



U.S. DEPARTMENT  
*of* ENERGY

# Simulated EPICS IOCs

Thomas Hopkins

May 16<sup>th</sup>, 2025

    @BrookhavenLab

# Overview

Background: What are EPICS IOCs?

Motivation: Why do we want simulated EPICS IOCs?

Methods: How do simulated EPICS IOCs work?

Demonstration


Future Plans

# Background

EPICS IOCs

# EPICS

The **Experimental Physics and Industrial Control System (EPICS)** comprises a set of software components and tools that can be used to create distributed control systems. EPICS provides capabilities that are typically expected from a distributed control system:

- Remote control & monitoring of facility equipment
- Automatic sequencing of operations
- Facility mode and configuration control
- Management of common time across the facility
- Alarm detection, reporting and logging
- Closed loop (feedback) control
- Modeling and simulation 
- Data conversions and filtering
- Data acquisition including image data
- Data trending, archiving, retrieval and plotting
- Data analysis
- Access security (basic protection against unintended manipulation)

Source: <https://docs.epics-controls.org/en/latest/index.html>

# Usefulness of an IOC

Input/Output Controllers (IOCs) enable interacting with hardware in a convenient way by providing:

- An abstraction layer
  - A motor controller or a temperature sensor from different vendors may have very different underlying software interfaces, IOCs provide a unified interface for interacting with hardware.
- Network-based control
  - Exposing hardware interfaces over a network allows us to build software downstream of IOCs!
- Modularity
  - Each IOC can be independently developed, configured, and tested.



# Example EPICS IOCs – Area Detector

AXIS detector control  
screen at CSX 

Used to control data acquisition.  
The IOC itself uses the vendor  
provided SDK to interact with the  
hardware.

Everything here, except the UI  
itself, is a Process Variable (PV).  
PVs describe a single aspect of  
the process or device under  
control.

XF:23ID1-ES{AXIS}cam1:

**Setup**

Asyn port **AXIS1**  
EPICS name **XF:23ID1-ES{AXIS}car**  
Manufacturer **Unknown**  
Model **Dhyana XF4040BSI**  
Serial Number **Unknown**  
Firmware Vers. **232fd100002022050900**  
SDK version **2.0.7.0**  
Driver Vers. **3.12.1**  
ADCore version **3.12.1**  
Debugging ☐

**Shutter**

Shutter mode **None**  
Status: Det. **Closed** EPICS **Closed**  
Open/Close **Open** **Close**  
Delay: Open **0.000** Close **0.000**  
EPICS shutter setup ☐

**Collect**

Exposure time **1.000** 1.000  
Acquire period **1.000** 1.000  
# Images **1** 1  
# Images complete **20004**  
Image mode **Continuous** Continuous  
Internal  
Software trigger **Disconnect**  
Retry on timeout **Disabled** Disabled  
# Retries **0** 0  
Acquire **Start** **Stop**  
# Queued arrays **0**  
Wait for plugins **No**  
Detector state **1**  
Status **Collecting**  
Image counter **0** 20009  
Image rate **1.00**  
Array callbacks **Enable** **Enable**

**Plugins**

All File ☐ ROI ☐  
Stats ☐ Other #1 ☐ Other #2 ☐

**Readout**

	X	Y
Sensor size	4096	4096
Binning	<b>None</b>	<b>None</b>
Region start	<b>0</b>	<b>0</b>
Region size	4096	4096
Reverse	<b>No</b>	<b>No</b>
Image size	4096	4096
Image size (bytes)	33554432	
Frame Speed	<b>Disconnect</b>	<b>Disconnect</b>
Bit Depth	<b>Disconnect</b>	<b>Disconnect</b>
Gain Mode	<b>HDR</b>	<b>HDR</b>
Frame Format	<b>Disconnect</b>	<b>Disconnect</b>

**Cooler**

CAMERA TEMPERATURE **OK**  
Temperature **1** 1 1.8  
Manual TEC Control **Disabled** Disabled  
Auto TEC Control **Disabled** Disabled  
Auto TEC Threshold **10** 10

**Buffers**

Buffers used **30**  
Buffers alloc/free **42** 12  
Memory max/used (MB) **0.0** 429.3  
Buffer & memory polling **1 second**  
Empty free list **Empty**

**Attributes**

File   
Macros   
Status **File not found**  
**More**

# Area Detector Common Plugins

AXIS detector  
common plugin  
control screen at  
CSX.

