OAK RIDGE National Laboratory

PV Access

Oct. 2018

Kay Kasemir

Python examples by Amanda Carpenter

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

History

Channel Access

- DBR_*: Numbers, enums, string, scalar and array, with time, alarm, limits
- Since beginning of EPICS
- Still fully supported

PV Access

- PV Data: Arbitrary structures
- Started as “EPICS V4” development
- Since EPICS 7 (Dec. 2017) included in EPICS base

PV Access

Fundamentally similar to Channel Access

- Name search via UDP
- Connection for data transfer via TCP
- EPICS_PVA_ADDR_LIST, EPICS_PVA_AUTO_ADDR_LIST

Get, put, monitor

- Plus an 'RPC' type operation

Arbitrary PV Data structures instead of DBR_.. types

Custom Data: Great, but then what?

```
structure:
double    value
short     status
short     severity
string    units
time      timeStamp
…
```

```
structure:
short     level
double    data
string    type
time      stamp
…
```

```
structure:
short     level
double    wert
string    typ
long      zeit
…
```

```
structure:
short     info
double    content
string    meta
long      ms
…
```

- Which number to show on a user display?
- What units?
- Is this an alarm?
- Time stamp?

“Normative Types”

- Channel Access

```
struct  dbr_ctrl_double:
short    status
short    severity
short    precision
char     units[8]
… no timestamp …
double   value
```

```
struct  dbr_time_double:
short     status
short     severity
timestamp stamp
double    value
```

You get what you request
(network always transfers complete struct)

- PV Access

```
epics:nt/NTScalar:
double   value
short    status
short    severity
string   units
time     timeStamp
…
```

You get what you request
(but network only transfers changes)

