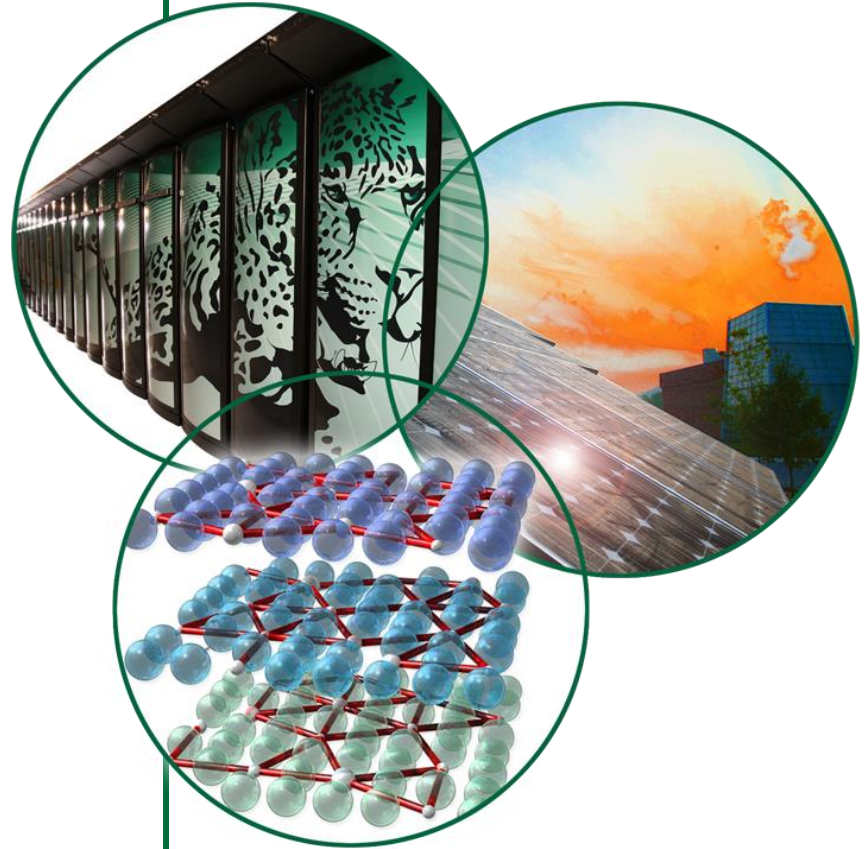


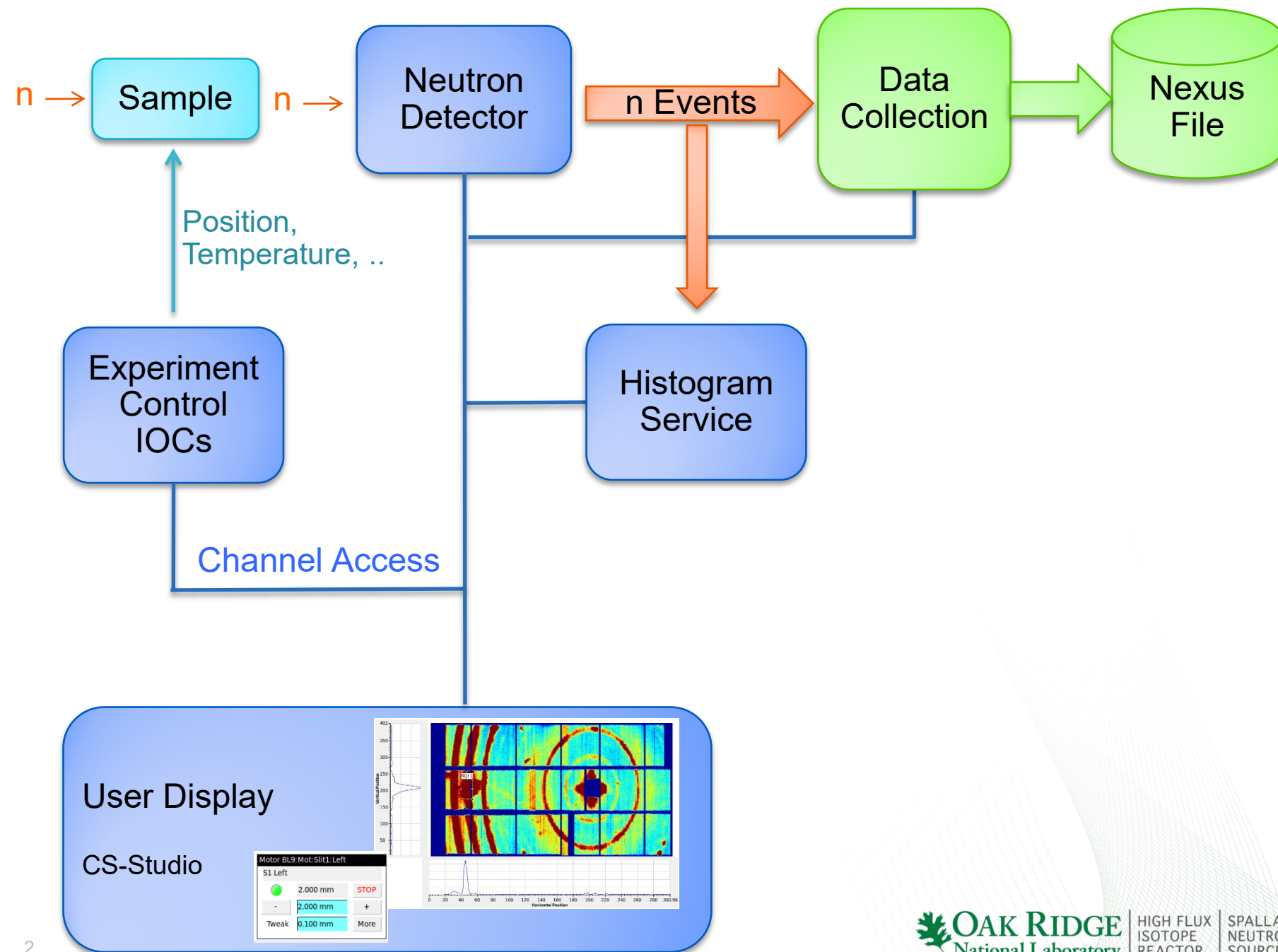


# Scan System Parallelization

## Recent Scan System Additions

Kay Kasemir,  
Oct. 2015







# Basic “Set” Command

**Set("some\_pv", 3.14)**

## Optionally

- ✓ **Await completion (put-callback)**
- ✓ **Check readback to match written value (same or other pv)**
  - ✓ **.. With numeric tolerance**
- ✓ **Timeout**

**Set("some\_pv", 3.14,  
    completion=True,  
    readback="some\_pv.RBV", tolerance=0.02,  
    timeout=30)**

# Other Commands

**Loop("pv", 1, 10, 0.1)**

**Wait("pv1", 100)**

**Wait("pv2", 100, comparison="increase by")**

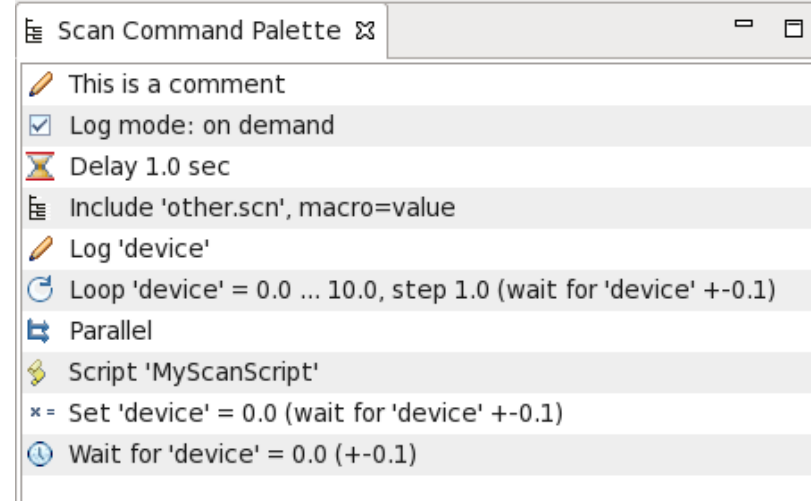
✓ Get-callback for initial value, then monitors

**Log "pv1", "pv2", "pv2"**

✓ Get-callback for current value, to RDB

**Invoke 'MyScript'**

✓ Jython-based custom commands



# PyScanClient

[home](#) | [search](#) |

[previous](#) | [next](#) | [modules](#) | [index](#)

## Scan Client

This example shows how to connect to the scan server, submit a scan, and monitor its progress using the basic API, without using site-specific settings or abstractions like the table-based scan.

### Example

```
import time
from scan import *

client = ScanClient('localhost')
print client

print client.serverInfo()

# Assemble commands for a scan
# Much more on that later...
cmds = [ Comment('Hello'), Set('motor_x', 10) ]

# Optionally, request a simulation that shows
# how 'Include' and 'Loop' commands get expanded.
simulation = client.simulate(cmds)
print simulation

# Submit scan for execution
id = client.submit(cmds, 'My First Scan')
print id

# Fetch information about scan
info = client.scanInfo(id)
