Controls User & Troubleshooting Guide

Controls Seminar Series

EPICS - IOCmanager: IOC health monitoring & first level trouble shooting

If a PV gives you trouble, a first good step is to see if the IOC is up stably and possibly reboot it. For that you will use the IOC manager which has a user guide here and is started via **iocmanager**. This guides describes how to start it and how to it works. More detailed information for commonly used IOCs can be found below. The iocmanager allows you to find the IOC in question by using the findPV option. Middle-clicking on elm (or pyDM) screens will show to the PV that has issues.

For a given PV you can also determine which server the IOC is running on and if rebooting the IOC does not help you can consider rebooting the server as described in the next paragraph.

Checking server health and power cycling (via serverStat)

The serverStat script as a few different options. **serverStat -l** will list those.

serverStat <machine name> will check the power status and ping the relevant interfaces (controls & DAQ is appropriate).

"**serverStat** <machine name> cycle" will powercycle the server, assuming the device as a working ipmi interface. You will need to be on the same (controls) network as the target machine. psdev has access to all networks, use your powers here responsibly! Be particularly mindful if the machine in question is a recorder machine as the RAID arrays ideally get extra care. serverStat also takes DAQ devices names (aliases) or PVs (only for PVs originating from the same subnet). If ipmi does not work, you can use 'netconfig view <servername>' to find the location information needed for a physical power cycle. Should this location not be correct, please let your controls POC know!

Controls Camera User Guide

We have a convenience script for controls cameras that can be called from the command line called **camViewer**. In case "which camViewer" does not return anything, this script is located at /reg/g/pcds/engineering_tools/<hutch>/scripts. This should be on the path for <hutch>opr accounts if the currently existing .bashrc is following our common pattern.

"camViewer -h" will list the available options. If no option is passed, you will get a list of cameras to choose from. By default, the camera viewer described here will be opened for the requested camera. The camera name can be passed using the "-c <cam #/cam name>" options. If you pass "-m" in addition to the "-c XXX" argument, you will get the expert edm screen. Should the viewer not work (e.g. not update), open the edm screen and start simple trouble shooting described. You can reboot the IOC either using the iocmanager or adding "-r" to your camViewer command. "camViewer -c camID -n " will ask the IOC to start acquiring images again: when you put a camera back online, it will often not acquire and needs to be told to. A reboot does the same thing, but with a lot more overhead. If you have a gige object defined in <hutch> python, you can also start the expert screen by using the <gige>.expert_screen() command.

If you would like to record your controls camera in the xtc stream along with the DAQ data, contact your PCDS-POC. The IOC will need to be build with support for timestamps so that images can be assigned to events. After that, they can be added to the DAQ under the "Camera IOCs" section.

Setting hardware binning and ROIs is described here.

GigE camera deployment instructions are found here

Troubleshooting:

The guide to trouble shoot the FEE cameras has many points valid across most cameras.

FEE Imagers trouble shooting

FEE Imager Camera Debug

IPIMB/Wave8

The configuration GUI for your IPIMB or Wave8 boxes can be started using: "**ipmConfigEpics** <-b boxname>". If you don't pass the "-b" option, you get a list of ipimb and wave8 boxes relevant for your hutch.

A troubleshooting guide for ipimb boxes can be found here: Troubleshooting for Controls IPIMB and Wave8s.

EPICS Gateways

Each hutch's home has a button that will show the health of all gateways. Each hutches gateway is responsible for sending data from that network out to other networks. At the current moment, it will appear white/down in that screen, this tis unfortunate, but normal.

Generally, red is not good. Having the Search Post rate higher than the Search Req rate indicates an issue. You may try resetting a gateway, but that WILL cause issues for other's using it, so please use with care! It will also not solve a problem when some requests simply fill up the queue. The BSA for ACR is a well known culprit and we have tried to move that all to its's own gateway to at least isolate problems. We are also about to move to newer server which will hopefully decrease the frequency of issues here.

Motor expert screens

Motor screens can be found in various formats at various places in the different hutches. If you are interested in the "expert" screen for most of our motor types, you can use

motor-expert-screen <PVNAME> where PVNAME is e.g. XPP:SB2:MMS:19

This script will bring up the right kind of motor screen (old & new IMS as well as Newport screens). In hutch python, you can also start the motor expert screens my using <motor>.expert_screen().

Newport screens are described here and the current setup for the new XPS-D controllers is described here

Motor setup

Newport

Turn XPS off before connecting motor!

Once everything is connected, turn XPS back on. Go to the IOC edm screen and reconnect, then click the "autoconfig" button. On success, a screen should come up telling you the motors the XPS found. Now your next steps are to initialize all motors and home or reference them. Now they should be accessible though the IOC.

To get them into your hutch python, you either have to:

(old python): add them to the correct epicsArch file. All motors found in the user/experiment specific file will be added to the "x" module (XPP/XCS).

(new python): motors added to the questionnaire (where they should be anyways) will be instantiated automatically.

XPS Newport Troubleshooting

IMS - smart motors

Add the motor to the questionnaire in the CDS tab. Given a python-name&PV, a motor python object will be instantiated upon restart. You can open the expert screen from the motor python object <motor>.expert_screen(), re-initialize from there & clear the power cycle (if applicable). *You can then use "<motor>.pmgr.diff()" to see the difference of current and saved configurations. "<motor>.pmgr.apply_config()" will apply the configuration. Check after applying the config that all parameters "made" it by calling pmgr.diff() on the same motor again.* (to be checked)

IMS - dumb motors (HXR)

This works very similar to the smart motor procedure, except that the serial number cannot be used to find the configuration. You will need to pass the configuration name to the "diff" or "apply_config" functions. If you don't, you will be asked. There is a search option where smart string matching is used when you don't know the config name. It is planned to alliviate this issue by labelling the stages with their config name (HXR). Dumb motor recognition does rely on the parameter manager knowing the serial numbers of the dumb motor controllers. Should you have a new chassis or a newly repaired one, this might not work and PCDS personnel will have to fix this.

Note: When running Navitar motors on the new MForce chassis there are two additional settings in the controller MCode that need to be set. **D1** and **D2** need to be set to 50. This can be done through the **:CMD** PV.

New Hutch-python

The new hutch python has extensive documentation at https://pcdshub.github.io/hutch-python/.

Vacuum

Old system Stubborn Cold Cathodes

Hutch specific instructions

TMO quick nodes/how-tos

CXI

Expert guides (incl LCLS1 DAQ troubleshooting)

Link to the old "Controls User Guide"

LCLS-1 DAQ Tier-1 Troubleshooting