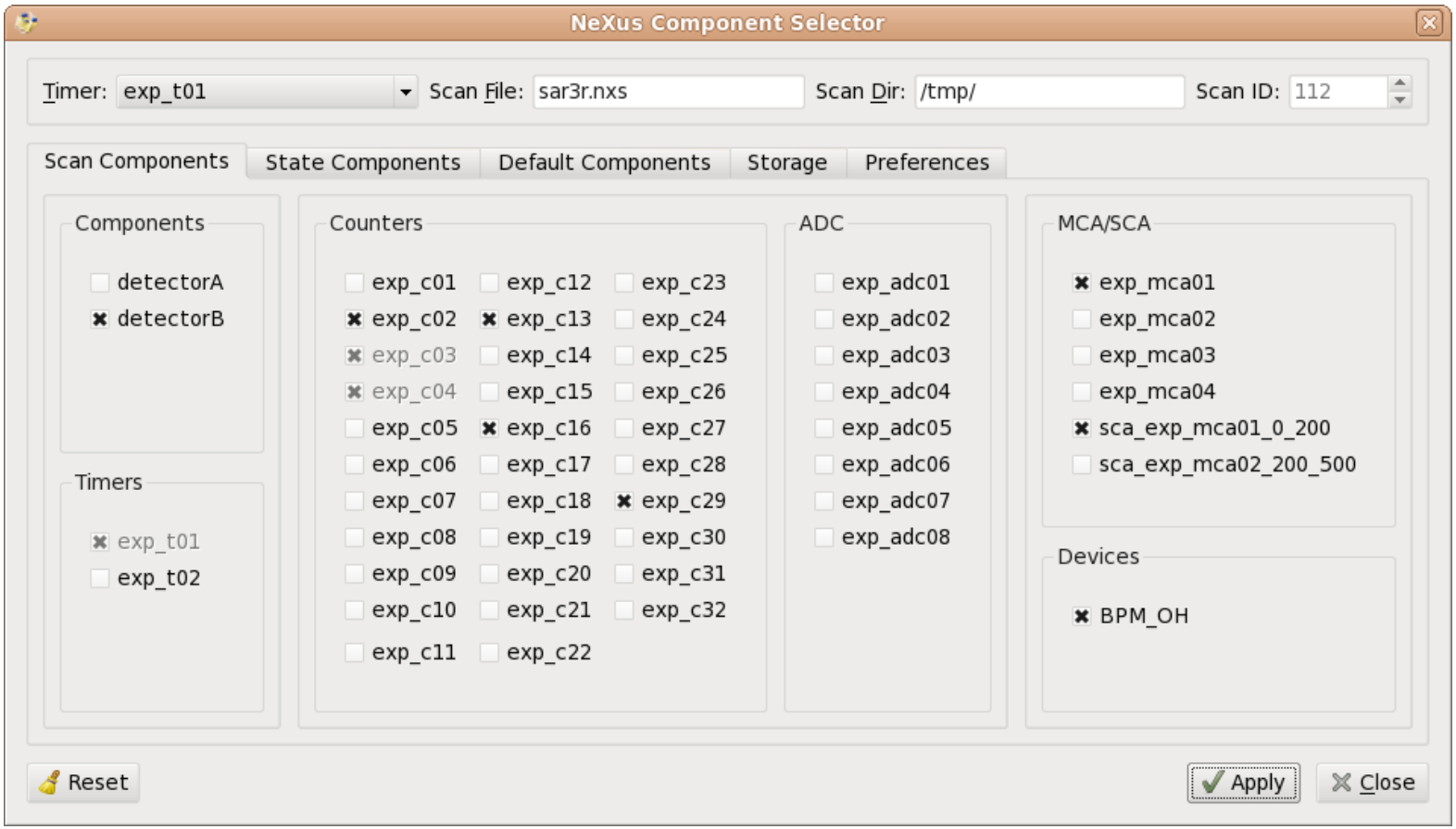
User scan in spock with the `exp_mot04` motor.