Channel Access vs. PV Access

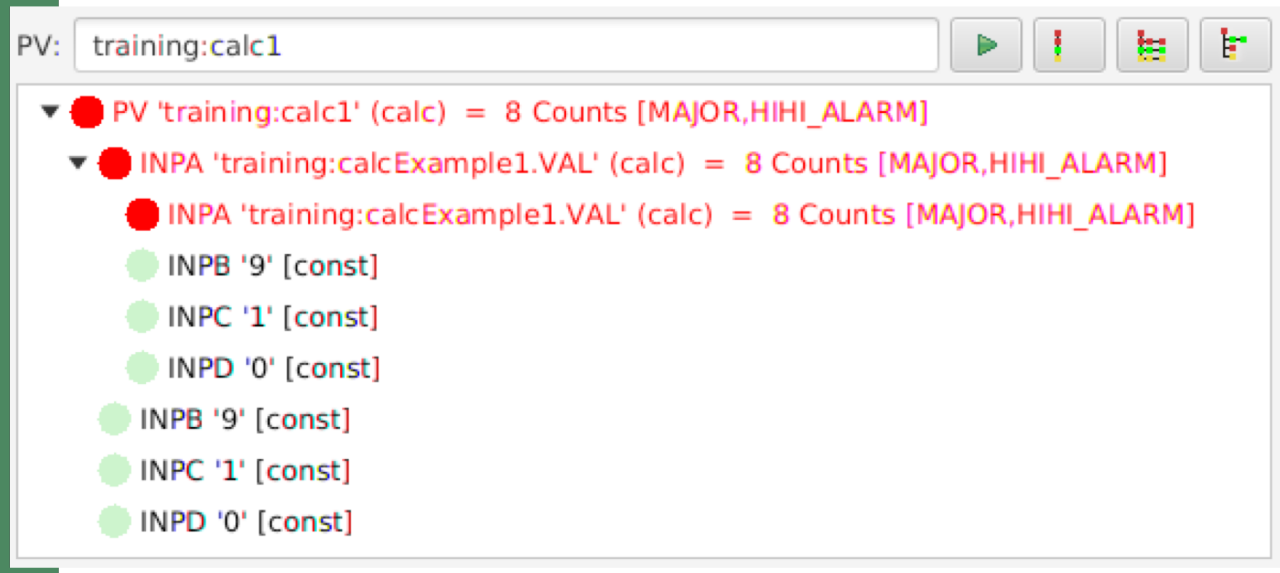
EPICS 7 IOCs include PVA server

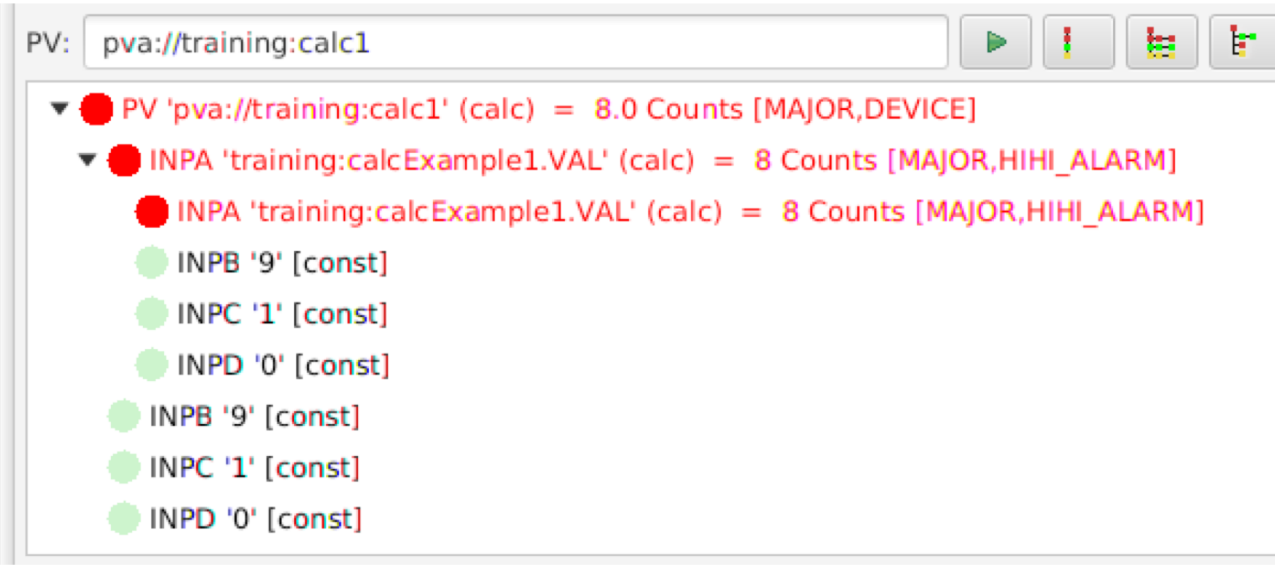
Similar command line tools:

```
cainfo training:ai1

caget training:ai1

camonitor training:ai1

caget –d CTRL_DOUBLE training:ai

caget training:ai1.SCAN
```

```
pvinfo training:ai1

pvget training:ai1

pvget –m training:ai1

pvget -r 'field()' training:ai1

pvget training:ai1.SCAN
```

