EPICS 7 and CS-Studio Seminar

Date: Friday May 19, 2017
Provided by: Ralph Lange ralph.lange@gmx.de
Kay Kasemir kasemirk@ornl.gov
Megan Grodowitz grodowitzml@ornl.gov

Overview

This will be an interactive introduction to EPICS 7, i.e. the combination of EPICS V4 with EPICS base, and CS-Studio.

Preparation

Please prepare a laptop on which you installed a virtual machine runtime.

- The laptop must run a 64 bit OS (Windows, Linux, Mac OS X).
  In the BIOS of your computer, you may have to enable the “VT-X” or “AMD-v” option to support 64 bit guest operating systems inside Virtualbox.
- Download and install the 64 bit version of Virtualbox for your OS from https://www.virtualbox.org/wiki/Downloads

During the preceding days of the EPICS meeting we will circulate USB sticks with a VM image. It can also be downloaded from http://ics-web.sns.ornl.gov/kasemir/Training.ova.

To import into VirtualBox, invoke the menu File, Import Appliance…
Select the file `Training.ova`, press ‘Continue’.
On the next page, you can adjust the number of allocated CPUs and RAM. Try 4 CPUs, 4GB.

If you are on Mac OS X, after the import, select the imported VM, open the menu Machine/Settings, and in the Display section, enable “Unscaled HiDPI Output”.

Start the VM.
After about 10 seconds, Linux should start up and automatically log you into the account for the user `training` (with password `$training` in case you should later need that).
From the VirtualBox window menu, select View, Auto-resize Guest Display, then full-screen mode (revert via Apple Command-F or right-Ctrl-F).

To close the VM, use the drop-down at the right edge of the desktop menu bar.

If you want to update the “VirtualBox Guest Additions” (not mandatory if your version of VirtualBox is close to the one that was used to create the appliance), insert the CD image into the drive using the “Device” menu, then let auto-run do its job. Use the mentioned password for authorization..

Review VM

Basic tools included in the Linux image:

- Terminal: Open via Applications, Favourites or from context menu of desktop
- Editors: vi, gedit, css

In your home directory, find an `epics-train` folder with two sub folders:

- epics-train/tools: Contains EPICS base, V4, Java, CS-Studio.
- epics-train/examples: Contains example IOCs, displays, python scripts.

Each has a `recipe.txt` that explains how these have been created.
Your ~/.bashrc sources epics-train/settings.sh to configure the shell environment.
The source code for all is in the Github repository https://github.com/kasemir/epics-train.

Review of EPICS IOC and Channel Access *(Ralph)*

Check examples/CAApp and examples/iocBoot/iocCA that were created by

```
makeBaseApp.pl -t example
```

Start the IOC in a terminal window:

```
cd ~/epics-train/examples/iocBoot/iocCA
./st.cmd
```

In another terminal window, run some basic Channel Access commands:

```
cainfo training:calcExample            (connection and type info)
caget training:calcExample             (get the value)
caget -a training:calcExample          (same, with full output)
camonitor training:calcExample         (subscribe to changes)
```

With Channel Access, channels can be accessed via a predefined list of DBR_* types:

```
caget -h
caget -d CTRL_DOUBLE training:calcExample
```

Records and their Fields vs. Channels and their Properties

- The Record “training:calcExample” has Fields VAL, TIME, EGU, STAT, SEVR, ..
- The Channel “training:calcExample”, read as a CTRL_DOUBLE, has Properties for the value, timestamp, units, alarm status, severity, ..
  Q: How are the fields of the record mapped to the properties of the channel?
  Q: What are the properties of the channel “training:calcExample”, read as a STRING?

Q: What are the properties of the channel “training:calcExample”, read as a CTRL_ENUM?

There is no Channel to capture the complete set of fields for each record, but one can create a channel for each individual field of a record.

- The channel “training:calcExample.EGU”, read as a CTRL_STRING, has properties for the value, timestamp, units, alarm status, severity, ..
  Q: How are the fields of the record mapped to the properties of this channel?

pvAccess *(Ralph)*

examples/PVAApp was manually created as a minimal IOC with just one database, but adding pvaSrv, so all channels can be accessed via the new pvAccess protocol in addition to Channel Access.
Check examples/PVAApp/src/Makefile and examples/iocBoot/iocPVA/st.cmd, comments indicate the additions for the PVAccess server.