For NeXus the file extension is `.nxs`

NeXus Component Selector

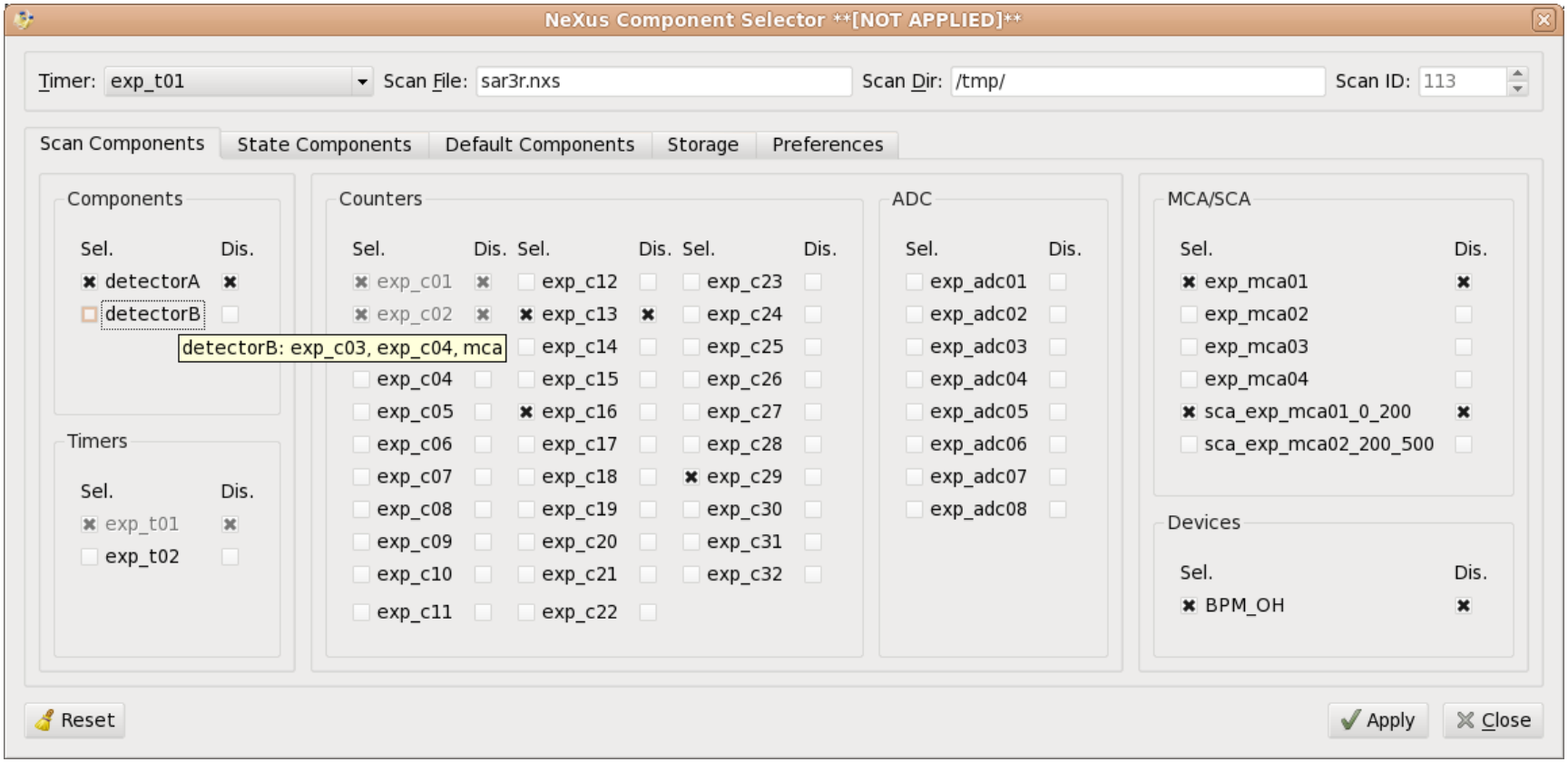
Scan Components



Selecting components containing other devices.

