

Operating the DAQ

Contents:

- [Editing a configuration](#)
 - [Example: Opal camera configuration](#)
 - [Changing a configuration while running](#)
 - [Configuring the EVR](#)
- [Selecting detectors for readout](#)
- [Starting/stopping a run](#)
- [Watching progress of a run](#)
- [Running scans](#)
 - [Launching a Scan from DAQ Control GUI](#)
 - [Launching Scans Remotely \(scripted\)](#)
- [Running the sequencer](#)
- [Recovering from errors:](#)

Editing a configuration

If the DAQ system is not already running, then there should be a "Restart DAQ" icon on the operators console. Open that to start the DAQ system. When you do you will get a number of windows opening. In XPP or XCS, use "restartdaq" from a terminal. "restartdaq -w" will put the windows in predetermined locations on the screens/

Example: Opal camera configuration

To edit the configuration click the Edit button near the top of the control gui window shown below.

DAQ Control (on rddaqcal)

Configuration

Type JACK

Edit

Scan

Partition

Select

Display

Control

Record Run

Target State

Last Transition

Detector

Control State

NOT READY

Run Statistics

Duration

Events

Damaged

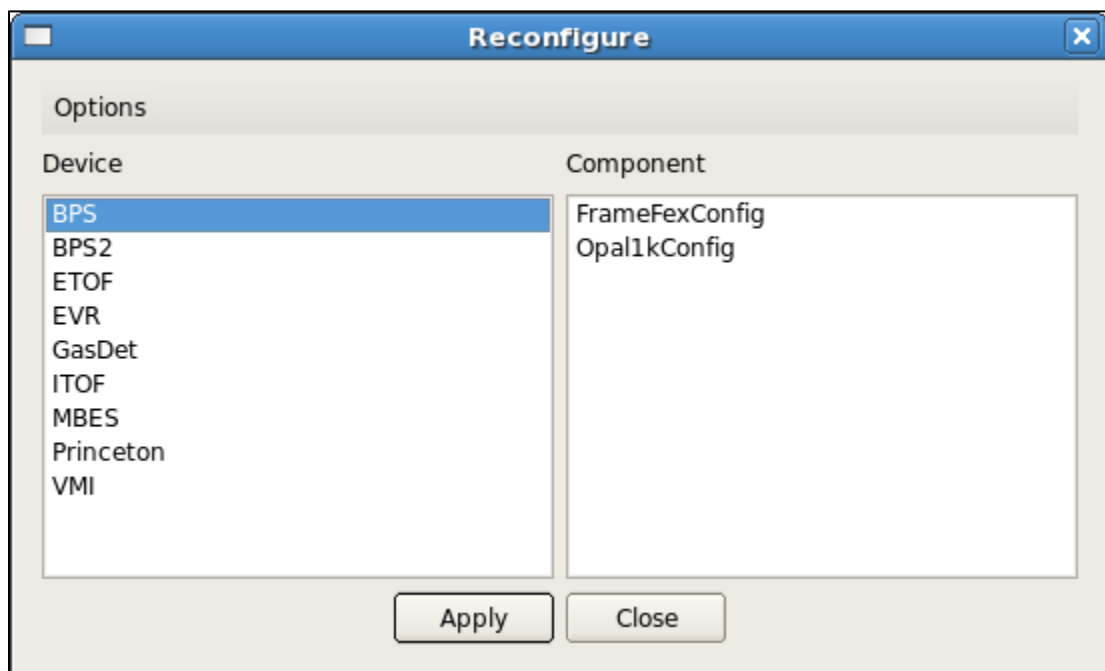
Size

Damage Stats

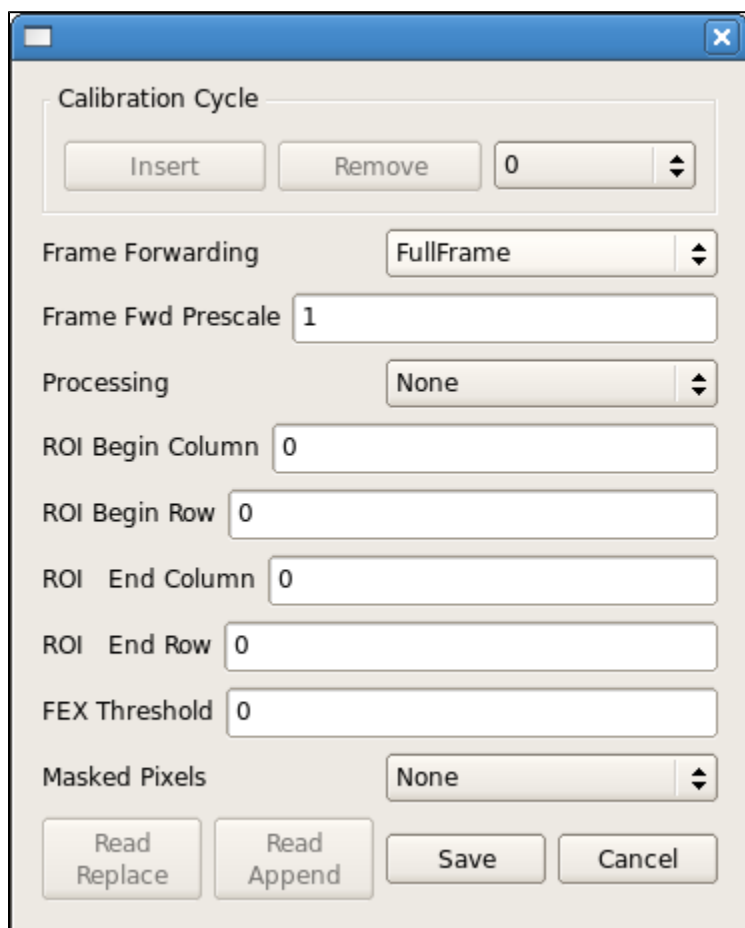
17:02:44: Run control started

Connected to platform.

Clicking Edit brings up a new Reconfigure window with two columns labeled Device and Component. Click on an entry in the Device column to display one or more corresponding entries in the Component column. In this example, an OPAL-1000 device named BPS has been selected.