Start IOC:

```
cd ~/epics-train/examples/iocBoot/iocPVA
./st.cmd
```

While pvAccess is similar to Channel Access, you can now list available servers:

```
pvlist
pvlist 0x1232134234  # Using the number shown by previous command
```

You can read respectively monitor PVs via pvAccess:

```
cainfo training:ramp
pvinfo training:ramp
caget training:ramp
pvget training:ramp
pvget -m training:ramp
```

Pvget can actually fall back to Channel Access as a provider

```
pvget -p ca training:ramp training:calcExample
```

Examples for -r “field()”.

```
pvget -r “field()” training:ramp
pvget -r “field(value)” training:ramp
pvget -m -r “field(value, timeStamp, display.units)” training:ramp
pvget -t -m -r “field(value, timeStamp, display.units)” training:ramp
```

- pvAccess is the name of the new network protocol (compare Channel Access).
- It transfers pvData (compare DBR_* structures).
- The pvData types we’ve seen so far are “Normative Types”, representing legacy DBR_* types.

- pvAccess only transfers changed data sections. You can effectively subscribe to the complete structure (compare DBR_CTRL_DOUBLE) without burdening the network by always transferring the fields that didn't change (compare with Channel Access clients that tend to read DBR_CTRL_native_type once, then monitor DBR_STS_native_type, maybe also use DBE_PROPERTY).
- Strings are no longer limited to 40 chars.
- Enums are no longer limited to 16 values.
- Arrays are no longer limited to a certain upper limit. Clear distinction between scalar and array of size 1.
- Custom Data Types: You can define your own data types for specialized servers and clients. We'll see an example later.
- In this example, we added "pvaSrv" to the IOC. "pvaSrv" was the initial development for serving IOC records via pvAccess.
  "QSRV" is an updated version with better performance and support for "groups".
- "groups": Read or write record A.VAL and record B.VAL as an atomic operation (details outside of this introduction).
- pvAccess has 'get', 'put', 'monitor' as Channel Access.
  In addition there is 'putget' and an RPC type call for interacting with services.
  Putget writes a data structure and returns the same data type.
  RPC call writes one data structure and then returns a different data type.
  (details again beyond this introduction).

For more, see http://epics-pvdata.sourceforge.net/informative/training/introduction.slides.html

CS-Studio: First Steps *(Kay)*

Start "css".

- Workspace Selection Dialog: This is where window locations etc. are stored.
  Typically, just press "OK".
  To run multiple instances of CS-Studio as the same user on the same computer, enter different workspace locations for each instance.
- Navigator: Displays files.
  Training setup uses '-share_link' option to add "examples".
- Open 'Probe' from toolbar, enter for example "training:ramp"
- Open menu CS-Studio, Diagnostic Tools, EPICS PV Tree. Try same PV
- Move Probe and PV Tree by dragging their title tab into different sections of the window.
  Note faint gray rectangles that indicate new locations in window.
  Drag outside of window.
- Open 'Data Browser' from toolbar.
  Use context menu of plot, "Add PV", to add same PV
- Close Data Browser. Use context menu of EPICS PV Tree to open it for the same PV.

CS-Studio and pvAccess (Kay)

CS-Studio currently defaults to using Channel Access.

Enter PV names with “pva://” prefix to select pvAccess.

- Try ‘Probe’ with “pva://training:ramp”
  - Behaves just like the Channel Access case
- Try ‘EPICS PV Tree’ with “pva://training:ramp”
  - You only see the value, not the database tree.
    While pvaSrv will serve “pva://training:ramp”, is currently doesn’t serve “pva://training:ramp.RTYP”, “pva://training:ramp.INPA” etc.
    Check with ‘Probe’, or try pvget on the command line!
    Remember records+fields vs. channels+properties.
    Channel access serves each field of each record.
    pvaSrv serves normative pvData for each record, not for each field.

Elements of the pvData structure can be read via pva://channel/structure/element.

- Try ‘EPICS PV Tree’ with the following:
  - “pva://training:ramp/display”
  - “pva://training:ramp/display/limitLow”

Basics of Perspectives (Kay)

Data Browser has several auxiliary panels:

- From context menu of Data Browser plot, open Properties Panel.
- From context menu of Data Browser plot, open Data Export Panel.
- Close Properties and Data Export panel

A “Perspective” is an arrangement of such auxiliary panels:

- From context menu of Data Browser plot, select Data Browser Perspective
- Close the properties panel, then “Reset” the perspective via its context menu.

The “Saved Perspectives” section will show how to save display layouts.

CS-Studio Display Builder *(Kay)*

- Install and view example displays:
  - CS-Studio, Utilities, Install Samples: Display Builder
  - Window, Open Perspective: Display Runtime
  - File, Top Displays, Main
- Create CS-Studio display builder panel for ca:// and pva://
  - Editor Perspective
  - Create new display
  - Label
  - TextUpdate
  - Execute
  - Context menu: Open PV Tree from Text Update

- Re-opening file in in editor or runtime
- Button to write value
- Button to open new display
- Rules
- Fishtank example
- Classes
- Handling of existing *.opi files

Saved Perspectives (Kay)

- Eclipse, the basis of CS-Studio, distinguishes between
    - Editors: Text editor, Data Browser, Display Editor
    - Views: Probe, PV Tree, Properties, <u>Display Runtime</u>
- Perspective = Arrangement of Views
    - Can be saved under a name
    - Closed
    - Re-opened
    - Restored to saved state

