PV Access Introduction

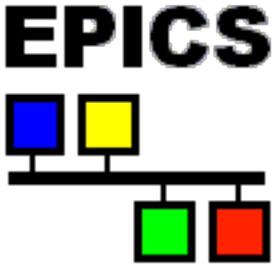
EPICS Collaboration Meeting

Slovenia, Sept. 2022

Kay Kasemir

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

What is EPICS?



Network Diagram

Network Diagram (new)

Network Protocol plays central role

“Integrate into EPICS”

=

“Make accessible on Network”

EPICS Network Protocols

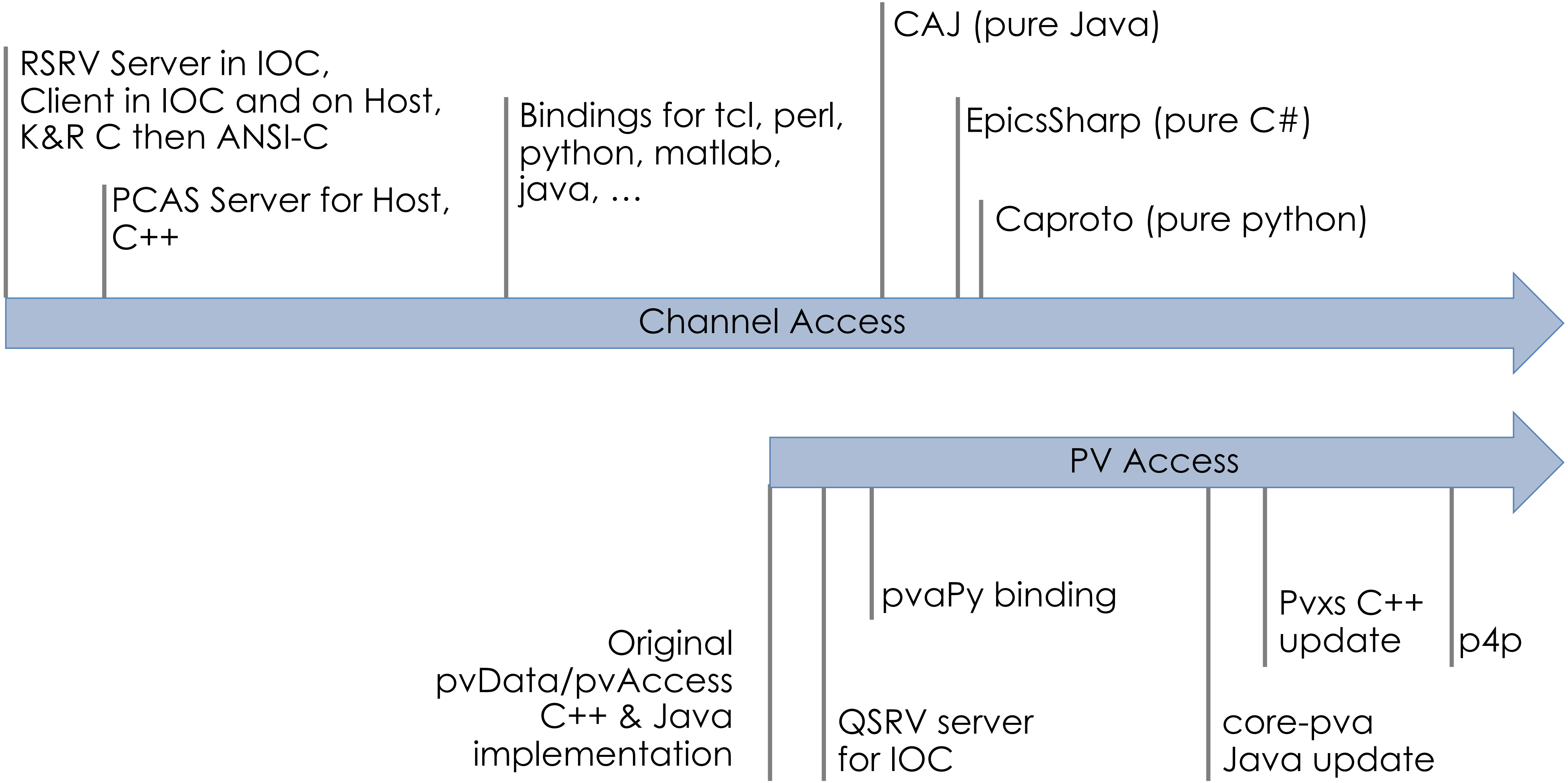
Channel Access

- 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

History



RSRV Server in IOC,
Client in IOC and on Host,
K&R C then ANSI-C

PCAS Server for Host,
C++

Bindings for tcl, perl,
python, matlab,
java, …

CAJ (pure Java)

