EPICS 7 =

EPICS ‘base’
- Records,
- Device Support,
- Channel Access,
- softIoc

Almost like R3.13 from 1994

EPICS ‘V4’
- PVAccess,
- softIocPVA

Started ~2010

Available since Dec. 2017
EPICS 7 ?

Revolutionizes EPICS
Everything is
- Easier
- Faster
- More colorful
- Service-oriented

Kills EPICS as you know it
- Services replace IOCs
- Channel Access clients no longer connect
- Breaks your device support
- Needs more CPU & Memory
1. Use EPICS as before
   - No need to change anything
   - Look at ‘RELEASE.local’ mechanism
   - Support for 64bit numbers, SMP, locking tweaks

2. Start to use PV Access
   - Images
   - Custom structures

3. Transition everything to PV Access
   - .. Once there’s a PVA Gateway, access security, everything “works”
EPICS Base

- Records, device support, databases, sequences, ... as before
- Modules encouraged to use RELEASE.local

<table>
<thead>
<tr>
<th>XXX/configure/RELEASE</th>
<th>XXX/configure/RELEASE</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ASYN=/path/to/asyn1-2-3</code></td>
<td><code>-include $(TOP)/../RELEASE.local</code></td>
</tr>
<tr>
<td><code>EPICS_BASE=...</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>YYY/configure/RELEASE</th>
<th>YYY/configure/RELEASE</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AUTOSAVE=/path/to/auto1-2-3</code></td>
<td><code>-include $(TOP)/../RELEASE.local</code></td>
</tr>
<tr>
<td><code>EPICS_BASE=...</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RELEASE.local</th>
<th>RELEASE.local</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ASYN=/path/to/asyn1-2-3</code></td>
<td><code>AUTOSAVE=/path/to/auto1-2-3</code></td>
</tr>
<tr>
<td><code>...</code></td>
<td><code>EPICS_BASE=...</code></td>
</tr>
</tbody>
</table>
RELEASE.local in Training Setup

cd ~/epics-train/tools
ls -d base* seq* asyn*
cat RELEASE.local
cat seq*/configure/RELEASE
PV Access

• How does it differ from Channel Access?

• How do I...
  ... add PVA to an IOC?
  ... ‘caget’ with PVA?
  ... use PVA in UI tools (Operator Displays, ..)?

• Area Detector: Any advantage using PVA?

• Custom Data: Any advantage using PVA?

• Some Python Examples
History

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
Review Channel Access

cd ~/epics-train/examples/first_steps
cat first.db
softIoc -m S=training -d first.db

echo $EPICS_CA_ADDR_LIST
caget training:random
camonitor training:random

cainfo training:random
caget -h
caget -d DBR_CTRL_DOUBLE training:random

cainfo training:random.SCAN
caget -d DBR_CTRL_ENUM training:random.SCAN
How Clients find Channels

Wants to read "PS1:Voltage"

Client → Client → Client → Client → Client → Client

Channel Access

IOC → IOC → IOC → IOC

Meter → Power Supply → Camera

Has PV "PS1:Voltage"
Important Environment Variables

- **EPICS_CA_ADDR_LIST**
  - Determines where to search
  - Is a list (separated by spaces)
    - “123.45.1.255 123.45.2.14 123.45.2.108”
  - Default is broadcast addresses of all interfaces on the host
    - Works when servers are on same subnet as Clients
  - Broadcast address
    - Goes to all servers on a subnet
    - Example: 123.45.1.255
    - Use `ifconfig -a` on UNIX to find it

- **EPICS_CA_AUTO_ADDR_LIST**
  - YES: Include default addresses above in searches
  - NO: Do not search on default addresses
  - If you set EPICS_CA_ADDR_LIST, usually set this to NO
Channel Properties

• Each channel comes with properties:
  – Value
    • of type string or double or int or …
    • Scalar or array
  – Time stamp
    • Up to nanosecond precision
  – Severity code
    • OK, MINOR, MAJOR, or INVALID
  – Status code to qualify the severity
    • OK, READ error, WRITE error, at HIGH limit, …
  – units, suggested display range, control limits, alarm limits.
Client interface to properties

• The available properties are fixed.
  – One cannot add a new 'color' property.