This allows saving different layouts for end users.
Typically, you start with a new workspace, and then repeat the following to create for example a “Main” and “Detail” perspective

1. Open the “Display Runtime” perspective, reset it
2. Open and arrange desired displays
3. Perspective “Save As”

Finally, select the desired initial perspective, for example “Main”, and close CS-Studio.
The workspace now contains a *.xmi file that contains the perspective information:

```
$ find ~/CSS-Workspaces/Default -name ‘*.xmi’
```

Create a copy of that file as ~/default.xmi:

```
$ cp CSS-Workspaces/Default/.metadata/.plugins/org.eclipse.e4.workbench/workbench.xmi ~/default.xmi
```

The ‘css’ start script will detect the presence of ~/default.xml and now always start CSS with the same initial layout.

Images *(Kay)*

Run example pva image server:

```
cd ~/epics-train/tools/EPICSV4Sandbox/ntndarrayServer/bin/linux-x86_64
./ntndarrayServerMain IMAGE
```

The ntndarrayServer source code is an example for a custom PVA server implemented in C++.

Details are beyond this seminar. If interested, please refer to EPICS V4 documentation, http://epics-pvdata.sourceforge.net .

Basic components of a PVA server:

- pvData 'fields': Description of a data structure, as e.g. seen by `pvinfo`.
- pvData 'pv': Data structure with values, as read by e.g. `pvget`.
- pvAccess 'channel': Named network channel that serves pvData to clients.
- pvDatabase: Helper for defining 'records' based on 'pvs'. Handles serving changes to the data of the record via pvAccess. Your application code needs to update the data in the record.

This PVA server serves a data structure of type NTNDArray, a normative type based on the data handled by the Area Detector and commonly used for images.

Explore image data:

```
pvinfo IMAGE
pvget IMAGE
pvget -r "field(dimension)" IMAGE
```

Create CS-Studio display builder panel for image:
Add an Image widget (listed in the Plots category), set its PV name to "pva://IMAGE".

Custom data structures *(Kay)*

Start SNS beam line demo data server:

```
cd ~/epics-train/tools/EPICSV4Sandbox/neutronsDemoServer/iocsBoot/neutrons
./st.cmd
```

This starts an IOC with some plain records and a pvAccess server that serves a custom data type to simulate SNS neutron data.

Alternatively, the custom PVA server can also be started as a standalone program, which offers command line options to configure the simulated data:

```
cd ~/epics-train/tools/EPICSV4Sandbox/neutronsDemoServer/bin/linux-x86_64
./neutronServerMain -h
./neutronServerMain -d 0.5 -m -e 1000 -r
```

Explore neutron data:

```
pvinfo neutrons
pvget -m neutrons
```

This type of data is not meant to be displayed "as is". It is instead read by custom client software which then for example creates a histogram. During initial evaluation of pvAccess for SNS neutron data, the neutronDemoClient program was used to test if the received series of monitors was complete, without gaps, for varying sizes of the data.

You can still create CS-Studio display builder panel for elements of the custom structure: pva://neutrons/proton_charge, pva://neutrons/pixel.

Python *(Kay)*

The first example reads a PV via pvAccess from python.
You can execute the example from the command line:

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

You can also execute it within CS-Studio:
Double-click the file in the Navigator to open it in the Python Editor.
Then use either the context menu to 'Run As, Python',
or press F2[*] to first open a console and later to execute the file line by line.
[*]: F2 is a shortcut to "Execute line in console" from the PyDev editor. If it is not recognized in the CentOS training VM, invoke the menu Edit, Preferences, search for "keys", then "execute", and change the "When" scope from "PyDev editor scope" to just "Editing Text".

Read and execute the other examples in epics-train/examples/python to learn

- How to subscribe to monitors
- How to fall back to Channel Access
- How to fetch elements of a custom structure.

The server.py requires the very latest snapshot of pvaPy:

```
cd ~/epics-train/examples/python
git pull
source updatePvaPy.sh          (takes about 5 minutes when called 1st time)
python server.py
# Try `pvinfo pair`, `pvget -m pair`, and CS-Studio displays/PVA_Server.bob
```

pvaPy also supports the RPC capability of pvAccess as both a server and client. Details are beyond this introduction, but you may check the following examples:

```
cd epics-train/tools/EPICS-CPP-4.6.0/pvaPy/examples
# In one terminal
python testRpcServer.py
# In other terminal
python test RpcClient.py
```

Conclusion

- IOCs remain fundamentally the same
- The new pvAccess protocol offers advantages over Channel Access
- Clients like CS-Studio will allow a smooth transition from CA to PVA

We hope you enjoyed this introduction to EPICS 7 and CS-Studio!