After selecting the device of interest on the left, click on each component on the right to edit it. In this example there are two configurable components. This screen opens to edit FrameFexConfig (feature extraction):



This screen opens to edit Opal1kConfig (OPAL-1000 camera):

Calibration Cycle

Insert Remove 0

Black Level 32

Gain 100

Depth 12 Bit

Binning x1

Mirroring None

Vertical Remap True

Defect Pixel Correction False

Output Lookup Table None

Read Replace Read Append Save Cancel

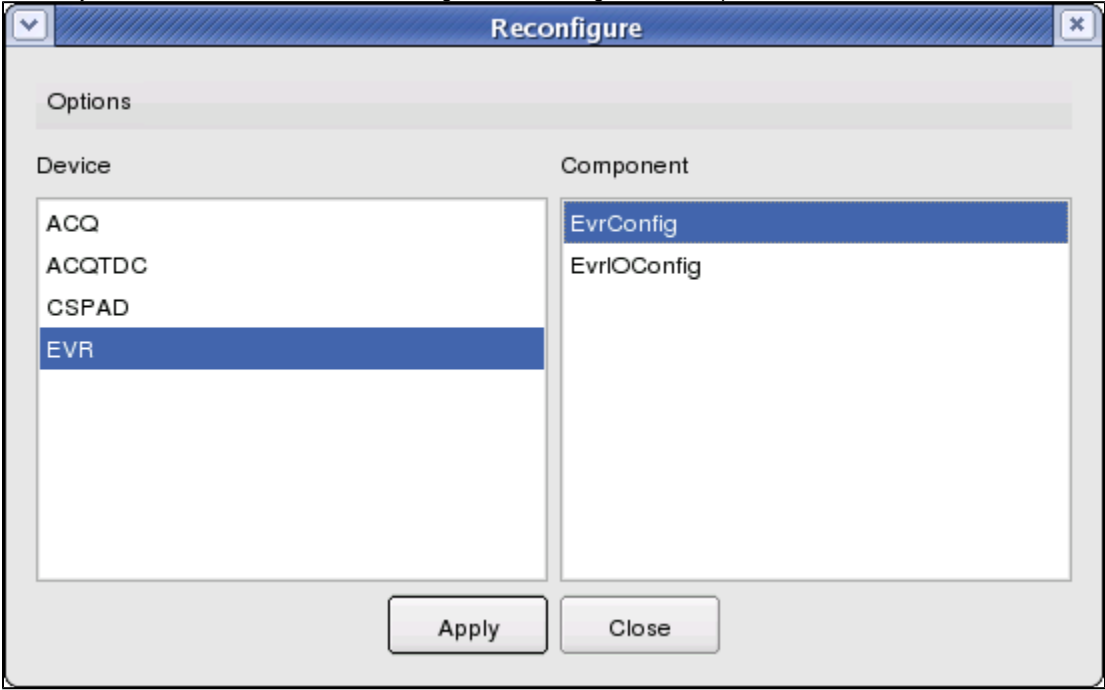
Review the current settings and make any changes necessary in the editing windows. Click on the Save button if you wish to close the window and save changes, otherwise click on the Cancel button to close the window without saving. If you saved any changes in the edit windows, click on the Apply button in the Reconfigure window to commit those changes to the DAQ configuration.

Changing a configuration while running

If changes are committed to the DAQ configuration while a DAQ run is in progress, the DAQ system will automatically end the run, unconfigure, configure, then begin a new run. If recording data, the run number will be incremented. The configuration does not vary within a single recorded run.

Configuring the EVR

The EVR (Event Receiver) is responsible for generating readout triggers for the detector. It can be configured to generate trigger pulses of a selected polarity, duration, and delay, and in response to signaled events (eventcodes) from the accelerator timing system like the arrival of beam. The EVR configuration screen, labeled "EVR" is found among the other detector configurations as described above. After clicking "EVR" in the "Reconfigure" window, you will be able to select either "EvrConfig" or "EvrIOConfig" in the "Component" section.



