ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH1002

adviewer: THE EPICS AREA DETECTOR CONFIGURATOR YOU DIDN'T KNOW YOU NEEDED *

Kenneth Lauer†, SLAC National Accelerator Laboratory, Menlo Park, CA, USA

Abstract

EPICS areaDetector connects area detector cameras to plugin pipelines through the standard flat namespace that EPICS provides. Visualizing and re-configuring this port connectivity in AreaDetector can be confusing and - at times - painful. adviewer provides a Qt-based interactive graph visualization of all cameras and plugins, along with per-plugin configuration capabilities and integration with an image viewer. adviewer is built on Python, ophyd, typhon, qtpynodeeditor, and Qt (via qtpy).

BACKGROUND

areaDetector

Each areaDetector [1] related device class from ophyd [2] redresents a single 'port', i.e., a camera or plugin. All plugins have a source port, indicating the location from which their input data will be retrieved. Cameras do not have a source port, as the information is sourced from a lower level (i.e., the C++ driver support).

To connect one camera to a plugin, one would need to set the plugin's source port to that of the camera. An ordered set of camera and any number of plugins is referred to in this document as a chain. A full configuration of plugins and their corresponding cameras can be represented in several intuitive ways, including as a directed acyclic graph comprised of one or more chains, or as a tree in which the parent of an item indicates its data source where a depth-first traversal reveals the individual chains.

adviewer provides a graphical user interface with both representations of detector configurations, an interactive tree or graph.

areaDetector DEVICES IN ophyd

ophyd makes available many device abstractions from the broader EPICS [3] community - from the motor record, scalers, and so on. There is first-class support for areaDetector detectors and plugins.

Camera and plugin support is summarized in Table 1. Support for unlisted cameras and plugins are welcomed in the form of Pull Requests to the main ophyd repository [4].

Versioning

All ophyd devices can be versioned[1] with user-provided metadata at the device definition time. Versioning is done in a class hierarchy, such that later versions are subclasses of previous versions and are marked as versions of the base.

Table 1: areaDetector Support in ophyd

Plugins	Cameras
AttrPlotPlugin	AdscDetectorCam
AttributePlugin	Andor3DetectorCam
CircularBuffPlugin	AndorDetectorCam
CodecPlugin	BrukerDetectorCam
ColorConvPlugin	DexelaDetectorCam
FFTPlugin	FirewireLinDetectorCam
FilePlugin	FirewireWinDetectorCam
GatherPlugin	GreatEyesDetectorCam
HDF5Plugin	Lambda750kCam
ImagePlugin	LightFieldDetectorCam
JPEGPlugin	Mar345DetectorCam
MagickPlugin	MarCCDDetectorCam
NetCDFPlugin	PSLDetectorCam
NexusPlugin	PcoDetectorCam
Overlay	PerkinElmerDetectorCam
OverlayPlugin	PilatusDetectorCam
PluginBase	PixiradDetectorCam
PosPlugin	PointGreyDetectorCam
ProcessPlugin	ProsilicaDetectorCam
PvaPlugin	PvcamDetectorCam
ROIPlugin	RoperDetectorCam
ROIStatPlugin	SimDetectorCam
ScatterPlugin	URLDetectorCam
StatsPlugin	
TIFFPlugin	
TimeSeriesPlugin	
TransformPlugin	

Users may select individual versions manually or programmatically request the most compatible class with a specific release.

This is especially of use for areaDetector as the PV interface to each of the cameras and plugins has changed significantly over the many releases from R1-9 through R3-7.

Cameras

Cameras in areaDetector generally differ from detector to detector. ophyd provides a base class that all cameras are derived from, along with individual cameras for each detector.

Standard Plugins

The standard plugins provided in areaDetector ADCore are all made available, based on a standard 'PluginBase' class, separated by plugin class and versioned by ADCore. As of the ophyd v1.4.0rc2, all listed plugins are supported from areaDetectorR1-9-1 through R3-4, inclusive.

* Work supported by U.S. D.O.E. Contract DE-AC02-76SF00515.
† klauer@slac.stanford.edu
[1] As of the v1.4 release candidate

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH1002

Detectors

Detectors in ophyd aggregate one or more cameras and one or more plugins into a device.

DETECTOR SPECIFICATION IN ADVIEWER

adviewer provides two main interfaces for specifying a detector. It allows for general discovery over Channel Access (CA) with only a PV prefix, and also specification of a partial or complete PV list from the IOC.

This "discovery" functionality is shown in Fig. 1.

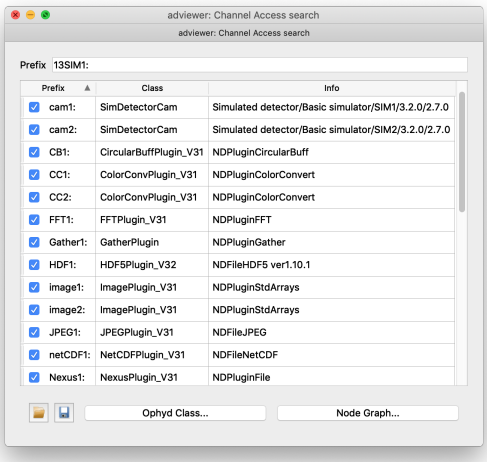

Figure 1: adviewer: search over Channel Access by prefix or use a PV list.

