OAK RIDGE National Laboratory

PV Access

Feb. 2022

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

EPICS Network Protocols

Channel Access

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

PV Access

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

First Glance

- softIocPVA instead of softIoc

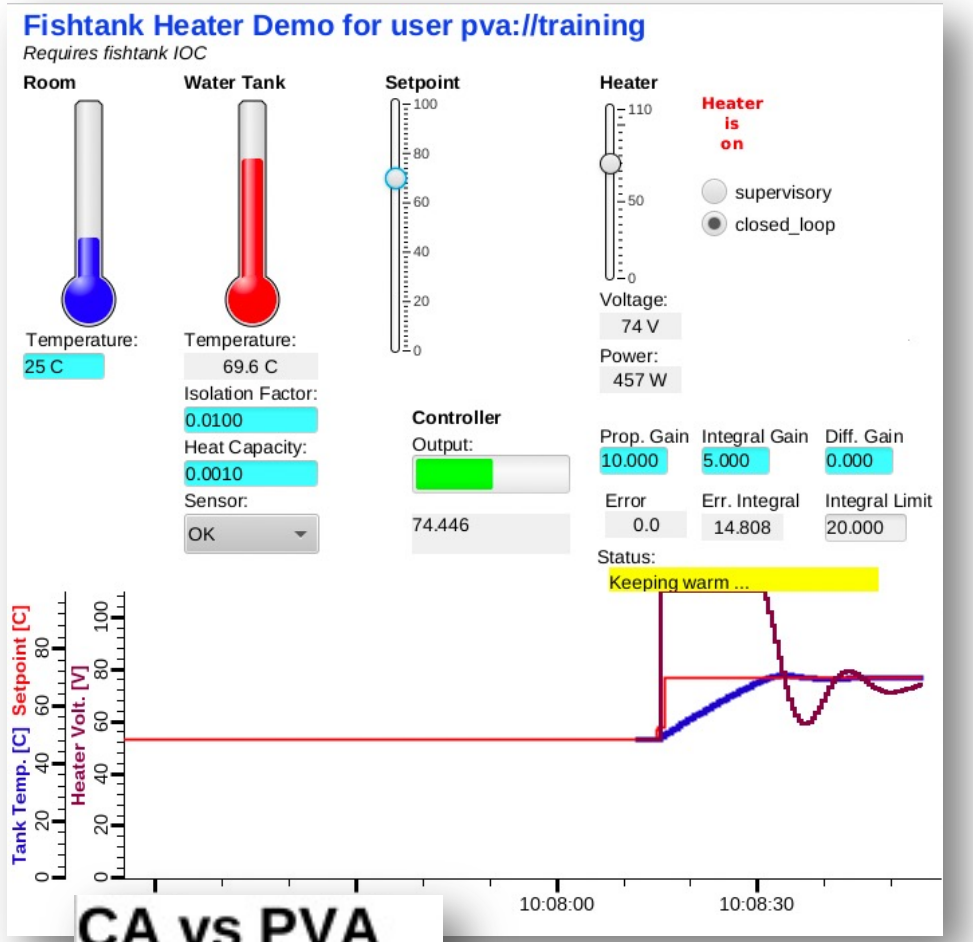
```
# We did this before:
cd /ics/examples/02_fishtank
cat st.cmd

# Compare, then run:
cd /ics/examples/24_pvaccess
cat st.cmd
./st.cmd
```

- pv... instead of ca...

```
camonitor training:setpoint training:tank
pvmonitor training:setpoint training:tank
pvput training:setpoint 40
caput training:setpoint 30
```

- CS-Studio:

```
css -resource /ics/examples/24_pvaccess/pva.bob
```



PV Access

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?