The "EvrIOConfig" panel is not in use anymore, the names for the triggerlines are now set from the <hutch>.cnf. The "EvrConfig" panel allows you to edit the trigger settings.

PulsesEventCodesSequencer

Pulses generated by "Readout" EventCode
Pulse delay is specified with respect to EventCode 140

EVR 0EVR 1EVR 2EVR 3EVR 4EVR 5EVR 6EVR 7

Pulses generated by "Readout" EventCode
Pulse delay is specified with respect to EventCode 140

Enable	Pulse Polarity	Pulse Delay (sec)	Pulse Width (sec)	Camera-0	Camera-1	ADC-0	TDC-0								
<input checked="" type="checkbox"/>	Pos	0.0008	1.0084e-07	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Neg	0.000842	0.001	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															
<input type="checkbox"/>															

SaveCancel

The "EvrConfig" panel has three tabs labeled "Pulses", "EventCodes", and "Sequencer". The Pulses tab allows you to edit the set of trigger pulse parameters along with the set of output lines they drive. Pulse delays and widths are expressed in seconds assuming the global timing system clock runs at exactly 119MHz. The pulse delays are expressed with respect to the timeslot reserved for the arrival of eventcode 140 (the eventcode that signifies the presence of beam). The software will internally modify this number when a readout eventcode other than 140 is selected (see below) in order to maintain the same pulse delay with respect to the timing system fiducial.

The screenshot shows the "EventCodes" tab of the "EvrConfig" panel. It contains two main sections: "Sequencer Codes" and "Global Codes".

Sequencer Codes: This section has a header with "Event Code Range" set to 67 to 74. Below is a table with columns: Enable, Code, Type, Describe, and Reporting. Event code 67 is enabled and set to "Control[Transient]" with a description of "laser" and a delay of 0. Event codes 68 through 74 are disabled.

Enable	Code	Type	Describe	Reporting
<input checked="" type="checkbox"/>	67	Control[Transient]	laser	Delay 0 Duration 1
<input type="checkbox"/>	68			
<input type="checkbox"/>	69			
<input type="checkbox"/>	70			
<input type="checkbox"/>	71			
<input type="checkbox"/>	72			
<input type="checkbox"/>	73			
<input type="checkbox"/>	74			

Global Codes: This section has a header with columns: Enable, Code, Type, Describe, and Reporting. It lists four codes: 42 [30 Hz] (enabled, Readout, readout), 40 [120 Hz] (disabled), 40 [120 Hz] (disabled), and 40 [120 Hz] (disabled).

Enable	Code	Type	Describe	Reporting
<input checked="" type="checkbox"/>	42 [30 Hz]	Readout	readout	
<input type="checkbox"/>	40 [120 Hz]			
<input type="checkbox"/>	40 [120 Hz]			
<input type="checkbox"/>	40 [120 Hz]			

At the bottom are "Save" and "Cancel" buttons.

The EventCodes tab is where you can select to record the arrival of various eventcodes and choose the eventcode from which the trigger pulses will be generated. The Sequencer Codes section is now deprecated and cannot be selected anymore. An eventcode can be designated as Readout, Command, Control(Transient), and Control(Latch). An eventcode designated as Readout will generate the trigger pulses on each occurrence. There should be only one eventcode designated as such. An eventcode designated as Command will generate a software event which can be used for software command generated readout (like for a Princeton camera). That software generated event will be collected along with the other detectors on the next occurrence of a Readout event. An eventcode designated as Control(Transient) will be recorded in the datastream for each occurrence with the specified delay and duration (in units of readout occurrences). This allows a record to track the occurrence of some event triggered from that eventcode elsewhere, like the pump laser for instance. It is also possible to add a list of event codes to record in the <hutch>.cnf file, removing the need to set commonly used event codes in the EVR config if they are needed for recording only. An eventcode designated as Control(Latch) will be recorded in the datastream on every readout event following its occurrence until the complementary Control(Latch) code is received. For example, a pulse picker state can be recorded by the occurrence of the commands responsible for its "Open" and "Close" operation. For many experiments, only the Readout eventcode needs to be designated.

The Sequencer tab allows the Event Sequencer to be configured in the same panel for convenience. It provides the same function as the EPICS Sequencer configuration described later.

Selecting detectors for readout

To select detectors for readout click the Select button in the Partition panel of the control gui window shown below.



A new window opens, and initially all detectors used the last time the DAQ was run are selected. If nodes had to be powercycled, the processes running on them are deselected. Sometimes processes will just become unselected, so be sure to check your partition!

The image shows a 'Partition Selection' dialog box with two sections: 'Readout Nodes' and 'Processing Nodes'. Each section contains a list of nodes with checkboxes. All checkboxes are currently checked. At the bottom are 'Ok' and 'Cancel' buttons.