Used for downstream  
processing of array data  
such as regions of  
interest (ROIs),  
statistics,  
transformations, file  
writing, and much  
more...

Plugin name	Plugin type	Port	Enable	Blocking	Dropped	Free	Rate		
Image1	NDPluginStdArrays	AXIS1	Disable	Disable	No	0	5	0.00	More
PVA1	NDPluginPva	OVER1	Enable	Enable	No	0	21	1.00	More
PROC1	NDPluginProcess	AXIS1	Enable	Enable	No	0	21	1.00	More
PROC2	NDPluginProcess	AXIS1	Enable	Enable	No	0	21	1.00	More
TRANS1	NDPluginTransform	PROC1	Enable	Enable	No	0	21	1.00	More
TRANS2	NDPluginTransform	PROC2	Enable	Enable	No	0	21	1.00	More
CC1	NDPluginColorConvert	AXIS1	Disable	Disable	No	0	21	0.00	More
CC2	NDPluginColorConvert	AXIS1	Disable	Disable	No	0	21	0.00	More
OVER1	NDPluginOverlay	TRANS2	Enable	Enable	No	0	21	1.00	More
ROI1	NDPluginROI	TRANS1	Enable	Enable	No	0	21	1.00	More
ROI2	NDPluginROI	TRANS1	Enable	Enable	No	0	21	1.00	More
ROI3	NDPluginROI	TRANS1	Enable	Enable	No	0	21	1.00	More
ROI4	NDPluginROI	TRANS1	Enable	Enable	No	0	21	1.00	More
STATS1	NDPluginStats	ROI1	Enable	Enable	No	0	21	1.00	More
STATS2	NDPluginStats	ROI2	Enable	Enable	No	0	21	1.00	More
STATS3	NDPluginStats	ROI3	Enable	Enable	No	0	21	1.00	More
STATS4	NDPluginStats	ROI4	Enable	Enable	No	0	21	1.00	More
STATS5	NDPluginStats	AXIS1	Enable	Enable	No	0	21	1.00	More
SCATTER1	NDPluginScatter	AXIS1	Disable	Disable	No	0	21	0.00	More
GATHER1	NDPluginGather	AXIS1	Disable	Disable	No	0	21	0.00	More
ROISTAT1	NDPluginROIStat	AXIS1	Disable	Disable	No	0	21	0.00	More
CB1	NDPluginCircularBuff	AXIS1	Enable	Enable	No	0	21	1.00	More
ATTR1	NDPluginAttribute	AXIS1	Disable	Disable	No	0	21	0.00	More
FFT1	NDPluginFFT	AXIS1	Disable	Disable	No	0	21	0.00	More
CODEC1	NDPluginCodec	AXIS1	Disable	Disable	No	0	21	0.00	More
CODEC2	NDPluginCodec	AXIS1	Disable	Disable	No	0	21	0.00	More
FileNetCDF1	NDFileNetCDF	AXIS1	Disable	Disable	No	0	21	0.00	More
FileTIFF1	NDFileTIFF	AXIS1	Disable	Disable	No	0	21	0.00	More
FileJPEG1	NDFilePEG	AXIS1	Disable	Disable	No	0	21	0.00	More
FileNexus1	NDPluginFile	AXIS1	Disable	Disable	No	0	21	0.00	More
Disconnected	Disconnected	Disconr	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	More
FileHDF1	NDFileHDF5 ver1.10.1	AXIS1	Enable	Enable	No	0	3000	0.00	More
Disconnected	Disconnected	Disconr	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	More
Disconnected	Disconnected	Disconr	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	More
Disconnected	Disconnected	Disconr	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	More
Disconnected	Disconnected	Disconr	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	Disconnect	More

# Example EPICS IOCs – VLS-PGM

Variable Line Spacing (VLS) Plane Grating Monochromator (PGM) at CSX

Used to control the optical device via motors (and other parameters).