NeXus Component Selector

Scan Components

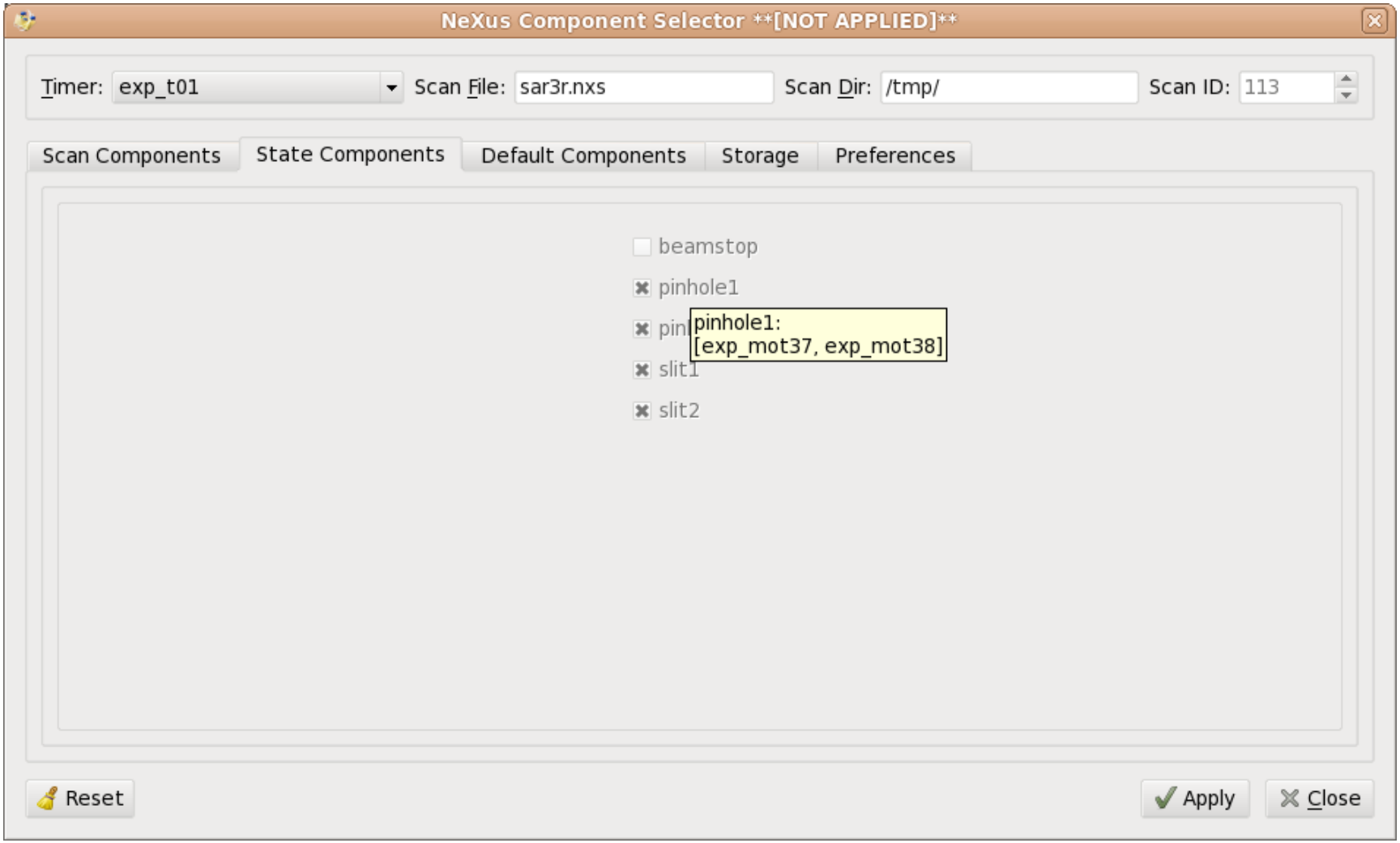


One can also disable display for Taurus.

Searching for the best user interface.