print info
```

#### Table Of Contents

##### Scan Client

- [Example](#)
- [API](#)

#### Previous topic

##### Overview

#### Next topic

##### Scan Commands

#### This Page

##### Show Source

#### Quick search

Enter search terms or a module, class or function name.

# Site-Specific Settings

```
class BeamlineScanSettings(ScanSettings):  
    def __init__(self):  
        self.defineDeviceClass("chopper:.*", completion=True)  
        self.defineDeviceClass("motor.*", completion=True,  
                                readback=True)  
  
    def getReadbackName(self, device_name):  
        if "motor" in device_name:  
            return device_name + ".RBV"  
        return device_name  
  
"Set('x', 42)" → Set('x', 42)  
"Set('motor_x', 42)" → Set('motor_x', 42, completion=True,  
                             readback='motor_x.RBV)
```

# Table Scan

temperature	position
50	1
	2
	3
100	1
	2
	3

temperature	position
[50,100]	[1, 2, 3]

```
Set('temperature', 50),  
Set('position', 1),  
Set('position', 2),  
Set('position', 3),  
Set('temperature', 100),  
Set('position', 1),  
Set('position', 2),  
Set('position', 3),
```

position	Wait For	Value
2	counter	10000

```
Set('position', 2.0, completion=true, readback='position.RBV', timeout=100)  
Wait('counter', 10000.0, comparison='>=')  
Log('position', 'counter')
```



# Table Scan



Table Scan 

Table: /tmp/10076-mapping\_Mg-4-1mm.csv 

Comment	BL7:CS:IPTS	X	Y	Z	Wait For	Value	Or Time
point 1 bent	10076		-0.45		bm2	17600000	15300
point 2 edge			4.45		bm2	17600000	15300
point 3 edge			9.55		bm2	17600000	15300
point 4 edge			36.2		bm2	17600000	15300