• The request types are fixed.
  – "DBR_..." types.
  – Available:
    a) Just value.
    b) Value with status and severity.
    c) Value with status, severity and time stamp.
    d) Almost Everything: Value, units, status, limits, ... but time
  – Not available:
    • Custom combinations like value, precision, units.
  – See `caget -h`
Example: AI record "fred"

- PV "fred" or "fred.VAL"
  - value property of channel = VAL field of record.
    - Type double, one element (scalar).
  - time property = TIME field
  - status = STAT
  - Severity = SEVR
  - units = EGU
  - Precision = PREC
  - display limit low, high = LOPR, HOPR
  - control limit low, high = LOPR, HOPR
  - alarm limits = LOLO, LOW, HIGH, HIHI

- Makes a lot of sense.
  - GUI can display the value together with units, formatted according to the precision, as e.g. "12.37 volts".
Example: AI record "fred"

- PV "fred.SCAN"
  - value property of channel = SCAN field of record.
  - Type enumerated, values: "Passive", "1 second", ...
  - time property = TIME field?
  - status = STAT?
  - Severity = SEVR?
  - control limit low, high = 0, ??
cd epics-train/examples
cat first.db
softIocPVA -m S=training -d first.db

echo $EPICS_PVA_ADDR_LIST
pvget training:random
pvmonitor training:random
pvget -m training:random

pvinfo training:random
pvinfo training:random.SCAN
Channel Access vs. PV Access

Similar command line tools:

- `start_iocExample`
- `cainfo training:ail`
- `caget training:ail`
- `camonitor training:ail`
- `caget -d CTRL_DOUBLE training:ai`
- `caget training:ail.SCAN`
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 ‘GetPut’, 'PutGet', ‘RPC’ type operations

Arbitrary PV Data structures instead of DBR_. types
Arbitrary Data:  Great, but then what?

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

• Channel Access

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

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

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

• PV Access

```c
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 in UI Tools
Getting Started with CSS

Start `css`
Probe

• Open Menu Applications, Display, Probe

• Enter PV name “sim://sine”

• Open another Probe for “training:random” (or some other PV from your IOC)

• Close Probe

• Open it again

• Note previously used PVs in history as you enter new PV

• Right-click on the “Probe” tab, Select “Split Horizontally”, and move one of the probes to...
Data Browser

- Menu Applications, Display, Data Browser
- Right-click on plot, Add PV, “sim://sine”
- Wait a little, press Stagger button, then zoom and select a region on the time axis
PV Tree

- Menu Applications, Display, PV Tree
- Enter a PV from an IOC, like “training:random”
CSS PV Exchange

- PV in *any* CSS Tool
  - Context Menu ➔ Select other PV Tool

Try:

Right-click on item in PV Tree, select Data Browser
CS-Studio: Use ‘pva://…’

For now, just “pvname” is same as “ca://pvname”.

“pva://” selects PV Access. Eventually, that could become the default.
Create New Display

Menu Applications, Display, New Display
- Enter a name with .bob file extension

Main Editor Area
Select Widgets
Move, resize widgets
Ctrl-C, V, X to copy, paste, delete (⌘ on Mac)

Save & Execute the Display

Property Panel
Edit properties of selected widgets
Editing a Display

Selecting Widgets
a) Click single widget
b) Ctrl-click to add widget (⌘ on Mac)
c) Drag ‘rubberband’ around widgets
d) Click or Ctrl/⌘ click in widget list

Quick Edit
Double-click widget to
a) Edit text of Label
b) Edit PV of widgets that use a PV

Widget Palette
Drag widget into editor
- or -
1) Select Widget Type
2) Draw rectangular area in display
Extend the First Display

- Drag a “Text Update” from the palette
  - Enter PV name “sim://ramp(1, 10, 1)”. Note PV name auto-completion popup.

- Add “Boolean Button”
  - PV name “loc://test”

- Add “LED”
  - PV name “loc://test”. Note name in PV History.

- Execute the display
  - Toolbar Button or Context Menu
Browse the Display Examples

Training setup: Open /home/training/epics-train/examples/Display Builder/01_main.bob

- Fresh CS-Studio Setup: Applications, Display, Examples, Install Example Displays

Try all of them...

Context Menu: Open in Editor to inspect how it's done
How is PV Access different?

Images!