Section	Node Name	IP Address	Port
Readout Nodes	AmoBPS/1/Opal1000/0	172.21.20.92	6737
	AmoETO/0/Acqiris/0	172.21.20.35	20934
	AmoITOF/0/Acqiris/0	172.21.20.37	12110
	AmoMBES/0/Acqiris/0	172.21.20.43	26039
	AmoVMI/0/Opal1000/0	172.21.20.92	6743
	BldEb/0/NoDevice/0	172.21.20.89	18321
	EpicsArch/0/NoDevice/0	172.21.20.43	26044
	NoDetector/0/Evr/0	172.21.20.34	16516
Processing Nodes	Event	172.21.20.87	7284
	Event	172.21.20.88	4385
	Event	172.21.20.89	18322
	Event	172.21.20.90	24142

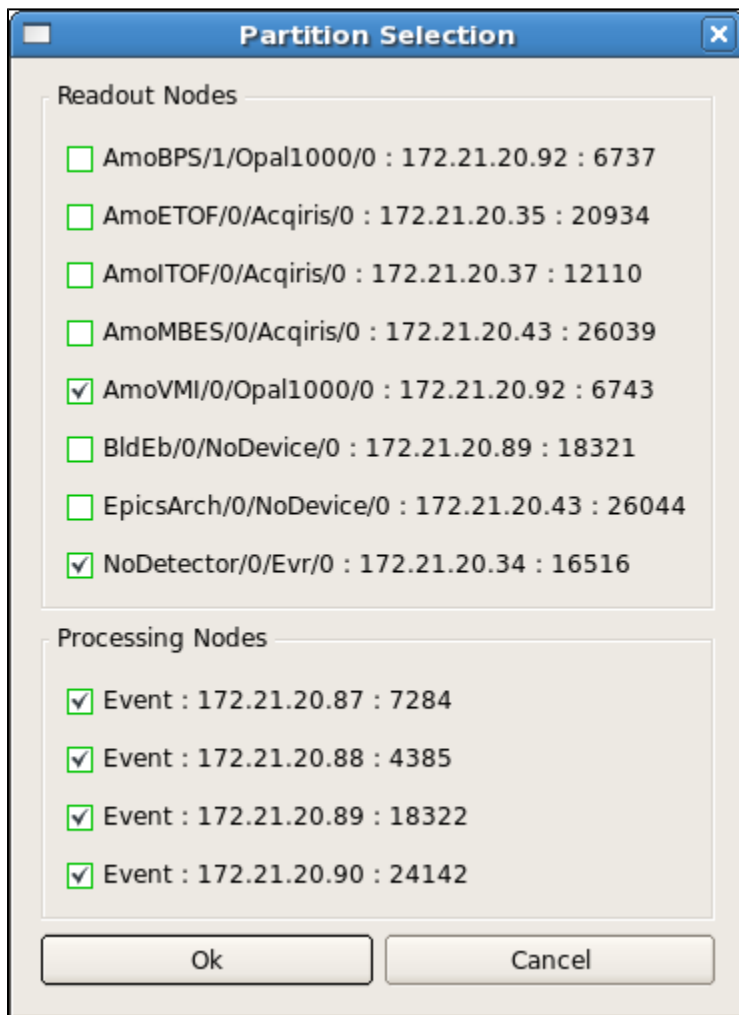
Click on a checkbox to unselect the corresponding readout node. **Do not unselect the Nodetector/0/Evr/0 node, as it provides timing for the DAQ system.**

Here are some reasons for selecting a subset of readout nodes:

- Remove unused readout nodes to reduce data volume
- Remove an unresponsive readout node to eliminate errors
- Test a readout node in isolation

Please note that for ami to (reliably) work, you need to select *all* dss nodes that are listed in the <hutch>.cnf file.

The following example shows the selection of a single camera. Note that the Evr is always required:



The image shows a 'Partition Selection' dialog box with a blue title bar and a close button. It contains two sections: 'Readout Nodes' and 'Processing Nodes'. Each section has a list of nodes with checkboxes. In the 'Readout Nodes' section, 'AmoVMI/0/Opal1000/0 : 172.21.20.92 : 6743' and 'NoDetector/0/Evr/0 : 172.21.20.34 : 16516' are checked. In the 'Processing Nodes' section, all four listed nodes are checked. At the bottom are 'Ok' and 'Cancel' buttons.

Section	Node Name	IP Address	Port	Selected
Readout Nodes	AmoBPS/1/Opal1000/0	172.21.20.92	6737	<input type="checkbox"/>
	AmoETO/0/Acqiris/0	172.21.20.35	20934	<input type="checkbox"/>
	AmoITOF/0/Acqiris/0	172.21.20.37	12110	<input type="checkbox"/>
	AmoMBES/0/Acqiris/0	172.21.20.43	26039	<input type="checkbox"/>
	AmoVMI/0/Opal1000/0	172.21.20.92	6743	<input checked="" type="checkbox"/>
	BldEb/0/NoDevice/0	172.21.20.89	18321	<input type="checkbox"/>
	EpicsArch/0/NoDevice/0	172.21.20.43	26044	<input type="checkbox"/>
	NoDetector/0/Evr/0	172.21.20.34	16516	<input checked="" type="checkbox"/>
Processing Nodes	Event	172.21.20.87	7284	<input checked="" type="checkbox"/>
	Event	172.21.20.88	4385	<input checked="" type="checkbox"/>
	Event	172.21.20.89	18322	<input checked="" type="checkbox"/>
	Event	172.21.20.90	24142	<input checked="" type="checkbox"/>

Click the OK button to save the selection and close the window, or click the Cancel button to discard changes.