Load

Save

Submit

Add Device

Help

\*mapping\_exp3.csv - Gnumeric

File Edit View Insert Format Tools Statistics Data Help

Sans 10

E1 = HROT

	A	B	C	D	E	F	G	H	I	J	K	L
	BL7:CS:IPTS	Comment	X	Y	HROT	Speed1	Speed2	BL7:Cho	Wavelen	Wait For	Value	Or Time
1	11090	60Hz, 1mm X step	-5		0	60	60	1.44	1.5	bm2	1000000	3600
2		60Hz, 1mm X step	-4		0.1					bm2	1000000	3600
3		60Hz, 1mm X step	-3		0.2					bm2	1000000	3600
4		60Hz, 1mm X step	-2		0.3					bm2	1000000	3600
5		60Hz, 1mm X step	-1		0.4					bm2	1000000	3600
6		60Hz, 1mm X step	0		0.5					bm2	1000000	3600
7		60Hz, 1mm X step	1		0.6					bm2	1000000	3600
8		60Hz, 1mm X step	2		0.7					bm2	1000000	3600
9		60Hz, 1mm X step	3		0.8					bm2	1000000	3600
10		60Hz, 1mm X step	4		0.9					bm2	1000000	3600
11		30Hz, 0.4mm Y Step	0	-2	1	30	30	2.88	2	neutrons	1000000	3600
12		30Hz, 0.4mm Y Step		-1.6	1.1					neutrons	1000000	3600
13		30Hz, 0.4mm Y Step		-1.2	1.2					neutrons	1000000	3600
14		30Hz, 0.4mm Y Step		-0.8	1.3					neutrons	1000000	3600
15		30Hz, 0.4mm Y Step			1.4					neutrons	1000000	3600

mapping\_exp3.csv Sum=19

# Parallel Command

Set two PVs to a value, each awaiting callback completion:

```
>>> cmd = Parallel(Set('x', 1, completion=True),  
...               Set('y', 2, completion=True))
```

+p x	+p y	Wait For	Value
1	2	counter	10000
3	4	completion	20

Result:

```
Parallel(Set('x', 1.0), Set('y', 2.0))  
Wait('counter', 10000.0, comparison='>=')  
Log('x', 'y', 'counter')  
Parallel(Set('x', 3.0), Set('y', 4.0))  
Log('x', 'y', 'counter')
```

# Sequence Command

- Build parallel command chains

```
>>> Parallel( Sequence(Set('x', 1), Wait('x_loc', 10) ),  
>>>             Sequence(Set('y', 2), Wait('y_loc', 20) ) )
```

- Define 'Meta' commands

```
def start():  
    """Start data acquisition"""  
    return Sequence(  
        Set('BL17:CS:RunControl:start', 1),  
        Wait('BL17:CS:RunControl:StateEnum', 3),  
    )
```

Table scan can use these before/after 'Wait For'

# Scripted Scan

The screenshot displays the PyDev IDE interface. On the left, the Navigator pane shows a project structure with folders like 'gateway', 'opi', 'pvttable', and 'scan'. Under 'scan', there is a 'demo' folder containing 'SequoiaScanExample2.py', which is currently selected. The main editor window shows the code for 'SequoiaScanExample2.py'. The code is a Python script that uses the 'beamline\_scan' module to create a table-based scan. It sets 'pcharge\_per\_sec' to 1e9, creates a 'CommandSequence' object, appends commands to set the title and slits, and then creates a table scan with parameters 'demo:x', 'Wait For', and 'Value'. The table scan is defined with a range of '[0, 90, 180]' for 'pcharge' and a value of '5\*pcharge\_per\_sec'. The script prints the table and submits the scan to the scan client, waiting for completion. A context menu is open over the code, showing options like 'Run As', 'Debug As', 'Team', 'Compare With', 'Replace With', and 'PyDev'. The Console pane at the bottom shows the output of the script, including the table definition and the submission of the scan. A tooltip is also visible, providing documentation for the 'submit' method of the 'scan\_client' object.

```
# Example that combines basic commands with table-based scan
from beamline_scan import *

# 1e9 per second
pcharge_per_sec = 1e9

# Assemble commands
cmds = CommandSequence()
cmds.append(SetTitle("Script With Table"))
cmds.append(SetSlits(left=20, right=20))

table = createTableScan([ "demo:x",          "Wait For", "Value"          ],
                        [ "[0, 90, 180]", "pcharge", 5*pcharge_per_sec ] ])

print table
cmds.append(table.createScan())
id = scan_client.submit(cmds, "Table-based")
print scan_client.waitUntilDone(id)
```

PyDev Console [1]

```
rows= [ [ "[0, 90, 180]", "pcharge ", "5000000000.0" ],
        ]
>>> cmds.append(table.createScan())
>>> id = scan_client.submit(cmds, "Table-based")
>>>
```

Submit scan to scan server for execution

:param cmds: List of commands,  
:class: ~scan.commands.commandsequence.CommandSequence`  
or text with raw XML format.  
:param name: Name of scan  
:return: ID of submitted scan

Examples::

```
>>> cmds = [ Comment('Hello'), Set('x', 10) ]
```

# GUI for Routine Beam Line Task

File Edit Search Run CSS Window Help

90%

Scan Editor OPI Editor CSS

Navigator

Display

Camera Control

Exposure Time (S) 180.000

Binning 1 1

ADC Speed 1.00 MHz

Shutter Mode Auto

Camera State Idle

Start Stop

Cooling

Cooler On On

Temperature -60.00 -59.777

Status Stabilized at set pc

Advanced

Full Control (Simulated)

Full Control (Andor)

File I/O Configure

General Camera

File Path /home/controls/cg1d/data/Tuesday/Turbine\_4\_CT

File Name Turbine\_CT Next File # 280 280

Last File Name /home/controls/cg1d/data/Tuesday/Turbine\_4\_CT/20130109\_Turbine\_CT\_018

Andor Message

Motors

Motor	Readback	Position	Left/Move/Right	Limits
Lift Table	83.1 mm	83.1 mm		STOP
Short Axis	80.0 mm	80.0 mm		STOP
Long Axis	132.5 mm	132.5 mm		STOP
Large Rotation T.	90.0 deg	90.0 deg		STOP
Detector Table	225.0 mm	225.0 mm		STOP Enabled
Small Rotation T.	181.4 deg	181.4 deg		STOP
Camera Vert.	70.0 mm	70.0 mm		STOP
Robofocus	50	50	In Out Cabinet...	