EpicsSharp (pure C#)

Caproto (pure python)

Channel Access

PV Access

Original
pvData/pvAccess
C++ & Java
implementation

QSRV server
for IOC

pvaPy binding

core-pva
Java update

Pvxs C++
update

p4p

incomplete, not to scale, to be continued

First Glance

- softIoc vs. softIocPVA

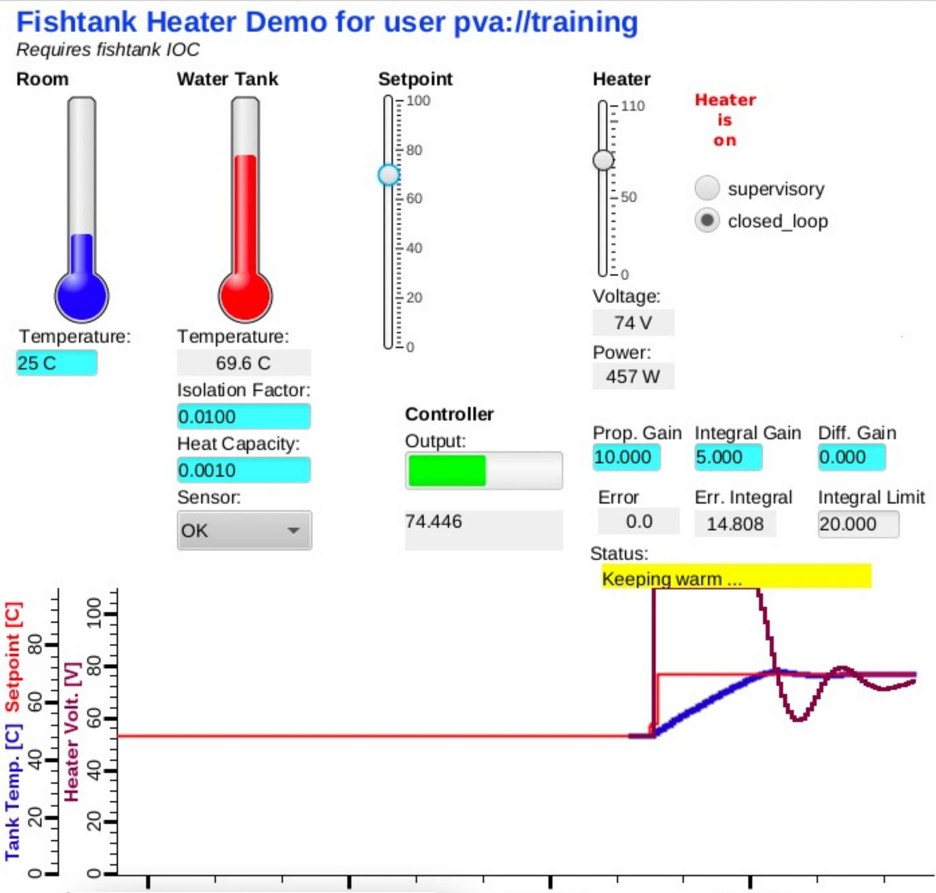
```
# Compare CA-only example:
cd /ics/examples/02_fishtank
cat st.cmd

# .. with this one:
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
```



Fishtank Heater Demo for user pva://training

Requires fishtank IOC

Room: Temperature: 25 C

Water Tank: Temperature: 69.6 C

Isolation Factor: 0.0100

Heat Capacity: 0.0010

Sensor: OK

Heater: Voltage: 74 V

Power: 457 W

Heater is on

○ supervisory

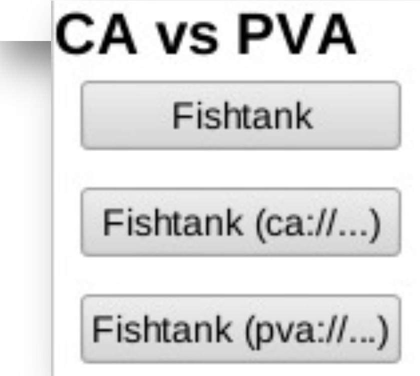
● closed_loop

Controller: Output: 74.446

Prop. Gain	Integral Gain	Diff. Gain
10.000	5.000	0.000

Error	Err. Integral	Integral Limit
0.0	14.808	20.000

Status: Keeping warm ...