Starting/stopping a run

If the DAQ system is not already running, then there should be a "Restart DAQ" icon on the operators console. Open that to start the DAQ system. When you do you will get a number of windows opening.

First, examine the control gui window shown below. From the Configuration panel, select the appropriate configuration Type.

DAQ Control (on rddaqcal)

Configuration

Type **JACK**

Edit

Scan

Partition

Select Display

Control

Record Run

Target State Last Transition

NOT READY

Detector

Control State

Run Statistics

Duration

Events

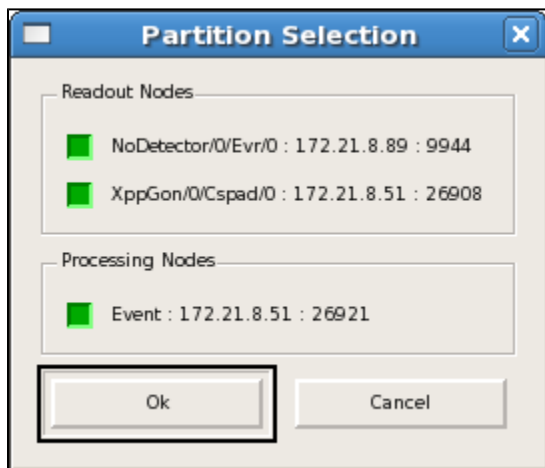
Damaged

Size

Damage Stats

17:02:44: Run control started
Connected to platform.

Next, you should click on the Partition Select button to choose what devices will be in the system during the run. You will get a window like the one shown below, just click on an item to toggle its state as in or out of the partition during the run.



Once you have clicked "OK" above, you will get a target state offering to allow you to start the run. You can see this below. Just select "BEGIN RUNNING" to start the run.

If you wish to record the run, make sure that the Record Run box in the Control panel is clicked and green (default). If you don't wish to record the run, unselect "Record Run". It will turn yellow to indicate the run will not be recorded.

DAQ Control (on rddaqcal)

Configuration

Type: JACK

Edit

Scan

Partition

Select

Display

Control

☐ Record Run

Target State: BEGIN RUNNING

Last Transition: Configure

Detector

Control State: READY

Run Statistics

Duration

Events

Damaged

Size

Damage Stats

17:02:44: Run control started
Connected to platform.

You can stop the run from the control gui window by selecting one of the available Target States from the box in the Control Panel. The states are "disable", "end running" or "shut down".

- Disable will allow you to continue the same run later.
- End Running will end the run so that next time you will get a new run number.
- If you need to reconfigure or are finished, then shut down.

Watching progress of a run

Once the run is going you can watch the progress of it on the control gui window. It will show you the statistics, like the duration of the run and the number of events including the damaged event count. The bottom pane gives the file status, recording or not. Notice that the file names are given even if not recording. Embedded in the file name, you will see the run number.

The image shows a GUI window titled "DAQ Control (on rddaqca)". The window is divided into several sections:

- Configuration:** Contains a "Type" dropdown menu set to "JACK", and "Edit" and "Scan" buttons.
- Partition:** Contains "Select" and "Display" buttons.
- Control:** Features a yellow "Record Run" button, a "Target State" dropdown menu set to "DISABLE", and a green "Enable" button. A "Last Transition" label is also present.
- Detector:** Shows a green "READY" button under the "Control State" label.
- Run Statistics:** Displays "Duration 0:0:8", "Events 933", "Damaged 0", and "Size 4286 MBytes". A green "Damage Stats" button is at the bottom of this section.
- Log:** A text area at the bottom showing timestamps and messages: "17:02:44: Run control started Connected to platform." and "17:04:52: Not recording. Transient data file: e0-r1287705892-sNN-cNN.xtc".

Other windows will show you the status of the DAQ processes, or allow you to do online monitoring of each device.

Running scans

Most every experiment requires efficient acquisition of data coordinated with changes in the beamline or detector; i.e. a parameter scan. The DAQ system provides two separate interfaces for automating such a scan.

Launching a Scan from DAQ Control GUI

The first interface is part of the DAQ Control GUI, and is found by clicking the "Scan" button in the "Configuration" section of the main GUI window. This opens a dialog which has two tabs : one for scanning an EPICS process variable ("PV") and one for scanning a DAQ trigger delay.