Fred's structure:

double	value
short	status
short	severity
string	units
time	timeStamp
…	

Keith's structure:

short	level
double	data
string	type
time	stamp
…	

Jürgen's structure:

short	grad
double	wert
string	typ
long	zeit
…	

Jane's 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

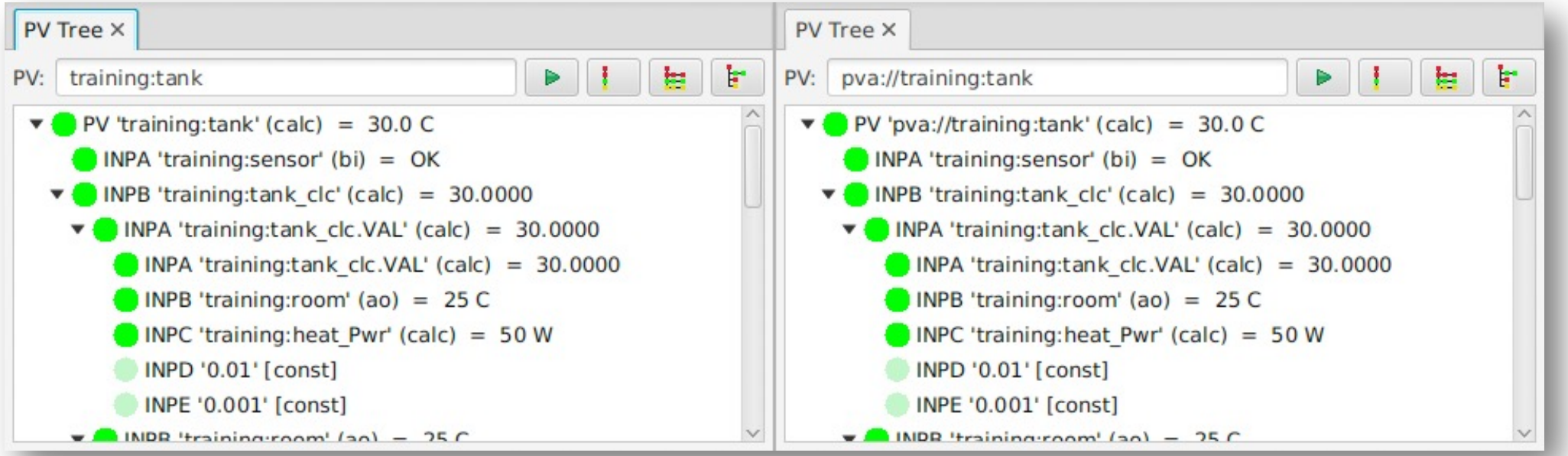
Similar command line tools:

```
caget training:tank

camonitor training:tank

cainfo training:tank


caget –d CTRL_DOUBLE training:tank

# Not supported
camonitor –d CTRL_DOUBLE training:tank


caget training:tank.SCAN
```

```
pvget training:tank

pvmonitor training:tank

pvinfo training:tank


pvget –M raw training:tank

# Note first few updates!
pvmonitor –M raw training:tank


pvget training:tank.SCAN
```

CS-Studio

- Use pva://… prefix to select PV Access



- Use ca://… prefix to select Channel Access
- Set default in /ics/tools/phoebus/settings.ini

```
# Default PV type: ca or pva?
org.phoebus.pv/default=pva
```