CA vs PVA

PV Access

Similar to Channel Access

- Name search via
  - UDP Broadcast (IPv4), Multicast or Unicast (IPv4, IPv6), configured via EPICS_PVA_ADDR_LIST, EPICS_PVA_AUTO_ADDR_LIST
  - TCP search via EPICS_PVA_NAME_SERVERS
- Connection for data transfer via TCP
- Same “channel” or “PV” abstraction from ”record”

Get, put, monitor

- Plus an ‘RPC’ type operation

Channel Access vs. PV Access

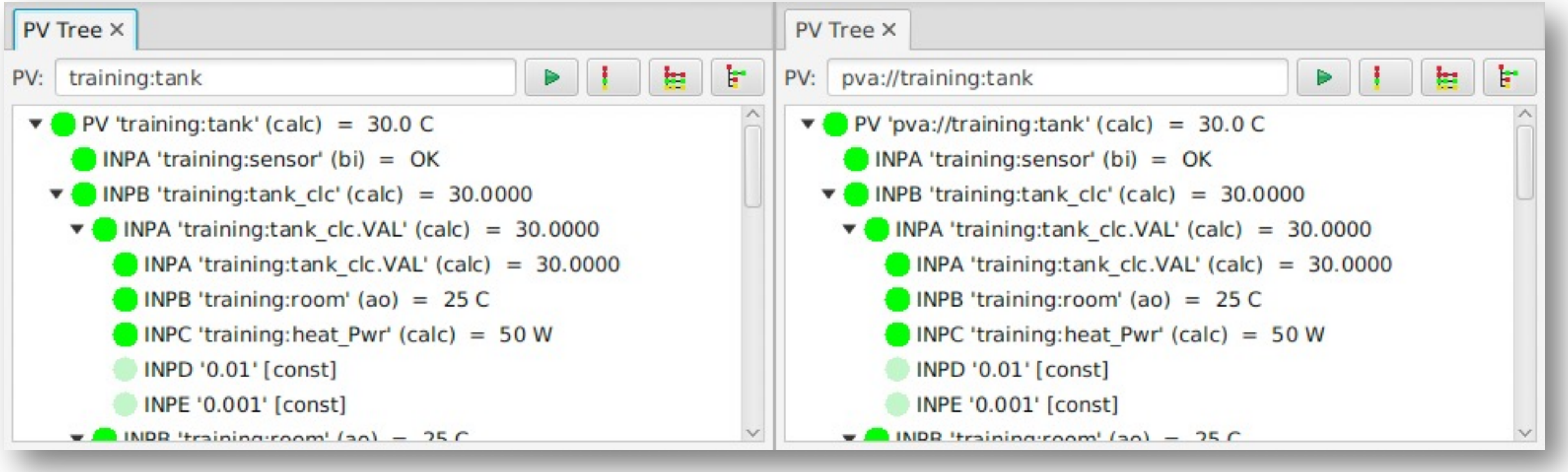
Similar command line tools:

```
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

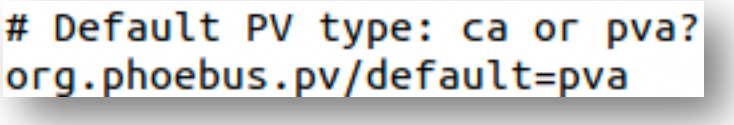
- Use pva://… prefix to select PV Access



```
training:tank

▼ ● PV 'training:tank' (calc)  =  30.0 C
    ● INPA 'training:sensor' (bi)  =  OK
  ▼ ● INPB 'training:tank_clc' (calc)  =  30.0000
    ▼ ● INPA 'training:tank_clc.VAL' (calc)  =  30.0000
        ● INPA 'training:tank_clc.VAL' (calc)  =  30.0000
        ● INPB 'training:room' (ao)  =  25 C
        ● INPC 'training:heat_Pwr' (calc)  =  50 W
        ● INPD '0.01' [const]
        ● INPE '0.001' [const]
    ▼ ● INPB 'training:room' (ao)  =  25 C
```