The screenshot shows the 'DAQ Control' dialog box. At the top, there's a title bar with a dropdown arrow, the text 'DAQ Control', and standard window controls. Below the title bar, there's a 'Steps' input field with the value '50'. Underneath, there are two tabs: 'EPICS PV' (which is selected) and 'DAQ Trigger'. The 'EPICS PV' tab contains several input fields: 'Control Channel' with the text 'AMO:ITOF:MMS', 'Start' with '10', and 'Stop' with '20'. Below these are 'Readback Channel', 'Offset', and 'Margin', all of which are empty. There is also a checkbox labeled 'Settle Time' followed by an empty input field and the word 'seconds'. At the bottom of the dialog, there's a section titled 'Acquisition / Step' containing two radio buttons: 'Events' (which is selected) and 'Time'. The 'Events' radio button is followed by an input field with the value '60'. The 'Time' radio button is followed by an empty input field and the word 'seconds'. At the very bottom of the dialog are two buttons: 'Apply' and 'Close'.

For the EPICS PV scan, the user chooses the number of steps, the name of the PV to control, and the start and stop value of that variable. The scan points will be calculated linearly from the steps, start, and stop values:

- 1 step = 2 scan points {start, stop},
- 2 steps = 3 data points {start, (start+stop)/2, stop},
- 3 steps = 4 data points {start, start*2/3+stop*1/3, start*1/3 + stop*2/3, stop}, and so on.

A readback channel may also be entered if there is a need to separately monitor the precision with which the control variable makes its steps. In this instance, the "offset" specifies any systematic difference of the readback value from the control value, and the "margin" specifies how accurately the readback channel (minus offset) must match the control channel for data acquisition to continue.

DAQ Control

Steps

EPICS PV DAQ Trigger

Pulse

Width [s]

Delay Begin [s]

Delay End [s]

Acquisition / Step

☒ Events

☐ Time seconds

Apply Close

For the DAQ trigger scan, the user provides the number of steps, the pulse number to scan, the width of the pulse (in seconds), and the begin and end delay settings of the pulse (also in seconds). The scan points for the delay values are calculated linearly from the steps, begin, and end values as described above for EPICS PVs.

Finally, the user enters either the number of events to acquire at each step or the amount of time with which to acquire events at each step. Clicking "Apply" will interrupt any current running and start the scan in a new run. The scan will proceed automatically to completion. It may be interrupted manually by choosing "EndRunning" from the "Target State" button. "BeginRunning" will launch another scan as long as the "Scan" button remains highlighted. Close the Scan dialog to disable scans.

Launching Scans Remotely (scripted)

The DAQ system also supports a socket interface to allow scripts or other processes to make acquisition requests of the DAQ. In this way, the script becomes responsible for iterating the scan variable (setting an EPICS PV) and commanding the DAQ to acquire data at each scan point. Thus, non-linear scans or multi-dimensional scans may be implemented as needed by the user.

This is integrated into the <hutch> python environment which most hutches have which is described in more detail elsewhere. We recommend using that over writing your own scripts.

An example Python script for running a scan may look like this:


```

import socket
import DaqScan
import ConfigDb

if __name__ == "__main__":
#
# Connect the socket to the appropriate hutch DAQ system
#
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((options.host,options.port))
#
# Get the current configuration key in use and set the value to be used
# (assumes we are using the same detector configuration as currently running)
#
    cdb = ConfigDb.Db()
    cdb.recv_path(s)
    key = DaqScan.DAQKey(s)
    key.set(key.value)
#
# Send the structure the first time to put the control variables
# in the file header
#
    data = DaqScan.DAQData()
    data.setevents(0)
    data.addcontrol(DaqScan.ControlPV('EXAMPLEPV1',0))
    data.addcontrol(DaqScan.ControlPV('EXAMPLEPV2',0))
    data.addmonitor(DaqScan.MonitorPV('BEAM:LCLS:ELEC:Q',options.qbeam,1.))
    data.send(s)
#
# Wait for the DAQ to declare 'configured'
#
    result = DaqScan.DAQStatus(s)
    print "Configured."
#
# Wait for the user to declare 'ready'
# Setting up monitoring displays for example
#
    ready = raw_input('--Hit Enter when Ready-->')
    for cycle in range(options.cycles):
        data = DaqScan.DAQData()
        data.setevents(options.events)
        data.addcontrol(DaqScan.ControlPV('EXAMPLEPV1',cycle))
        data.addcontrol(DaqScan.ControlPV('EXAMPLEPV2',100-cycle))
        data.addmonitor(DaqScan.MonitorPV('BEAM:LCLS:ELEC:Q',options.qbeam,1.))
        print "Cycle ", cycle
        data.send(s)
        result = DaqScan.DAQStatus(s) # wait for enabled , then enable the EVR sequence if needed
        result = DaqScan.DAQStatus(s) # wait for disabled, then disable the EVR sequence if needed
#
# Wait for the user to declare 'done'
# Saving monitoring displays for example
#
    ready = raw_input('--Hit Enter when Done-->')
    s.close()

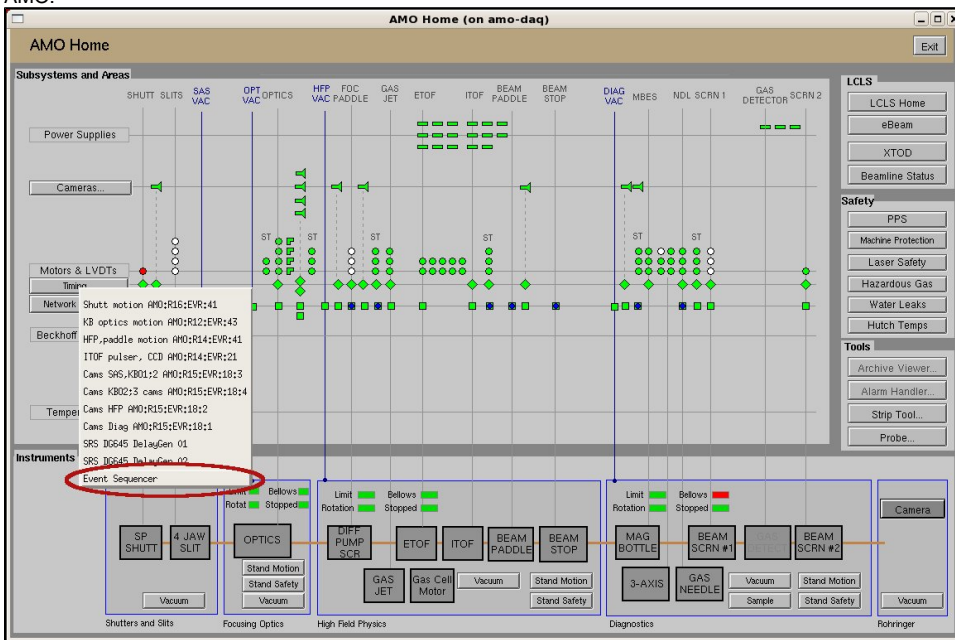
```