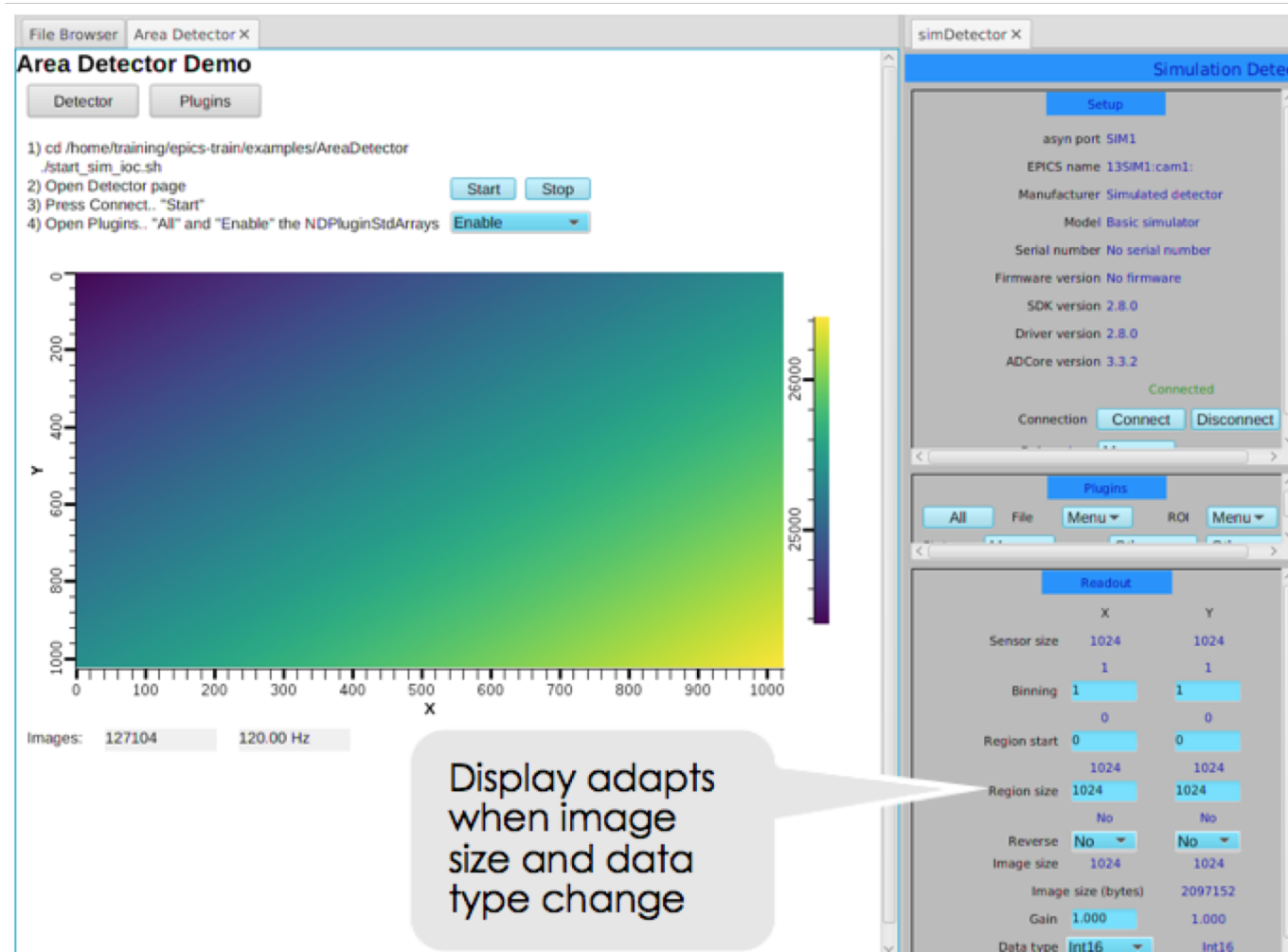
CS-Studio: Use ‘pva://…’





Images: Normative type NTNDArray

- Served by Area Detector (NDPluginPVA) or ‘start_imagedemo’
- pvinfo IMAGE
  - Value, dimensions, codec
- CS-Studio: Image widget
  - Only needs pva://IMAGE



Custom PV Data

SNS Beam Lines started to use this in ~2014

```
start_neutrondemo
pvinfo neutrons
```

Allows fetching just what's needed:

```
# For detector pixel display
pvget -r 'field(pixel)' neutrons
pvget –m -r 'field(timeStamp, pixel)' neutrons

# For energy displays
pvget –m -r 'field(time_of_flight, pixel)' neutrons
```