```
pva://training:tank

▼ ● PV 'pva://training:tank' (calc)  =  30.0 C
    ● INPA 'training:sensor' (bi)  =  OK
  ▼ ● INPB 'training:tank_clc' (calc)  =  30.0000
    ▼ ● INPA 'training:tank_clc.VAL' (calc)  =  30.0000
        ● INPA 'training:tank_clc.VAL' (calc)  =  30.0000
        ● INPB 'training:room' (ao)  =  25 C
        ● INPC 'training:heat_Pwr' (calc)  =  50 W
        ● INPD '0.01' [const]
        ● INPE '0.001' [const]
    ▼ ● INPB 'training:room' (ao)  =  25 C
```

- 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
```

How to add PVA to IOC?

a) Use `softIocPVA`

b) Start with `` `makeBaseApp.pl –t example` ``

c) Add to your own makeBaseApp-type Makefile:

```
myioc_DBD += PVAServerRegister.dbd
myioc_DBD += qsrv.dbd
myioc_LIBS  += qsrv
myioc_LIBS  += $(EPICS_BASE_PVA_CORE_LIBS)
```

Either way adds PVA and keeps CA

So it’s very similar, maybe more efficient…

What’s really new?

How about those custom structures?

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
…
```

Same record vs. PV abstraction.

There is no “NTRecord” or “NTCalcOut” that would fetch all fields of a record

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

Reminder: Channels/PVs vs. Records

- Records have fields
- Channels/PVs have properties

- IOC maps fields of records to properties of channel/PV
  - VAL → value
  - TIME → timestamp
  - STAT & SEVR → alarm
  - EGU → Units
  - PREC → display hints
  - HIGH → upper warning threshold

Nearly every record.FIELD can become a PV:

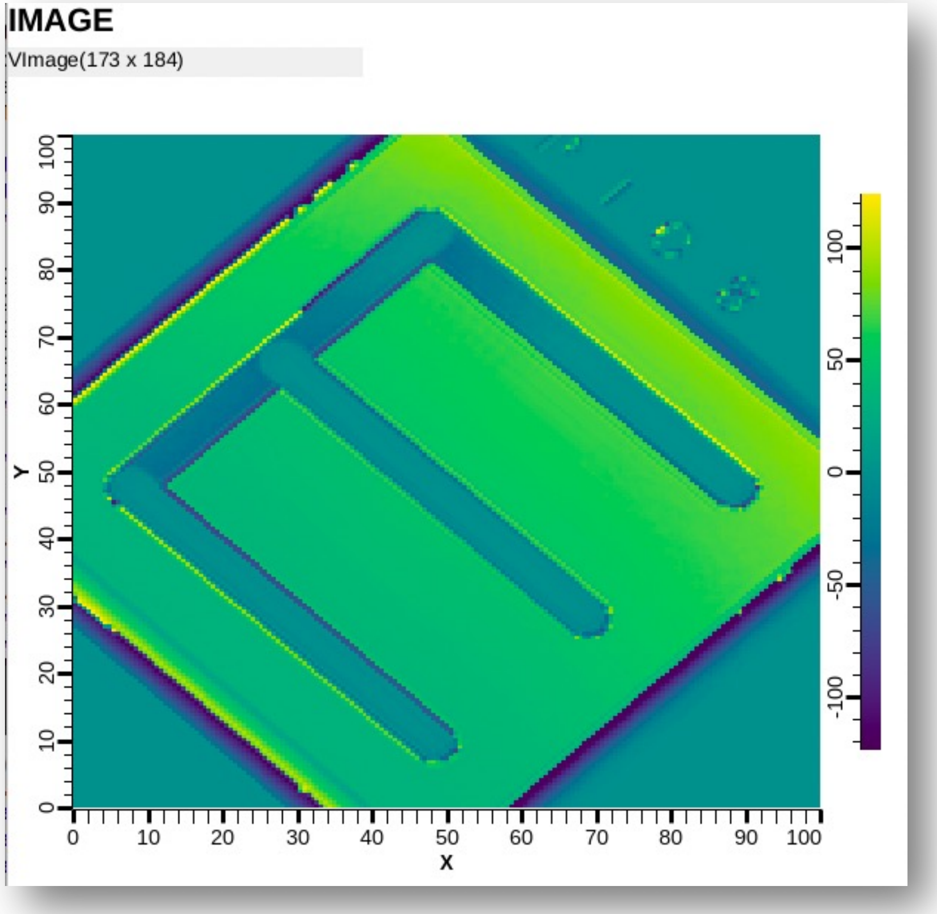
```
caget xxx
caget xxx.VAL
caget xxx.EGU
```

Detailed mapping of fields to channel/PV depend on record

Images: Normative type NTNDArray

- Standalone Demo Image

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



- CS-Studio Image widget

  Only needs pva://IMAGE

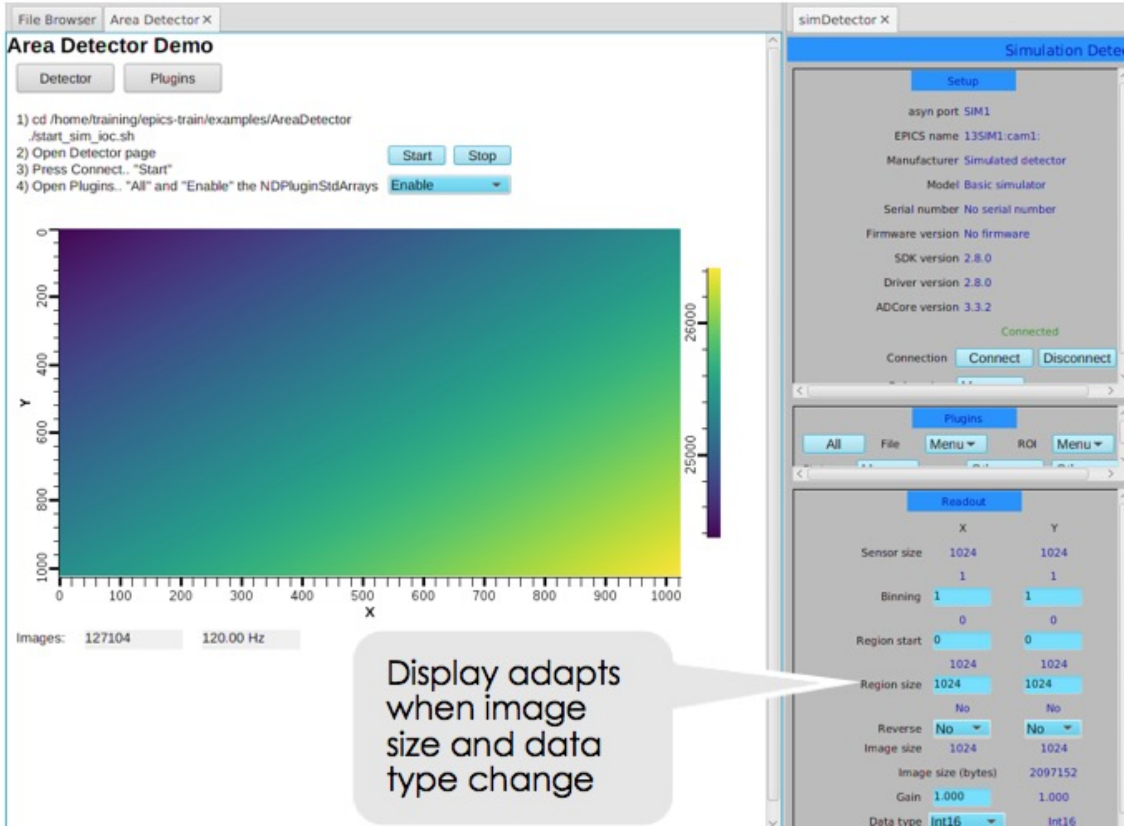
```
css -resource /ics/examples/24_pvaccess/PVA_IMAGE.bob
```

Images: Area Detector

- Area Detector with “Sim” and NDPluginPVA