Discovery over EPICS Channel Access

With this method, the only required user input is a PV prefix. That is, if a plugin PV such as `13SIM1:image1:ArrayData_RBV` exists, the common prefix would be only `13SIM1:`.

For each ophyd areaDetector plugin, a regular expression is provided that indicates what a common plugin suffix might be - as defined by the ADCore-provided `commonPlugins.cmd`.

adviewer utilizes this by taking the user-provided prefix, unrolling the common suffix expression to allow for multiple plugins, and searching for a single known PV indicating the plugin type (`:PluginType_RBV`) over Channel Access.

A similar method is used for the cameras, allowing for adviewer to determine the make, manufacturer, ADCore version, and camera driver version.

Discovery by PV List

For IOCs that do not follow the same convention as that suggested by ADCore `commonPlugins.cmd`, adviewer also allows for a PV list file to be specified.

The list is filtered to include only those that are likely to be cameras or plugins. adviewer then reaches out over Channel Access to determine camera and plugin versions and availability.

Using the Camera and Plugin List

Once a detector has been specified by either a PV list or a search over CA, there are several primary features then available:

- Creating the Python code for an ophyd detector
- Configuring the connectivity of ports via a port graph or tree
- Saving to a PV list

For the above, plugins may be removed from the full list by selecting a checkbox next to the item. That would mean that for a given unchecked plugin, it would no longer appear in the generated Python code nor in the port graph.

CONFIGURING THE PORT GRAPH

The main port graph window includes both a port tree and a graph, as pictured in Fig. 2.

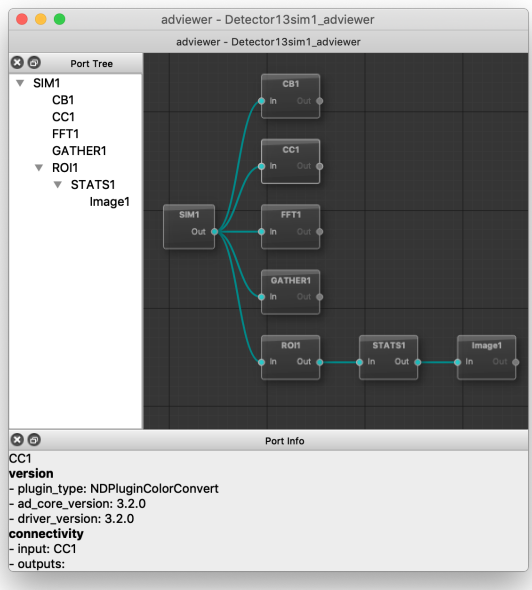

Figure 2: adviewer: the port tree and graph.

Using the Port Tree

In Fig. 2, the left side represents the hierarchy of all plugins using a tree. Each line is a single camera or plugin, using the port name as defined by the IOC startup script.

As the configuration is available through EPICS, this tree automatically updates as any client reconfigures a port chain.

The tree is interactive, allowing for dragging and dropping of one port to another location. For example, one could decide that an ROI is necessary prior to running some statistics, and as such they might drag `STATS2` to be under `ROI2`. The plugin chain in text would be the following: `SIM1 STATS2 ROI2`.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOSH1002

Using the Graph

In Fig. 2, the right side represents the hierarchy of all plugins using a directed acyclic graph. Each rectangle (node) is a single camera or plugin, annotated by the unique port name assigned to it. The direction is defined as from "Out" to "In" on the nodes, or generally left-to-right.

To reconfigure a plugin chain, it is only necessary to click and drag a new connection on the graph.

Right-clicking on a single port pops up a context menu that allows for the configuration of any parameter on the plugin, pictured in Fig. 3. For image plugins, it also allows the user to spawn a user-specified (by default, a PyDM-based) image viewer.

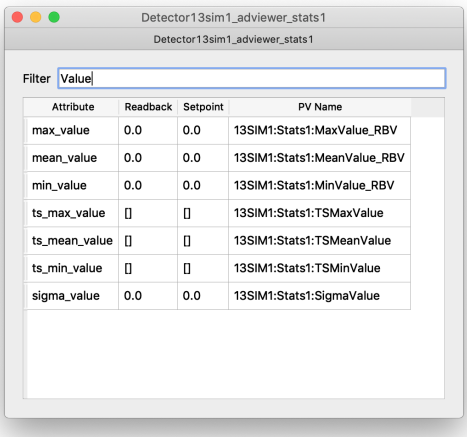

Figure 3: adviewer: view or change the settings of a plugin.

FUTURE WORK

Future work might include:

- Aiding PyDM to build out its image viewer
- Showing plugin settings in the graph nodes
- Showing preview images in the graph nodes
- Expanding on ophyd's detector definitions
- Include a full image viewer directly in adviewer

REFERENCES

[1] areaDetector: EPICS software for area detectors, https://cars9.uchicago.edu/software/epics/areaDetector.html

[2] ophyd and the bluesky project, https://blueskyproject.io/

[3] EPICS – Experimental Physics and Industrial Control System, https://epics.anl.gov/

[4] ophyd repository, http://www.github.com/bluesky/ophyd.