Motor Guide

- Lift Table

smaller - Short Axis

LARGER + Short Axis

- Long Axis

+ Lift Table + Long Axis

CT Scan Camera Scan

Configuration

Start 0 End 182 Step 0.650

Device Large .. Small Rot. Table

Exposure 180.000 Delay 0 sec Simulate?

Directory /home/controls/cg1d/data

File name Turbine\_CT

Go

Status

Angle 90.0 deg Scan Active

Camera Idle

Last file 1130109\_Turbine\_CT\_0180\_181.350\_0279.f

E-STOP

Console Scan Monitor

ID	Created	Name	State	%	Runtime	Finish	Command	Error
153	2013-01-08 17:54:24	Rotation Scan: Turbine_CT	Finished - OK		14:35:06	08:29:31	- end -	
152	2013-01-08 17:38:07	Rotation Scan: Turbine_CT_test	Finished - OK		00:15:35	17:53:42	- end -	

Scan Server Memory: 25.1 MB / 1744.0 MB (1.4 %)

# Under the hood


CT Scan Camera Scan

Configuration


Start  End  Step


Device

Exposure  Delay  ☒ Simulate?

Directory  

File name

 Go


```
ct 
from beamline_scan import *

''' Fetch parameters from display'''
start = getWidgetDouble("start")
# .. more

'''Create scan sequence'''
cmds = [
    Loop(device, start, end, step,
        [
            # More to trigger camera, wait until image saved, ..
        ])
]

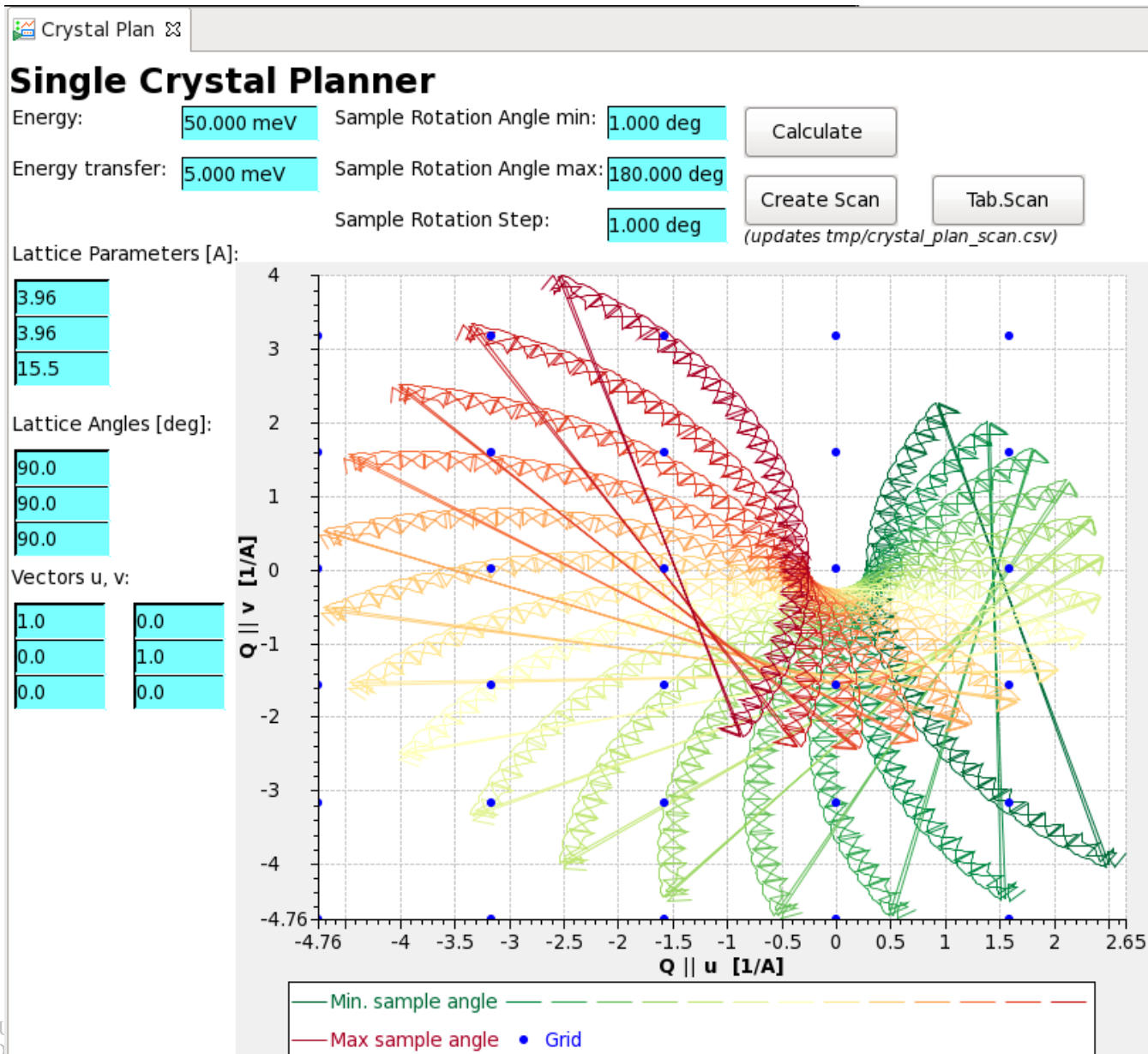
'''Submit for execution'''
scan_client.submit(cmds)
```