NeXus Component Selector

State Components

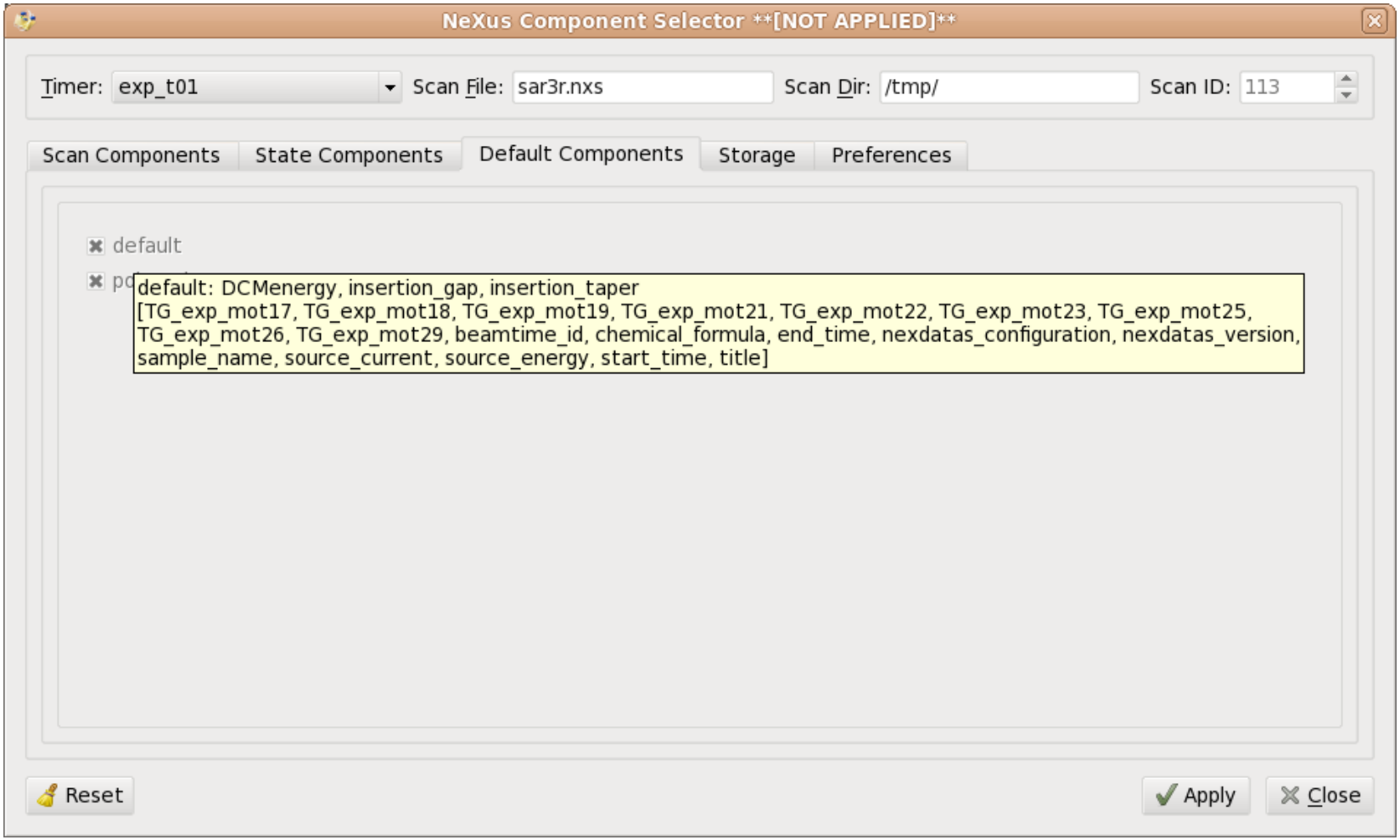


State components describe experimental setup.

They are automatically deselected if related to them motors are switch off.

NeXus Component Selector

Default Components



Default components describe mandatory data,

e.g. title, sample_name, beamtime_id, start_time, end_time,...