The screenshot displays the control interface for the 23-ID-1 VLS-PGM. The interface is organized into several panels:

- Mono Status:** Displays key parameters including Energy (929.954 eV), Energy Mode (VLS), Grating Density (100.00 l/mm), c - Value (1.218511), Mirror Coupling (Decoupled), Grating Coupling (Decoupled), and Grating Selection (Grating 2).
- Optical Parameters:** Shows Grating 2 and a Change button, with a link to the Engineering Screen.
- Energy Scan:** Includes Scan Status (Ready), Move Status (Idle), Energy Step (Energy Fly-Scan), and Energy (930.000 eV) with a STOP button.
- Motors:** This section, highlighted by a red arrow, controls the physical components. It includes:
  - Mirror Pitch:** Values 1.490497 deg and 1.490464 deg, with a 0.005000 deg range, STOP, More, and Positioned buttons.
  - Grating Pitch:** Values 1.637295 deg and 1.637286 deg, with a 0.005000 deg range, STOP, More, and Positioned buttons.
  - Mirror Translation:** Values 0.000 mm and -0.000 mm, with a 0.500 mm range, STOP, More, Disabled, Home, Kill Axis, and Enable buttons.
  - Grating Translation:** Values 60.100 mm and 60.097 mm, with a 0.005 mm range, STOP, More, Disabled, Home, Kill Axis, and Enable buttons.



# Ideally...

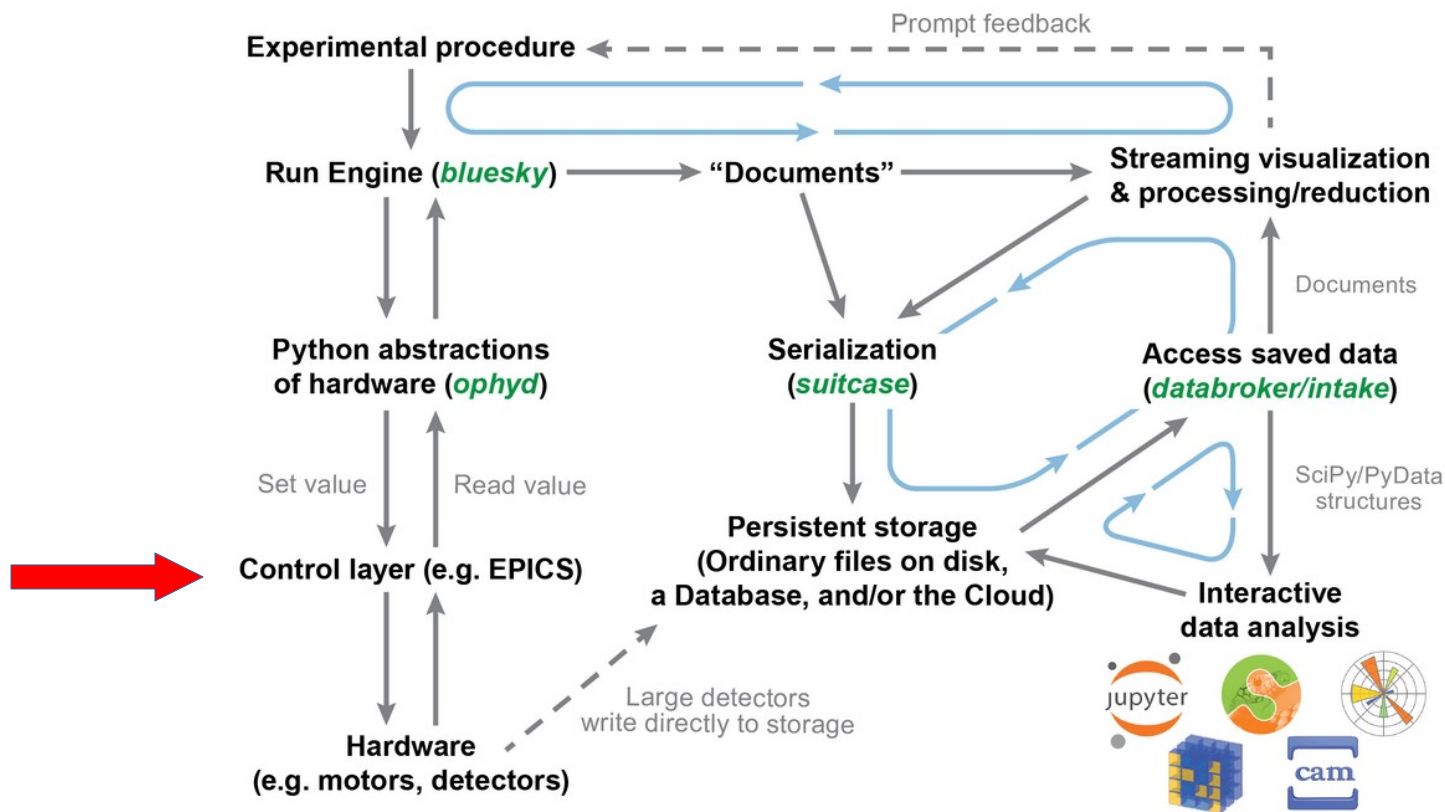
All of the data sent to and received from hardware at the beamline should go through an IOC.