```
cd /ics/examples/AreaDetector
./start_sim_ioc.sh
```

- CS-Studio: Image widget
  - Just “pva://….:Image
  - ”Limits from PV” option



Area Detector Demo

1) cd /home/training/epics-train/examples/AreaDetector
   ./start_sim_ioc.sh
2) Open Detector page
3) Press Connect.. "Start"
4) Open Plugins.. "All" and "Enable" the NDPluginStdArrays

Images: 127104 120.00 Hz

Simulation Dete

asyn port SIM1
EPICS name 13SIM1:cam1:
Manufacturer Simulated detector
Model Basic simulator
Serial number No serial number
Firmware version No firmware
SDK version 2.8.0
Driver version 2.8.0
ADCore version 3.3.2
Connected
Connection

	X	Y
Sensor size	1024	1024
	1	1
Binning	1	1
	0	0
Region start	0	0
	1024	1024
Region size	1024	1024
	No	No
Reverse	No	No
Image size	1024	1024
Image size (bytes)		2097152
Gain	1.000	1.000
Data type	Int16	Int16

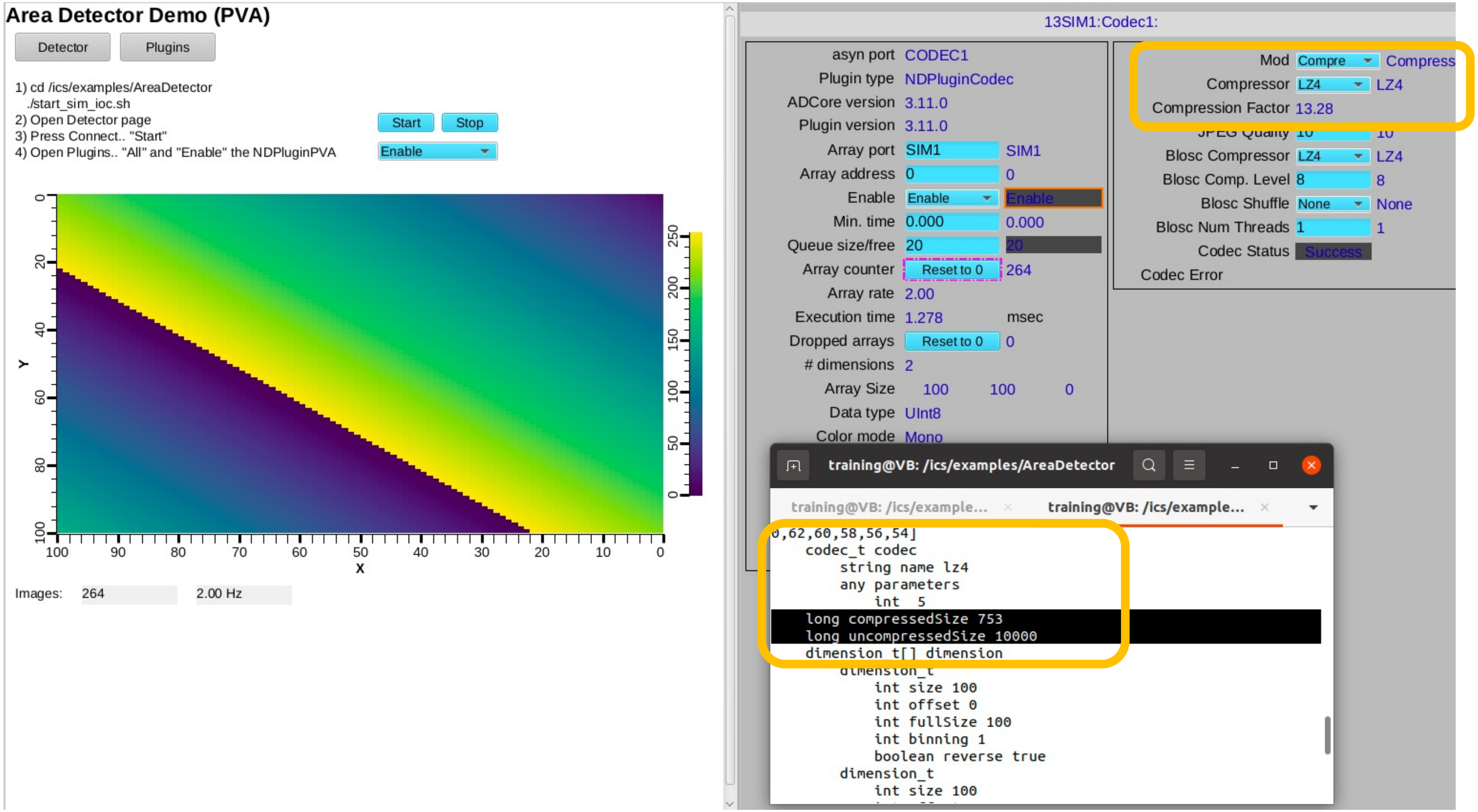
Display adapts when image size and data type change