Spock

User scan in spock with the `exp_mot04` motor.

jkotan@haso228k:jkotan

File Edit View Search Terminal Help

```
Door_demo1_1 [7]: ascan exp_mot04 0 1 20 0.2
Operation will be saved in /tmp/sar3r.nxs (nxs)
Scan #114 started at Wed May  7 09:21:53 2014. It will take at least 0:00:04.300006
Moving to start positions...
```

#Pt No	exp_mot04	exp_mca01	exp_c29	exp_t01	exp_c16	sca_exp_mca01_0_200	exp_c13	exp_c01	exp_c02	exp_c03	dt
0	0	(2048,)	0	0.2	3	3064	6	208	84	24	1.61741
1	0.05	(2048,)	0	0.2	0	2	15	166	35	37	2.3801
2	0.1	(2048,)	0	0.2	0	20176	31	108	62	50	3.14396
3	0.15	(2048,)	0	0.2	0	6408	58	63	93	53	3.87836
4	0.2	(2048,)	0	0.2	0	0	82	33	121	44	4.65853
5	0.25	(2048,)	1	0.2	0	0	107	16	134	30	5.48928
6	0.3	(2048,)	11	0.2	0	0	108	6	140	14	6.13104
7	0.35	(2048,)	46	0.2	0	3760	97	2	114	41	6.88056
8	0.4	(2048,)	109	0.2	0	0	66	0	88	93	7.62838
9	0.45	(2048,)	167	0.2	0	11600	105	9	57	146	8.3797
10	0.5	(2048,)	156	0.2	1	0	112	20	32	208	9.12957
11	0.55	(2048,)	0	0.2	12	0	104	44	15	223	9.87812
12	0.6	(2048,)	0	0.2	50	13040	74	78	6	179	10.6106
13	0.65	(2048,)	0	0.2	124	0	49	115	0	121	11.3805
14	0.7	(2048,)	0	0.2	187	0	28	147	2	64	12.124
15	0.75	(2048,)	0	0.2	163	9162	13	178	9	26	12.8862
16	0.8	(2048,)	2	0.2	89	0	5	170	30	8	13.636
17	0.85	(2048,)	7	0.2	27	0	1	155	73	2	14.3866
18	0.9	(2048,)	19	0.2	5	14292	0	111	121	3	15.135
19	0.95	(2048,)	39	0.2	0	13404	0	74	150	26	15.8856
20	1	(2048,)	66	0.2	0	0	0	41	135	59	16.6334