NSLS-II makes heavy use of EPICS Channel Access (CA) as the communication protocol for doing so.

*PVAccess (PVA) is a newer protocol but we will not be covering it in this training.*

CA is the unifying protocol which we can use to create many downstream services...

# For Instance – Bluesky Ecosystem



Source: <https://blueskyproject.io/>

# Motivation

Simulated EPICS IOCs

# Simulation

Beamtime is costly and working with real devices is risky.

- Simulated IOCs allow us to test *downstream services* without having to interact with real hardware.

Continuous integration (CI) and continuous deployment (CD) makes for fewer errors in “production” code.

Software release cycles for these downstream services can therefore be ***more frequent*** and ***reliable***.

# Why EPICS CA?

EPICS Channel Access (CA) is the unifying protocol we use to interact with real hardware devices.

This level of the software stack is a great choice for simulation because:

- EPICS IOCs are very close to the real hardware.
- Most services are downstream from these EPICS IOCs.
  - Like AD Plugins, Ophyd, Bluesky, Tiled, Databroker, Prefect, etc.
- We can create general simulators for “kinds” of IOCs rather than try to simulate the diverse array of real hardware.
  - E.g. Every Area Detector IOC (that I am aware of) produces a 2D or 3D array, simulated or not.

# Methods

Types of Simulated EPICS IOCs



# Disclaimer

Simulated IOCs should ***not*** be deployed to any beamline server unless you have a very specific (and approved) reason for doing so.

The EPICS CA communication protocol does not distinguish between what is “simulated” traffic or “real” traffic which can lead to confusion and costly mistakes.

Simulated IOCs are, however, a very useful development tool, as we shall demonstrate later.

# Low vs High Fidelity Simulation

A sufficiently high fidelity simulation could be used for

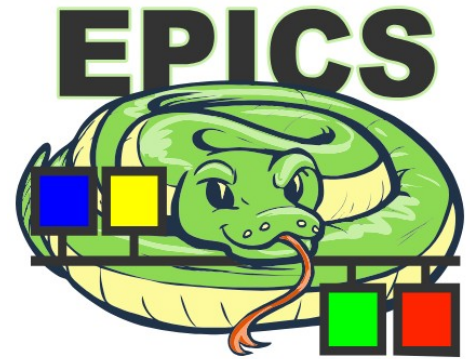
- Planning experiments
- Constructing new beamlines
- Training data-hungry machine learning models

Low fidelity simulations are more useful for automated testing.

We will be focusing on low fidelity simulation in this hour.

The second hour is showing a high fidelity simulator: XRT.





# Caproto

## EPICS Channel Access (CA) in pure Python

- Enables us to build IOCs in Python (rather than C/C++)
- Very useful for simple IOCs

Many example IOCs are available here:

[https://github.com/caproto/caproto/tree/main/caproto/ioc\\_examples](https://github.com/caproto/caproto/tree/main/caproto/ioc_examples)

A collaboration between SLAC LCLS and NSLS-II.

# Universal Low Fidelity IOC Simulator

One type of low fidelity simulation of an IOC is one that gives a default response for every request.

We call this a “Black Hole IOC” because it is a universal simulator that consumes all PV requests and returns a default response.

If we do ``caget TestPV:cam1:NumImages`` it will return ``0``

If we do ``caput TestPV:cam1:NumImages 5`` it will subsequently return ``5`` from the same ``caget`` command above

*Important Note: If an IOC such as this were to be deployed at the beamline, it could cause serious issues.*

# Black Hole IOC

This type of IOC is useful for a few reasons

- It allows us to build up higher fidelity simulators over-time
- It serves as a back-stop for all PV traffic that isn't actively used

The Black Hole IOC is most useful in combination with other *higher* fidelity simulators.