## Scan Server

CT.scn 

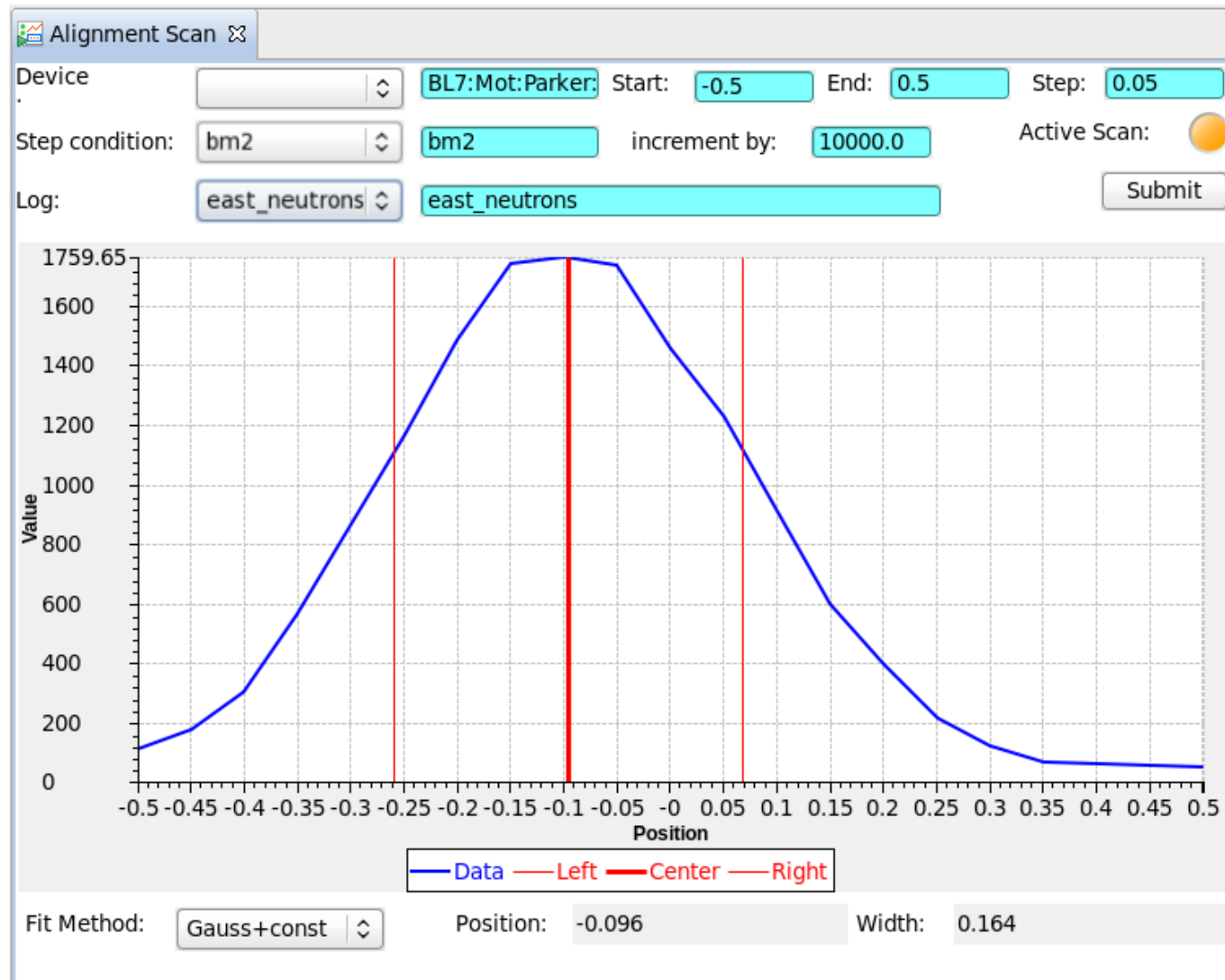
```
* = Set 'CG1D:Cam:Cam1:AcquireTime' = 180.0 (wait for 'CG1D:Cam:Cam1:AcquireTime' +-0.1)
* = Set 'CG1D:Cam:Cam1:FilePath' = "/home/controls/cg1d/data" (wait for 'CG1D:Cam:Cam1:FilePath' +-0.1)
* = Set 'CG1D:Cam:Cam1:FileName' = "Turbine_CT" (wait for 'CG1D:Cam:Cam1:FileName' +-0.1)
▼ Loop 'CG1D:Mot:RotTable' = 0.0 ... 182.0, step 0.65 with completion (wait for 'CG1D:Mot:RotTable.RBV' +-0.065)
  Script 'SetImageNameLarge'
  ▼ Loop 'CG1D:Scan:Index' = 1.0 ... 1.0, step 1.0 (wait for 'CG1D:Scan:Index' +-0.1)
    * = Set 'CG1D:Cam:Cam1:Acquire' = 1.0
    Delay 180.0 sec
    Wait for 'CG1D:Cam:Cam1:DetectorState_RBV' = 0.0 (+-0.1)
    Log 'CG1D:Mot:RotTable', 'CG1D:Cam:Cam1:FileNumber_RBV'
    Script 'ProcessImage'
```

# Crystal Planner





# Alignment Scan





# Summary

- ✓ **Scan System helps automate SNS since 2013**
- ✓ **New *Parallel* & *Sequence* commands**
- ✓ **Python API**
  - ✓ **Pure Python instead of Jython**
  - ✓ **Site-specific settings**
  - **Thanks to Qui Yongxiang, Guobao Shen, Dylan Maxwell**