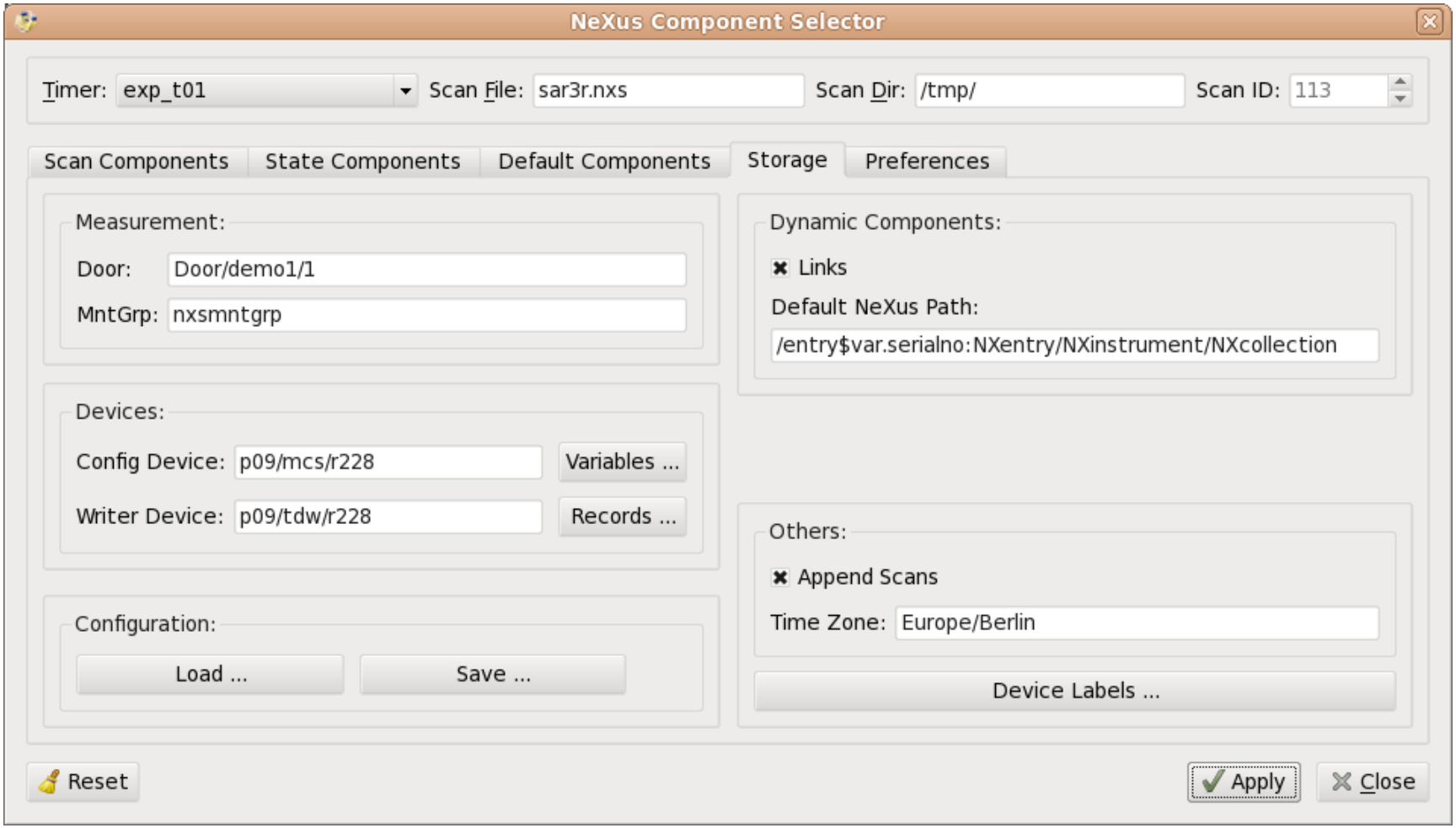
```
Operation saved in /tmp/sar3r.nxs (nxs)
Scan #114 ended at Wed May  7 09:22:10 2014, taking 0:00:17.231842.Dead time 75.6% (motion dead time 18.5%)

Door_demo1_1 [8]:
```