It is built using Caproto and Python!

```
(2025-2.0-py311-tiled) [thopkins1@... iocs]$ caget TestPV:cam1:NumImages
TestPV:cam1:NumImages
0
(2025-2.0-py311-tiled) [thopkins1@... iocs]$ caput TestPV:cam1:NumImages 5
Old : TestPV:cam1:NumImages      0
New : TestPV:cam1:NumImages      5
(2025-2.0-py311-tiled) [thopkins1@... iocs]$ caget TestPV:cam1:NumImages
TestPV:cam1:NumImages
5
```

```
(2025-2.0-py311-tiled) [thopkins1@... iocs]$ python spoof_beamline.py

*** WARNING ***
This script spawns an EPICS IOC which responds to ALL caget, caput, camonitor
requests. As this is effectively a PV black hole, it may affect the
performance and functionality of other IOCs on your network.

The script ignores the --interfaces command line argument, always
binding only to 127.0.0.1, superseding the usual default (0.0.0.0) and any
user-provided value.
*** WARNING ***

Press return if you have acknowledged the above, or Ctrl-C to quit.

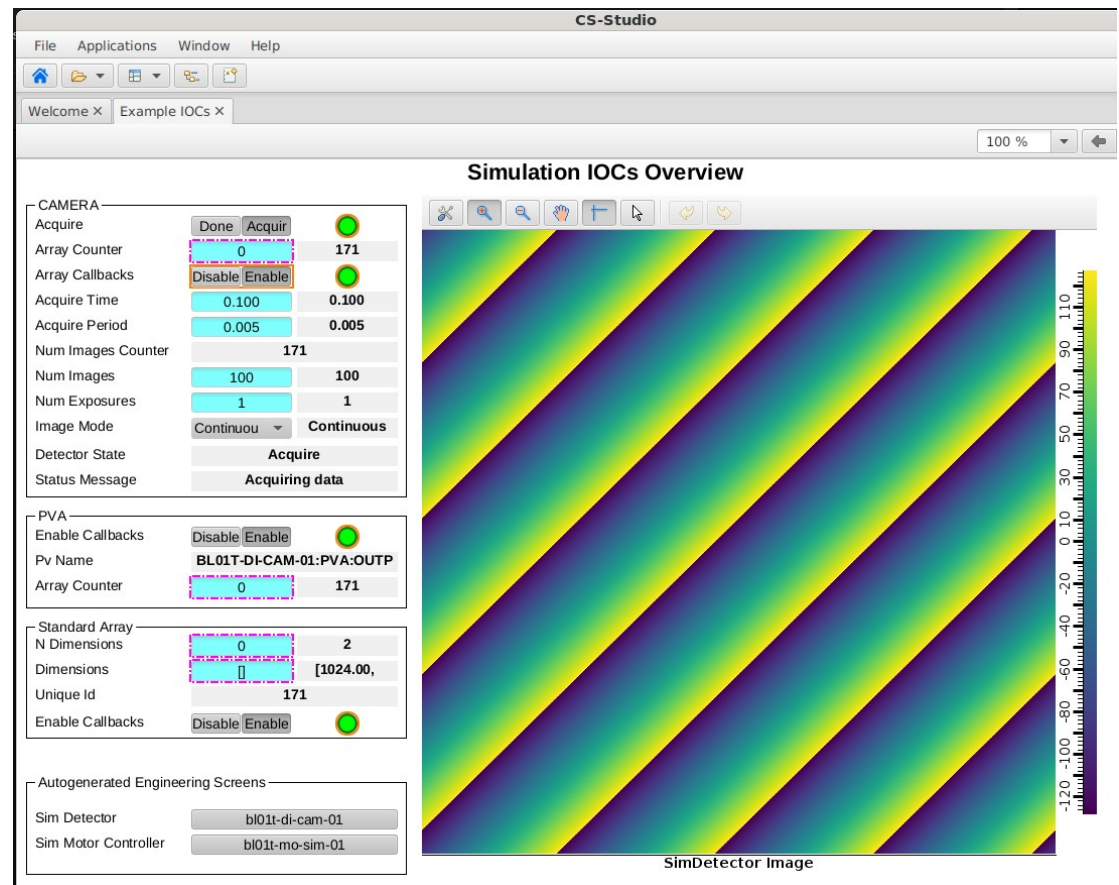
PV blackhole started

[I 14:41:01.865      server: 152] Asyncio server starting up...
[I 14:41:01.865      server: 159] Listening on 127.0.0.1:5064
[I 14:41:01.865      server: 196] Server startup complete.
```

# ADSim IOC

Simulation of a detector built into the Area Detector (AD) framework.

- Provides generic functionality that is similar to most detectors such as capturing a 2D frame at different frequencies.
- This is used for unit testing in Ophyd and we would like to expand it to the IPython profiles.
- Enables testing of AD plugins.