Custom PV Data in CS-Studio

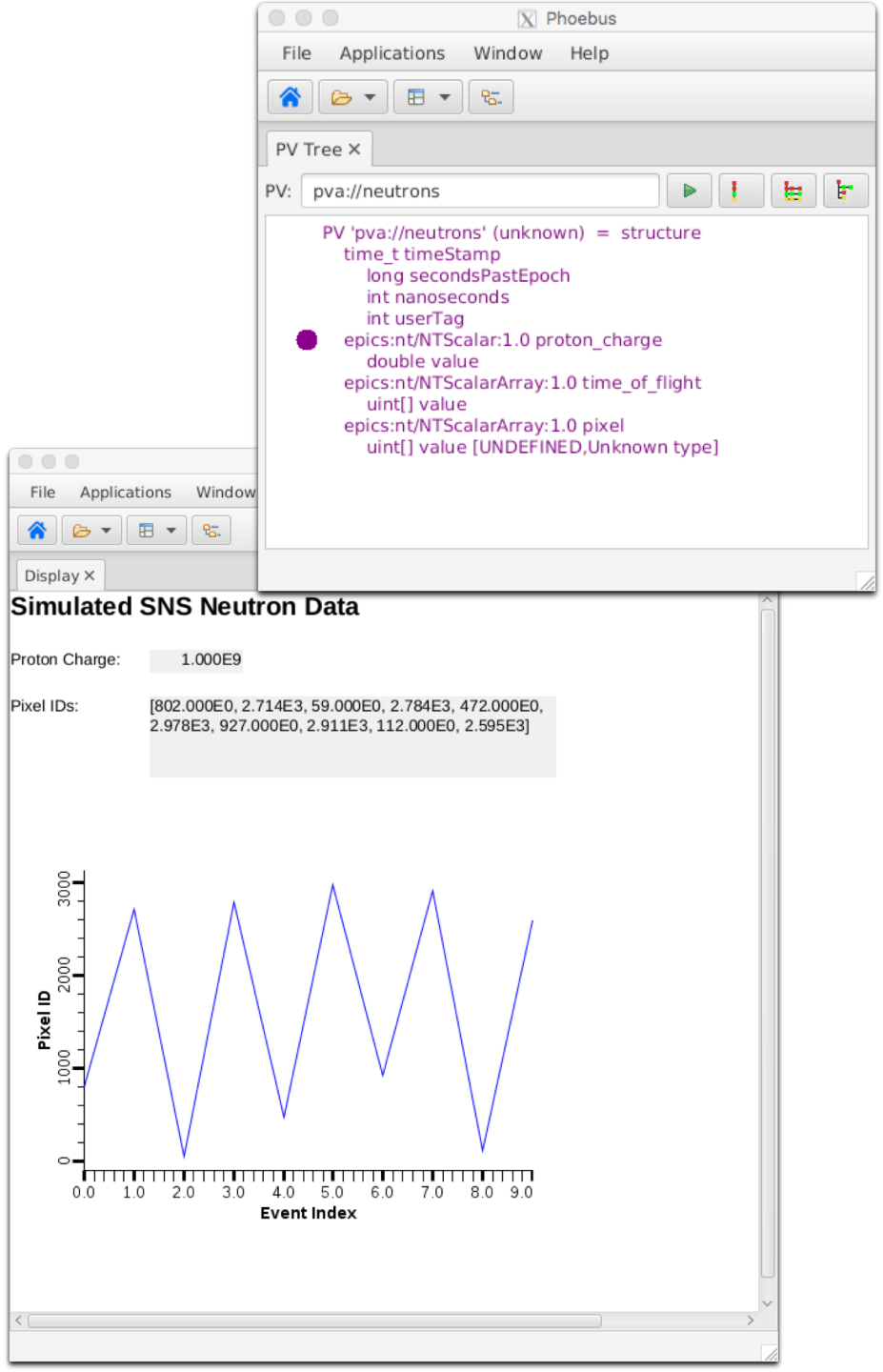
Cannot handle *arbitrary structure*

```
pva://neutrons
```

Can handle fields which are *scalar* or *array*

```
pva://neutrons/proton_charge
```

```
pva://neutrons/pixel
```



PV Access and Python

Basic ‘get’

```
cd ~/epics-train/examples/python/
python example1.py
```

‘monitor’

```
python example2.py
```

PV Access API with Channel Access as “Provider”

PV Access supports both the actual PvAccess protocol but also Channel Access.

New tools, written for PVA, can thus fall back to CA:

```
python example3.py
```

Tools like CS-Studio can use both ca:// and pva://, so multiple transition options.

Custom PV Data in Python Client

Python receives data as dictionary, access to any element

```
python neutrons.py
```

Custom PV Data from Python Server

```
# Server
python server.py


# Client
pvinfo pair
pvget -m -r "x, y" pair
```

Surprisingly easy:

```
pv = PvObject({'x': INT, 'y' : INT})

server = PvaServer('pair', pv)

x = 1
while True:
    pv['x'] = x
    pv['y'] = 2*x
    server.update(pv)
    sleep(1)
    x = x + 1
```

More Examples

Display Builder `pva_server_ramp`

Python code that serves ‘pva://ramp’ with alarm, prec, timestamp, …

Display Builder `table_server`

Python code that serves ‘pva://table’ as “NTTable”

→Not practical to replace regular IOCs with python,
but useful when custom data is needed

Custom PV Data from IOC Records

`makeBaseApp.pl –t example` includes "group",
see `~/epics-train/examples/ExampleApp/Db/circle.db`

Calc records ..:circle:x & ..:circle:y compute (x, y) coordinate on circle

info() annotations create PV "training:circle" PV as struct `{ angle, x, y }`

PVA "training:circle" updates atomically

`camonitor training:x training:y` receives separate x, y updates
`pvget –m training:circle` will always see `sqrt(x²+y²)==1`

```
cd ~/epics-train/examples/python
python circle.py
```

PV Access

- Update to Channel Access
  - Both can be used in parallel
- Similar, but supports custom data types
  - Won't replace IOC, but useful for special cases
- Since EPICS 7 included in base
  - Details of 'group', PVA gateway, 'field(…)' access still evolving