When the socket connects, manual control through the DAQ Control GUI is disabled. The script retrieves detector readout configuration from the DAQ system, creates a new one if needed (advanced), and instructs the DAQ which configuration to use. Then, the script informs the DAQ what are the scan variables to be controlled (for communication to the online monitoring and recorded data), and what EPICS PVs, if any, should be monitored as a condition for taking data (none is common). Then, the script takes responsibility for controlling the scan variables, and informs the DAQ of the scan variable settings at each step and the number of events to acquire. The DAQStatus method is called to enable the DAQ to acquire data. A second call to DAQStatus then waits for the DAQ to complete the acquisition for that step. When the socket is closed, manual control through the DAQ Control GUI resumes.

Running the sequencer

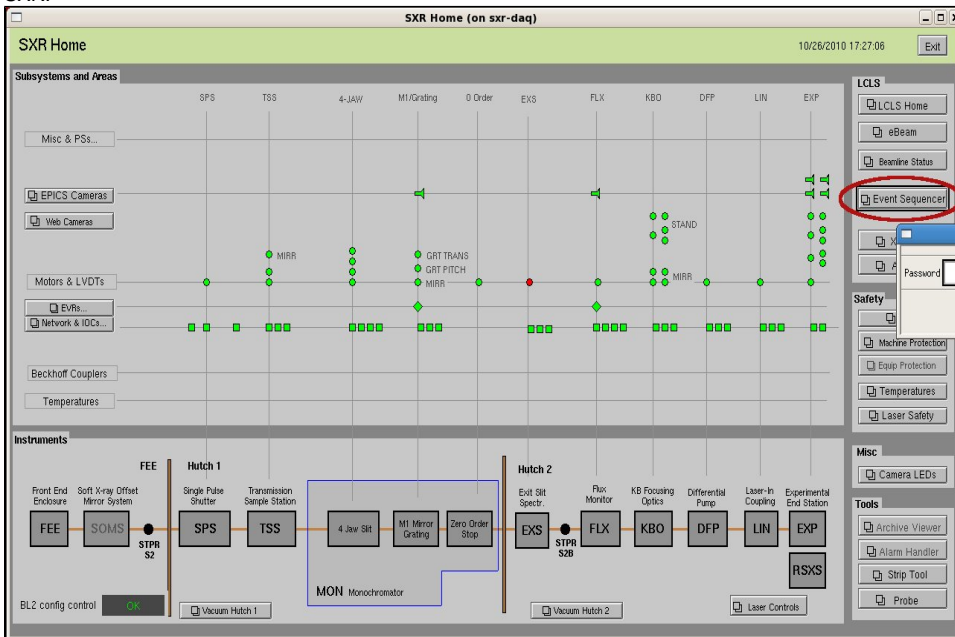
The event sequencer window can be launched from the main epics window for each experiment. The following images show the location of event sequencer menus in different epics windows:

1. AMO:



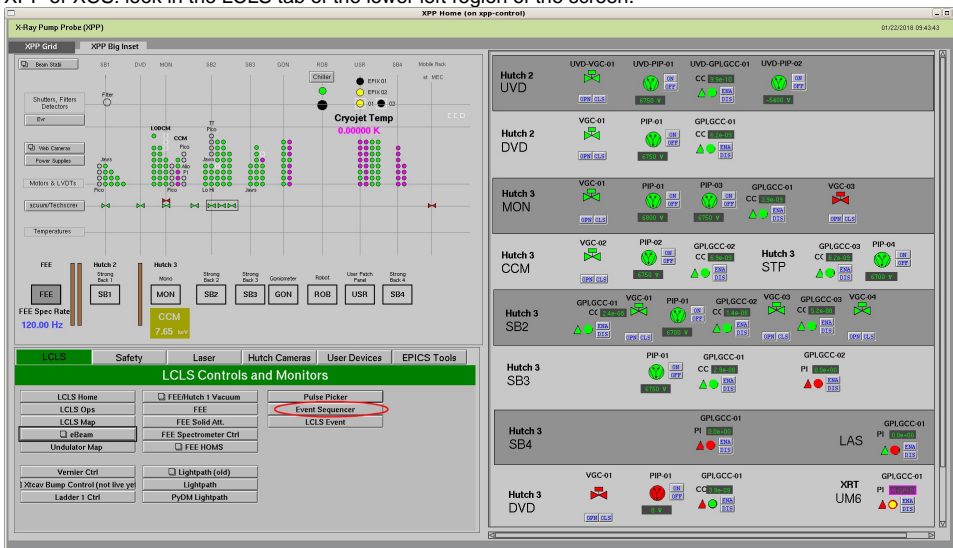
Note: Click on the menu "Timing"->"Event Sequencer" to bring up the event sequencer window.

2. SXR:



Note: The event sequencer window is password protected.

3. XPP or XCS: look in the LCLS tab of the lower left region of the screen.



After clicking on the event sequencer button in the epics window, and entering the password (if necessary), the event sequencer window will pop up, as below:

☐
Event Sequence Configuration (on xpp-control)

Event Code Sequence 3

Sequence Owner 3 XPP
Photon Beam Owner 2 SXR

Once
Repeat N Times
Repeat Forever

Status
Beam Rate 120.0
Play Count 0
Total Play Cnt 0
Play Status Stopped
Current Step 0

Synchronization Settings
Sync Marker 120Hz ☐
Next Sync Immediate
Run using Timeslot ☐
☐ Spare Sequences

Start Stop
Not the beam owner!

Define Sequence 3

Valid Sequence at step 0

☐ Event Code Allocation
Load spreadsheet
Save spreadsheet

Stop at Step: 14

Step	Event Code	Δ Beam	Δ Fiducial	Burst Count	Device/Function
0	94	1	0	0	
1	96	0	0	0	
2	96	1	0	0	
3	95	1	0	0	
4	95	1	0	0	
5	95	1	0	0	
6	95	1	0	0	
7	95	1	0	0	
8	95	1	0	0	
9	95	1	0	0	
10	95	1	0	0	
11	95	1	0	0	
12	94	0	0	0	
13	95	1	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	

The main function of the "Run Sequence" part is to let you start or stop the sequence playing, but clicking on the "Start" or "Stop" button. Also it shows the current beam rate, play counts and the running status of the sequencer. You can also control the play looping by clicking on

"Once", "Repeat N Times" or "Repeat Forever". More details can be found at [this confluence page](#).

The event sequencer window has upper (titled as "Run Sequence") and lower parts (titled as "Define Sequence"). Here we only focus on the upper part.

If you click on "Repeat N Times", there will show an additional input field for entering the number of loops.

In the DAQ, you have the option to select "Sync Sequence" where you have to enter the sequence you want to sync to. The DAQ will now start the sequencer on the "enable" transition and stop it on "disable". This is useful for scans in which a sequence will be repeated for each step. Please check that the timing of the sequence running the the DAQ works, in particular if you have slow to configure detectors in the DAQ.

☐ **DAQ Control (on xcs-conti** — □ ×

xcsx29616 [1129]

Configuration

Type

☒ Sync Sequence

Partition

Control

☐ **Record Run**

Target State Last Transition

Detector

Control State

READY

Run Statistics

Duration

Events

Damaged

Size

(XCS:SND:DIA:DCC:X.RBV)
snd_DIA_dcc_Y
(XCS:SND:DIA:DCC:Y.RBV)
snd_DIA_dd_x
(XCS:SND:DIA:DD:X.RBV)
snd_DIA_dd_Y
(XCS:SND:DIA:DD:Y.RBV)
snd_DIA_dci_x
(XCS:SND:DIA:DCI:X.RBV)
snd_DIA_dco_x
(XCS:SND:DIA:DCO:X.RBV)

Recovering from errors:

[LCLS-1 DAQ Tier-1 Troubleshooting](#)

[Scans etc \(presentation from Oct 2021\)](#)

[Running scans, PVs, elog, and smalldata tools overview](#)