```
start_imagedemo
pvinfo IMAGE
# CSS displays/PVA_Image.bob
```
Area Detector

Disclaimer:

This will only scratch the surface.

EPICS web site has several days of training material if you are serious about using the A.D.
Area Detector

- EPICS framework for image manipulation
- Detectors/Cameras
  - Cheap “Web Cam”
  - $$$ high speed, high res.
  - Neutron, X-Ray detectors
- Plugins collection
  - ROI
  - Transform
  - ColorConvert
  - Etc.
ADSImDetector

• Simulated images
  cd ~/epics-train/examples/AreaDetector ./start_sim_ioc.sh

• Open the AreaDetectorDemo.bob
  – On “Detector” page, “Start” the SIM1 detector

By itself, this creates an Area Detector port “SIM1”. To see it, need to publish via CA or PVA
NDPluginStdArrays

- Serves image as Channel Access waveform
- On Detector, Plugins, All, find NDPluginStdArrays
  - Port = “SIM1”
  - Enable

- AreaDetectorDemo.bob shows image
  - PV: 13SIM1:image1:ArrayData
  - Width x Height: 1024 x 1024
  - Unsigned
NDPluginOverlay

• Adds rectangles, text etc. to image

• On Detector, Plugins, All, find NDPluginOverlay “OVER1”
  – Set its Port to “SIM1”, Enable
  – Change NDPluginStdArrays’s Port to “OVER1”

- Press “More”, select first of the “Individual Overlays”
NDPluginOverlay.. Overlay #1

Set Use: Yes, Shape: Rectangle, set X and Y as shown
What we did

Area Detector IOC

Plugins

Image1

OVER1

Driver

SIM1

OVER1 offers 8 overlays:
1) Rectangle
2) Text “Hello”
3) ...
Many More Plugins...

• Process
  – Background subtraction, clipping, recursive averaging over N images, ..

• Saving images in various formats
  – Adding data from PVs as “Attributes”
  – PNG, JPEG, TIFF, HDF5, ...

• Serving NDArray via PVA
  – For displays: No need to configure size, data type, ...
  – For ADPvAccess Driver: Process data on different hosts
NDPluginPVA – Serve PVA ‘Image’

• In Plugins, “PVA1”
  – Set its Port to “SIM1” or “OVER1”, Enable

• PVAccess Tests
  – pvinfo 13SIM1:Pva1:Image
  – pvget -r 'dimension' 13SIM1:Pva1:Image

• In Display
  – Use “Image” widget
  – Set PV
  – No need to configure data size, data type
NDPluginPVA – Serve PVA ‘Image’

Display adapts to image size
How is PV Access different?

Custom Data!
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`
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:circle:x training:circle:y separate x, y updates
  pvget -m training:circle will always see sqrt(x²+y²)==1

  cd ~/epics-train/examples/python
  python circle.py
Python
PV Access and Python

start_iocExample

Basic ‘get’

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

‘monitor’

   python example2.py
Custom PV Data in Python Client

Python receives data as dictionary, access to any element

```python
neutrons.py
```
Custom PV Data from Python Server

Surprisingly easy:

```
# Server
python server.py

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

```
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”

→ Impractical to replace all regular IOCs with python, but useful when custom data is needed
Ongoing Work

• PVA Gateway

• Access Security

• Normative Type details: ‘format’, precision, ...

• Database: Support PVA links.
  field(INP, “pva://other_record”)
EPICS 7

EPICS ‘base’
Records, Device Support, Channel Access, softloc

EPICS ‘V4’
PVAccess, softlocPVA

• No need to worry about existing R3.x setups
• You may start using PVAccess
  ✓ Images
  ✓ Custom Data
• Good Python support
• CS-Studio is one of the early ‘bilingual’ tools