```
css –resource /ics/examples/AreaDetector/0_AreaDetectorDemoPVA.bob
```

Area Detector Demo

- “Start” Sim Detector, “Enable” PVA Plugin
- Open display in editor to check Image widget config
  - Simple config via PV Name, maybe Limits from PV option
  - Data size, Color Mode, Unsigned are not needed!
- Open “Detector” page, change “Region Size”, “Reverse”, “Data Type”
  - Image updates accordingly!
- Open “Plugins”
  - Enable “NDPluginCodec”
    - Read SIM1
    - Select “LZ4” compressor, note “Compression Factor”
  - Set NDPluginPva to use Port “CODEC1”
  - Verify via `pvget 13SIM1:Pva1:Image`
  - Set Sim Data Type back to UInt8, then compress with “JPEG” of Quality 10
  - Compression support (LZ4 and JPEG)!

Image Compression



Area Detector Demo (PVA)

Detector | Plugins

1) cd /ics/examples/AreaDetector
./start_sim_ioc.sh
2) Open Detector page
3) Press Connect.. "Start"
4) Open Plugins.. "All" and "Enable" the NDPluginPVA

Start | Stop

Enable

Images: 264 2.00 Hz

13SIM1:Codec1:

asyn port CODEC1
Plugin type NDPluginCodec
ADCore version 3.11.0
Plugin version 3.11.0
Array port SIM1 SIM1
Array address 0 0
Enable Enable Enable
Min. time 0.000 0.000
Queue size/free 20 20
Array counter Reset to 0 264
Array rate 2.00
Execution time 1.278 msec
Dropped arrays Reset to 0 0
# dimensions 2
Array Size 100 100 0
Data type UInt8
Color mode Mono

Mod Compre Compress
Compressor LZ4 LZ4
Compression Factor 13.28
JPEG Quality 10 10
Blosc Compressor LZ4 LZ4
Blosc Comp. Level 8 8
Blosc Shuffle None None
Blosc Num Threads 1 1
Codec Status Success
Codec Error

training@VB: /ics/examples/AreaDetector

training@VB: /ics/example... | training@VB: /ics/example...

```
0,62,60,58,56,54]
    codec_t codec
        string name lz4
        any parameters
            int  5
    long compressedSize 753
    long uncompressedSize 10000
    dimension_t[] dimension
        dimension_t
            int size 100
            int offset 0
            int fullSize 100
            int binning 1
            boolean reverse true
        dimension_t
            int size 100
```

Custom PV Data

SNS Beam Lines started to use this in ~2014

```
cd /ics/examples/24_pvaccess
./start_neutrondemo -h
./start_neutrondemo -d 0.5 -r -m
```