Here, scans are stored in one file: `sar3r.nxs`

NeXus Component Selector – for EXPERTS

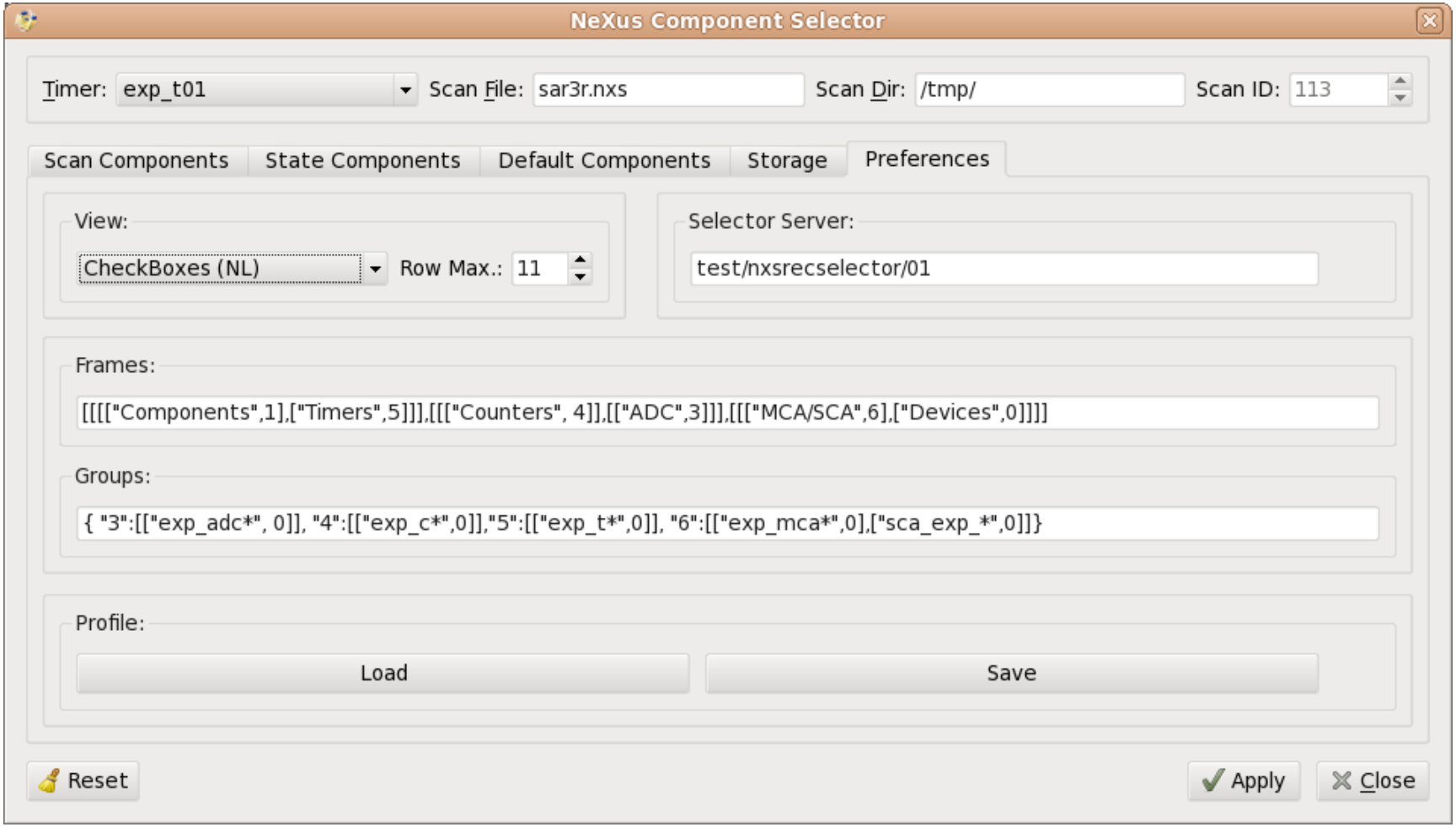
Storage parameters



Append Scans – scans stored in one NeXus file

NeXus Component Selector – for EXPERTS

Preferences



Changing the scan components layout.

Summary

NeXus Storage Software

- Creating fully describing files
- Configuration in components with datasources and strategy
- User Interfaces:
  - Sardana + GUI, Sardana + Macros, Python scripts
- Deployment:
  - roll-out software (FS-EC)
  - provide information about experiment (Beamline Scientists)
  - provide components (FS-EC)
  - select devices/components in Selector GUI and run scans (Users)

Summary

NeXus Storage Software

- Creating fully describing files
- Configuration in components with datasources and strategy
- User Interfaces:
  - Sardana + GUI, Sardana + Macros, Python scripts
- Deployment:
  - roll-out software (FS-EC)
  - provide information about experiment (Beamline Scientists)
  - provide components (FS-EC)
  - select devices/components in Selector GUI and run scans (Users)

Summary

NeXus Storage Software

- Creating fully describing files
- Configuration in components with datasources and strategy
- User Interfaces:
  - Sardana + GUI, Sardana + Macros, Python scripts
- Deployment:
  - roll-out software (FS-EC)
  - provide information about experiment (Beamline Scientists)
  - provide components (FS-EC)
  - select devices/components in Selector GUI and run scans (Users)

Summary

NeXus Storage Software

- Creating fully describing files
- Configuration in components with datasources and strategy
- User Interfaces:
  - Sardana + GUI, Sardana + Macros, Python scripts
- Deployment:
  - roll-out software (FS-EC)
  - provide information about experiment (Beamline Scientists)
  - provide components (FS-EC)
  - select devices/components in Selector GUI and run scans (Users)

Summary

NeXus Storage Software

- Creating fully describing files
- Configuration in components with datasources and strategy
- User Interfaces:
  - Sardana + GUI, Sardana + Macros, Python scripts
- Deployment:
  - roll-out software (FS-EC)
  - provide information about experiment (Beamline Scientists)
  - provide components (FS-EC)
  - select devices/components in Selector GUI and run scans (Users)

Thank You