So it's very similar, maybe more efficient…

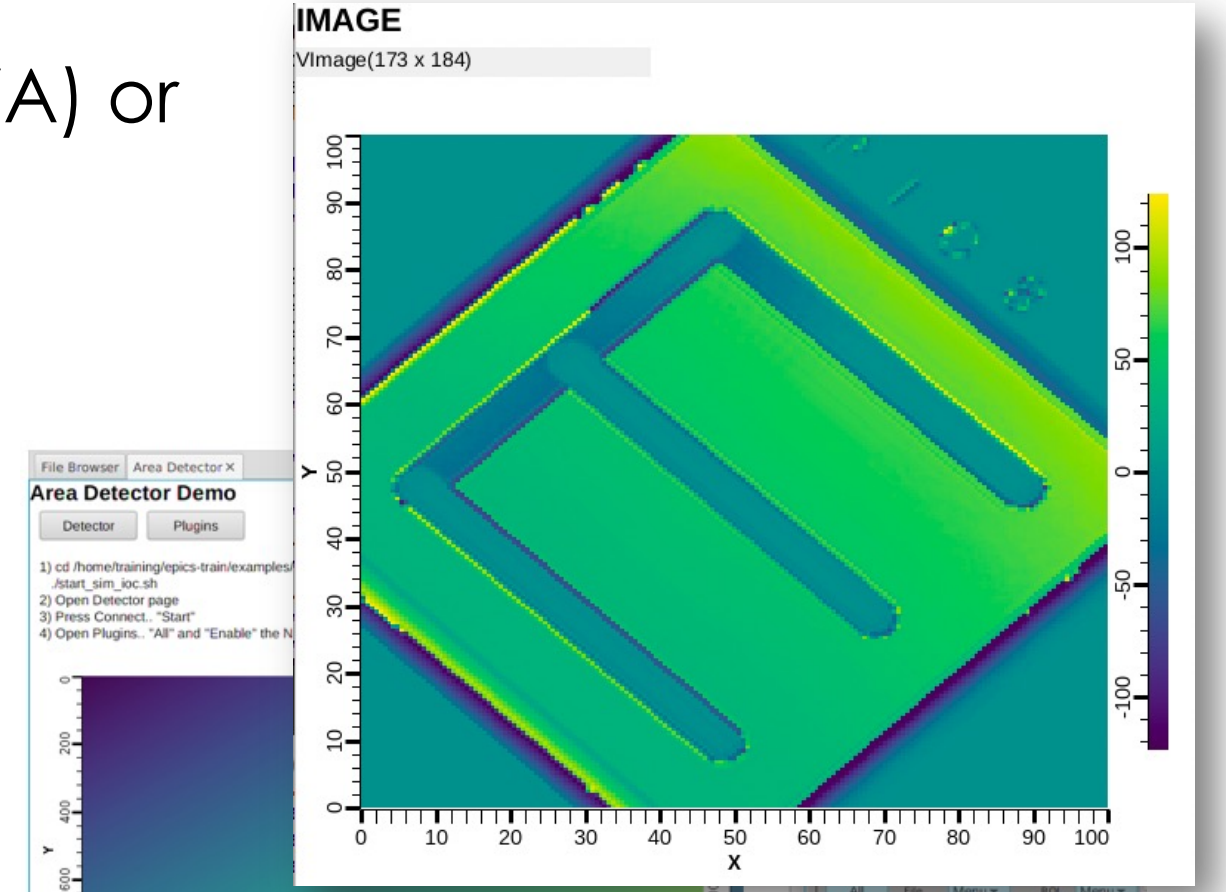
What's really new?

How about those custom structures?

Images: Normative type NTNDArray

- See Area Detector (NDPluginPVA) or

```
cd /ics/examples/24_pv_access
./start_imagedemo
```



- CS-Studio: Image widget
  - Only needs pva://ImagePV

Custom PV Data

SNS Beam Lines started to use this in ~2014

```
cd /ics/examples/24_pvaccess
./start_neutrondemo

pvinfo neutrons
pvmonitor neutrons
```

Allows fetching just what's needed:

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


# For energy displays
pvmonitor -r 'field(time_of_flight, pixel)' neutrons
```

Custom PV Data in CS-Studio

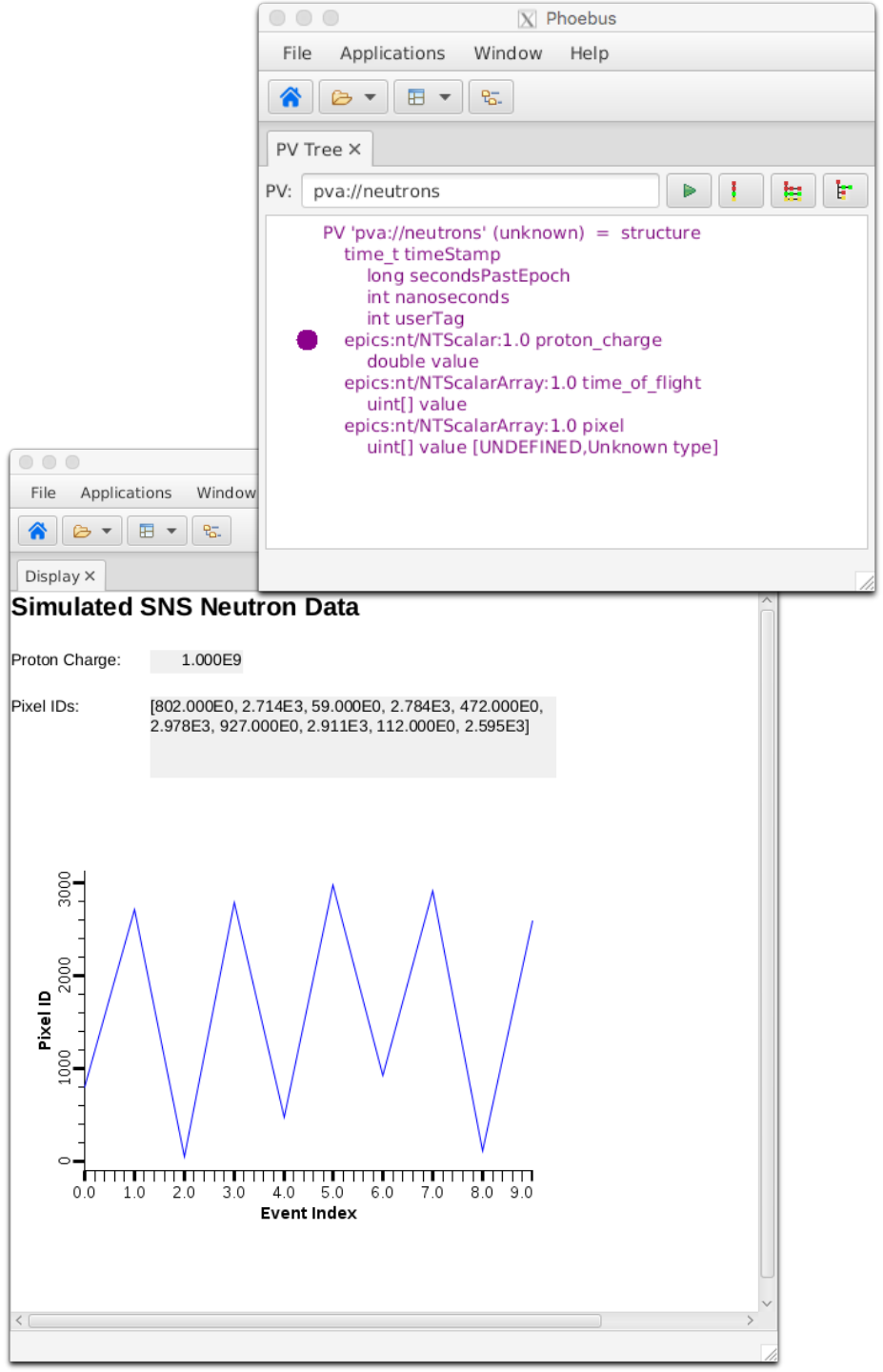
Cannot handle *arbitrary structure*

```
pva://neutrons
```

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

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

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



PV Access and Python

- Basic ‘get’, ‘put’, ‘monitor’
- PV Access server for normative types or custom data

  – See *.py examples under

    `cd /ics/examples/24_pvaccess`

Custom PV Data from IOC Records

`makeBaseApp.pl –t example` includes "group",
see `/ics/examples/10_customApp/Db/circle.db`,
`/ics/examples/iocBoot/ioc_custom`

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

```
cd /ics/examples/24_pvaccess
python circle.py
```

State of PV Access by end of 2021

Done, operational

- Server and client libraries for C++, Java, Python
  - 2nd version
- Area Detector image transfer
  - Used to distribute processing from camera hosts
- Custom data servers and clients
  - SNS: neutron data
  - APS: services

Done, to be tested

- PV server for records in IOC
  - All record types
  - Full ‘units’
  - Support changing metadata
- CS-Studio client
- Gateway

To do

- IOC links
  - “CP” links → PVA links
  - Channel Access Get/put callback → ??
- How to best combine data from records into custom PVA data?

Summary: PV Access is ..

- 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 IOC
  - Details of 'group', 'field(…)' access still evolving