```
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*

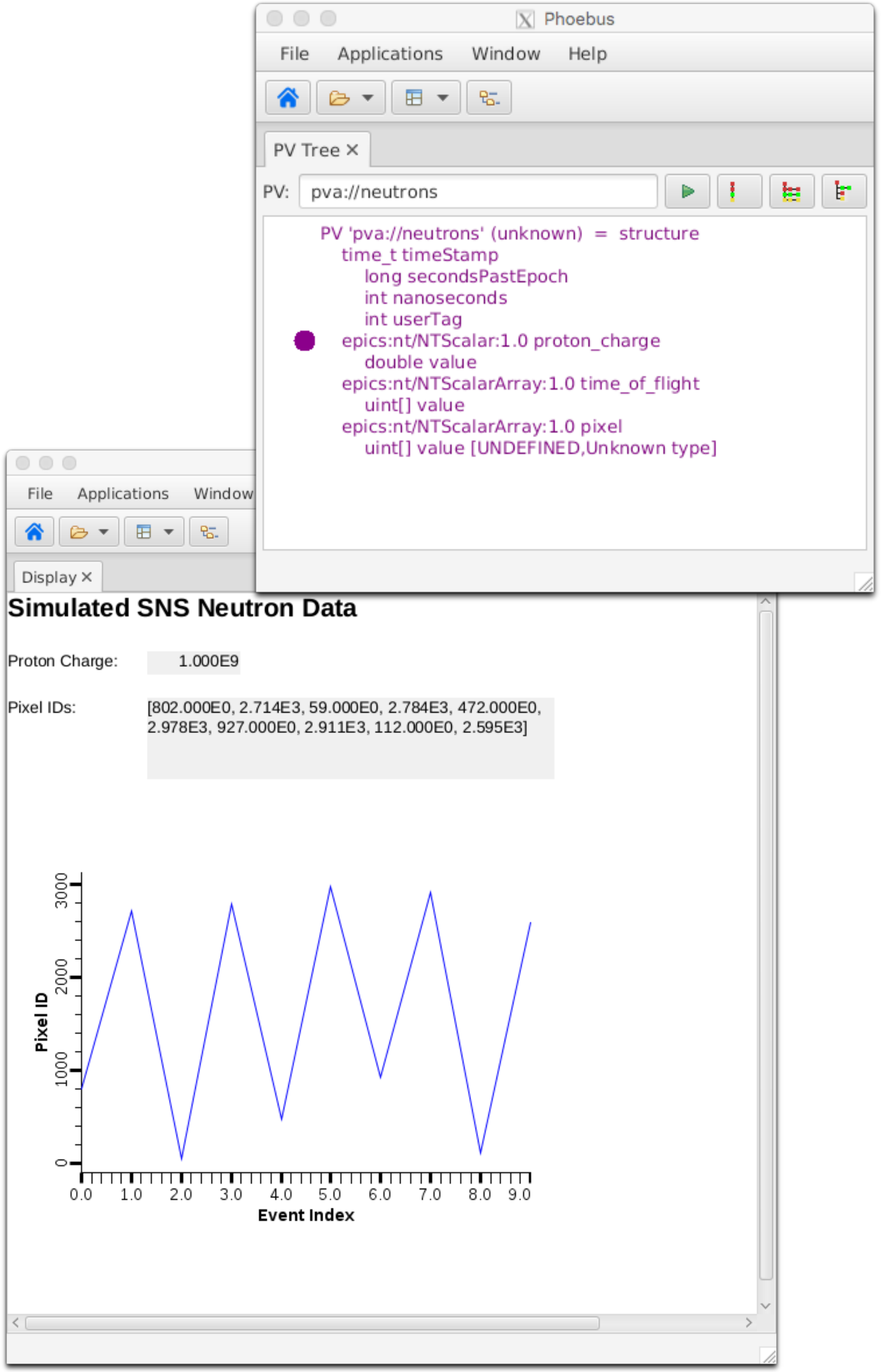
```
pva://neutrons
```

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

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

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

```
css –resource /ics/examples/24_pvaccess/PVA_Neutrons.bob
```



pva://neutrons

```
PV 'pva://neutrons' (unknown)  =  structure
  time_t timeStamp
    long secondsPastEpoch
    int nanoseconds
    int userTag
● epics:nt/NTScalar:1.0 proton_charge
    double value
  epics:nt/NTScalarArray:1.0 time_of_flight
    uint[] value
  epics:nt/NTScalarArray:1.0 pixel
    uint[] value [UNDEFINED,Unknown type]
```

Simulated SNS Neutron Data

Proton Charge: 1.000E9

Pixel IDs: [802.000E0, 2.714E3, 59.000E0, 2.784E3, 472.000E0, 2.978E3, 927.000E0, 2.911E3, 112.000E0, 2.595E3]

History: Two compatible Implementations

Initial Implementation (Since ~2014)

C++: pvDataCPP, pvAccessCPP, …

Java: pvDataJava, pvAccessJava, …

Python: pvaPy

Gateway: pva2pva

✓ Included in EPICS 7: softIocPVA, 'QSRV', pvget/put/info/monitor

✓ Used in successful operation

- Same API for C++ & Java:
  Lowest common denominator, missing language advantages.

- Bugfixes, but no additions.

Updated Implementation (~2020)

C++: PVXS

Java: core-pva

Python: p4p

Gateway: p4p gateway

✓ APIs take advantage of each language

✓ Gateway's "fair" scheduling helps with arrays; known UDP port allows use via firewalls

✓ Active Development

  ✓ IPv6 support

  ✓ EPICS_PVA_NAME_SERVERS for TCP-only usage

- Not in EPICS base, yet.

Same Protocol!

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
```

```
May have to use ‘python3’ since just ‘python’ is older version 2.x
```

Custom PV Data from IOC Records

`makeBaseApp.pl –t example` includes “group”,
see /ics/examples/07_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(x`^{2}`+y`^{2}`)==1`

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

State of PV Access by late 2022

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
  - 'Description'
  - Full 'units'
  - Full 64 bits for 'int64in', 'int64out'
  - No enum state limit
  - Supports changing metadata
- CS-Studio client
- Gateway

To do

- IOC links
  - Default to CA.

    Initial support for `field(INP, {pva:{pv:"tgt"}})`
  - Channel Access Get/put callback → ??
- How to best combine data from records into custom PVA data?

Summary: PV Access is ..

- Alternative to Channel Access
  - Both can be used in parallel

- Similar, but supports custom data types
  - Already useful for images and site-specific cases

- Since EPICS 7 included in base IOC
  - Details of ‘group’, ‘field(…)’ access still evolving