## Biggest Issue:

- ☐ **Can't use numpy (Fortran) inside *Script* command (Jython)**



# What it is and isn't

Is

- ✓ Automation via Channel Access
- ✓ Scan = Batch of commands
- ✓ Queue multiple scans
- ✓ Basic value logging
- ✓ Submit, monitor, pause, resume, abort

Isn't

- ☐ Synchronization of actions beyond Channel Access
- ☐ Data Acquisition (log every event, catalog, keep forever)

# Like Sequencer?

**Yes: Read/write Channel Access**

- ✓ **No compilation**
- ✓ **Monitor & control progress of scan**
- ✓ **Basic data log**
- ✓ **Schedule multiple scans**
- ☐ **No arbitrary C code**
- ☐ **No 'if-then-else' command**

# Like Scan Record?

Loop y=1..10:

Loop x=1..5:

Set "det\_trigger"=1 with completion,

Log "x.RBV", "y.RBV", "det\_counts"

Monitor, pause, resume, abort.

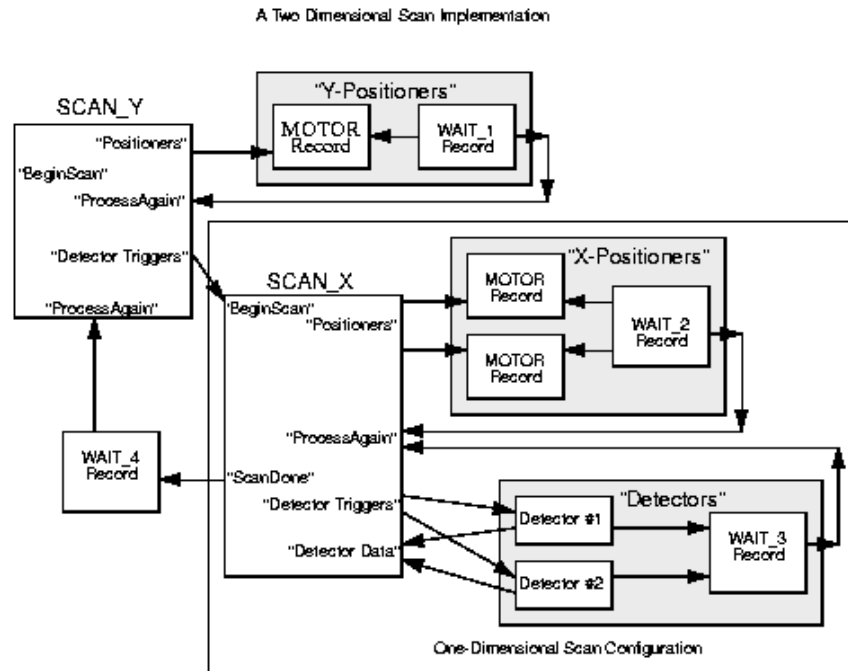
Save, edit, resubmit scans

vs. save/restore scan records.

✓ Add 3<sup>rd</sup> loop without rebooting IOC to add 3<sup>rd</sup> scan record

✓ Queue multiple scans

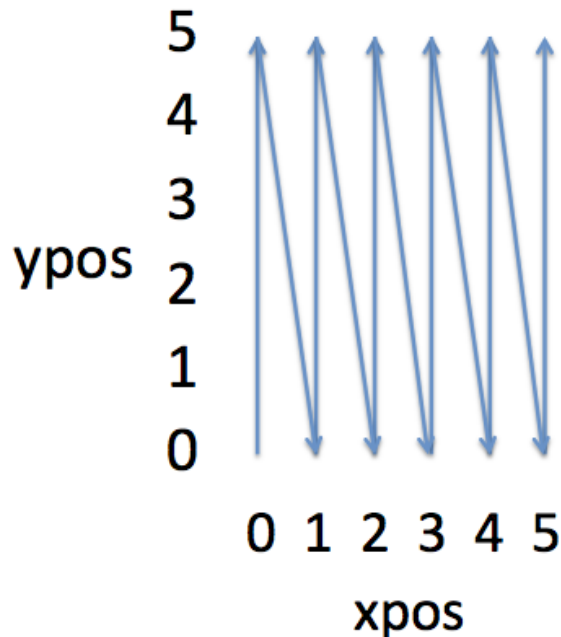
❑ Logs to RDB w/ REST readout. No MDA/XDR/Nexus.



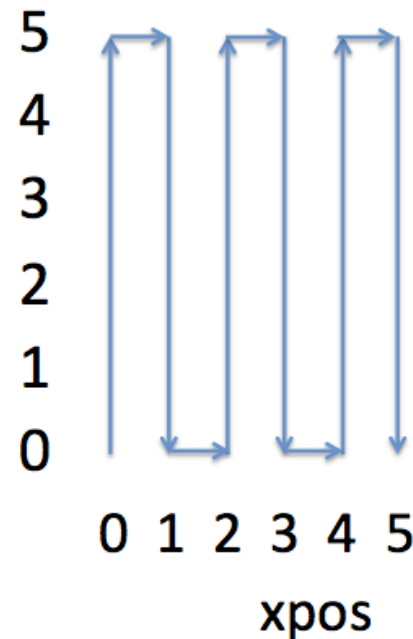
# Direction of (Nested) Loops

- `Loop('x', 0, 5, 1) → 0, 1, 2, 3, 4, 5`
- `Loop('y', 5, 0, -1) → 5, 4, 3, 2, 1, 0`
- `Loop('y', 0, 5, -1) → Alternate direction on 'mismatch'`

Normal Scan



Alternating Scan



# REST Interface

Scan Server REST Interface - Mozilla Firefox

Scan Server REST Interfa... x +

localhost:4810/submit.html

## Submit Scan

Name:

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
  <comment><text>Example scan submitted via REST interface</text></comment>
  <set><device>xpos</device><value>1</value></set>
  <log><devices><device>xpos</device></devices></log>
  <set><device>xpos</device><value>2</value></set>
  <log><devices><device>xpos</device></devices></log>
</commands>
```

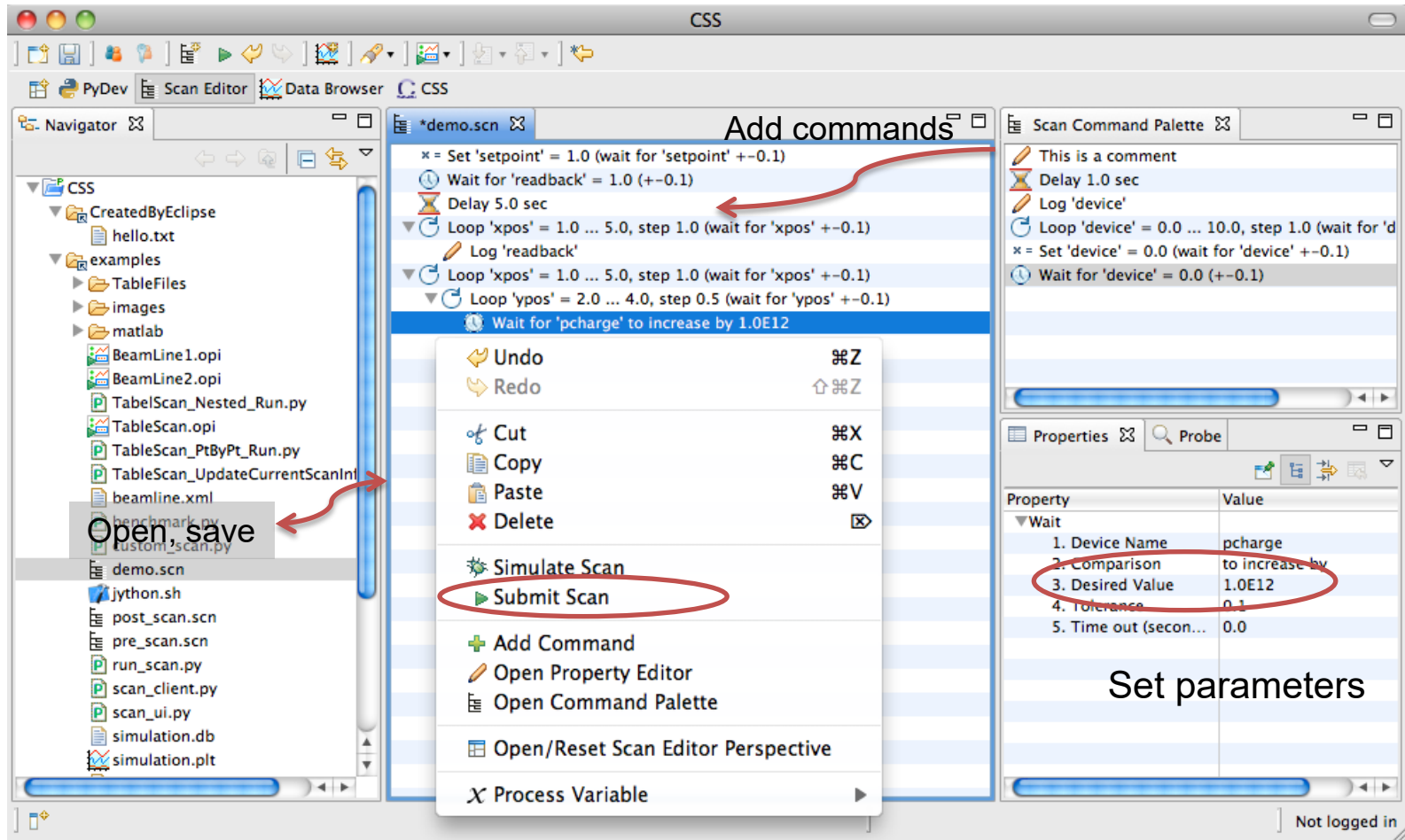
Scan:

[Back to main](#)

## Scans

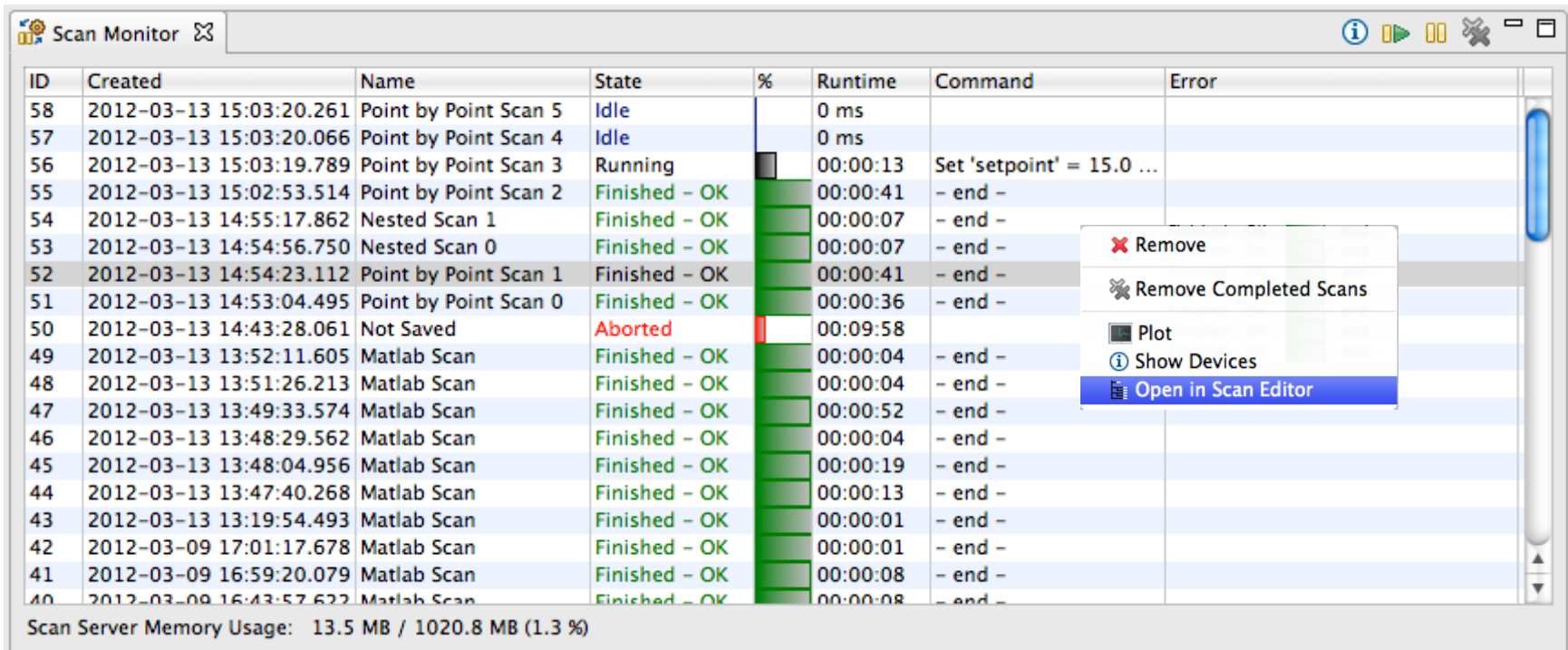
ID	Name	Created	State	Percentage	Runtime	Finish	Command
<a href="#">89 (cmds)</a> <a href="#">(data)</a>	Example	2015-05-07 13:02:15	Running <input type="button" value="Pause"/> <input type="button" value="Abort"/>	33.3%	1.4 seconds	2015-05-07 13:02:16	Delay 60.0 sec
<a href="#">88 (cmds)</a> <a href="#">(data)</a>	Example	2015-05-07 13:01:13	Finished <input type="button" value="Delete"/>	100.0%	0.3 seconds	2015-05-07 13:01:14	- end -

# Scan Editor

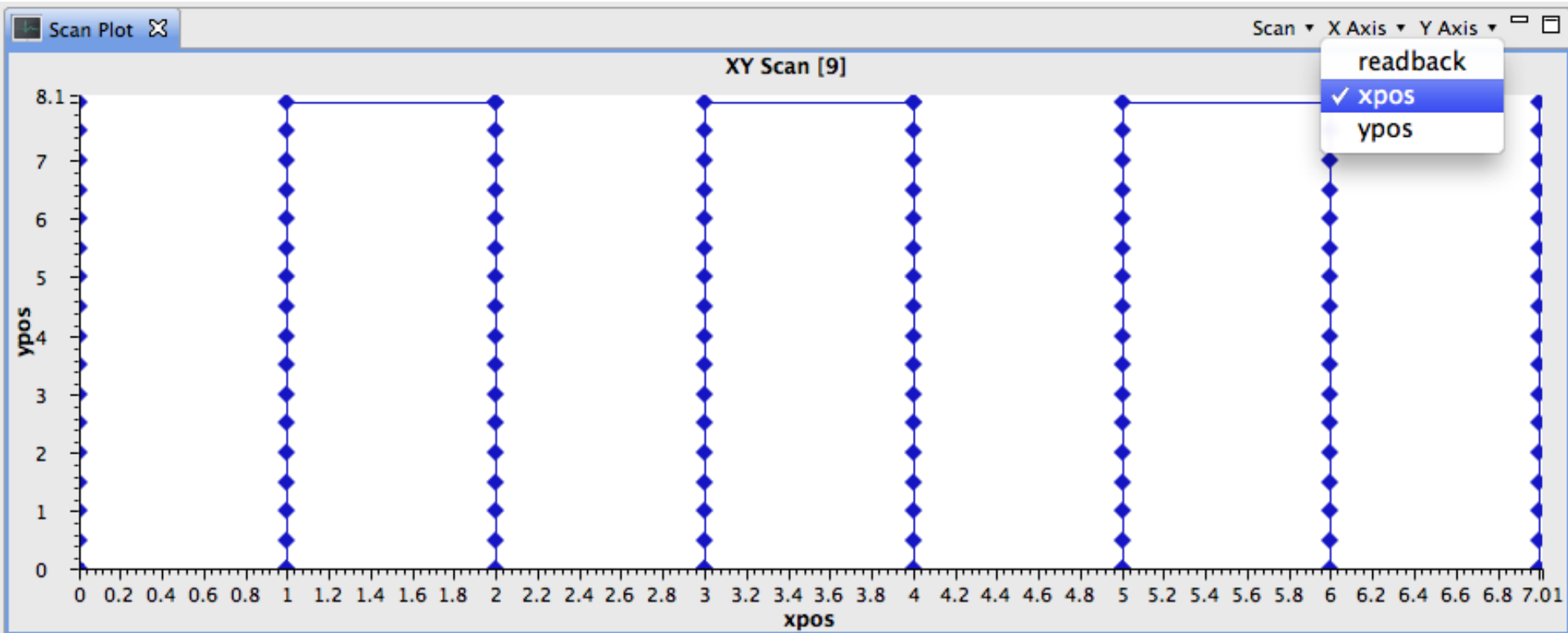




# Scan Monitor



# Scan Plot



- Plot variables logged by scan
- Get data from Running or Finished scans