# Simulation of realistic movement for a motor

- Builds upon EPICS Motor Records, like most motor IOCs do
- Supports positioning, velocity control, limit handling, and homing.
- Allows for trajectory scanning along multiple axes.

Welcome x
Example IOCs x
bl01t-mo-sim-01 x
asynMotorAxis x
100 %

## asynMotorAxis

**Motor**  
Position 0.00000 degree 0.00000  
Stop Stop  
Tweak Forward Tweak Forward  
Tweak Step 1.00000 degrees  
Tweak Reverse Tweak Reverse

**Status**  
Ready   
Moving   
Forward   
Homed   
High Limit   
Low Limit   
Soft Limit   
Missed

**Homing**  
Home Forward Home Forward  
Home Reverse Home Reverse

**Calibration**  
Direction Pos  
Offset 0.00000 degrees  
Freeze Offset Variable  
Set Or Unset Use

**Status Detail**  
Status   
Dial Read Back Value 0.00000 degrees  
Read Back Settle Time 0.00000 degrees  
Dval Minus Drbv 0.00000 degrees  
Rval Minus Rrbv 0 degrees  
Raw Motor Position 0 degrees  
Raw Encoder Position 0 degrees  
Direction Of Travel 0 degrees  
Following Error <BL01T-MO-SIMC-01:M1:FERR  
Max Following Error <BL01T-MO-SIMC-01:M1:FERR

**Motion**  
Max Velocity 0.00000 degrees  
Base Velocity 0.10000 degrees  
Velocity 1.00000 degrees  
Acceleration 0.20000 sec  
Jog Velocity 1.00000 degrees  
Jog Acceleration 5.00000 degrees  
Backlash Distance 0.00000 degrees  
Backlash Velocity 1.00000 degrees  
Backlash Secs To Vel 0.20000 sec  
Move Fraction 1.00000 degrees  
Rery Deadband 0.01000 degrees  
Retry Count 10 degrees

**Resolution**  
Motor Step Size 0.01000 degrees  
Steps Per Revolution 200 steps/r  
Encoder Step Size 0.01000 degrees  
Readback Step Size 0.00000 degrees  
Use Encoder If Present No

**Motor Limits**  
User High Limit 20000.00000 degrees  
User Low Limit -20000.00000 degrees  
Dial High Limit 20000.00000 degrees  
Dial Low Limit -20000.00000 degrees  
Limit Violation   
At High Limit Switch   
At Low Limit Switch   
At Raw High Limit   
At Raw Low Limit

# Low Fidelity Simulators

The Black Hole IOC, ADSim, and MotorSim are all considered “low fidelity” simulators because they operate in complete isolation from each other.

Changing the motor positions of simulated motors have no effect on the output of ADSim.

Furthermore, the data that is output from ADSim is a fixed pattern that is generated.

# Partially Simulated Beamline

The end result of combining:

- Black Hole IOC
- ADSim
- MotorSim

Leads to something that we can use to test basic motor and detector functionality directly from IPython profiles that are used on real hardware.

Furthermore, it allows us to verify that downstream services are compatible with each other when these services are updated.

# Containerization of IOCs

Since IOCs can have a large number of dependencies, we use Docker/Podman to deploy IOCs as containers.

- This allows us to compose and deploy multiple IOCs within the same network

We re-use the existing effort by Diamond Light Source:  
<https://github.com/epics-containers/>

- Particularly, example-services, which aims to provide a containerized simulated beamline:

<https://github.com/epics-containers/example-services>

- At the EPICS Collaboration Meeting in September 2024, there was an EPICS in containers workshop that goes over the example-services above:

[https://controlssoftware.sns.ornl.gov/training/2024\\_EPICS/](https://controlssoftware.sns.ornl.gov/training/2024_EPICS/)

# CI/CD with Simulated IOCs

We are rolling out CI/CD pipelines via GitHub Actions to “profile\_collection” repositories.

- Start out with Black Hole IOC for basic compatibility and Python interpreter checks

## Features

Black Hole IOC	✓
Downstream Services (Bluesky, Tiled, Redis, etc.)	✓
ADSim	✗
MotorSim	✗
Higher Fidelity Simulations	✗

# CSX CI/CD

We use GitHub Actions in csx-profile-collection to test new changes against the Black Hole IOC.

- This allows us to test that the profile can load and the Ophyd objects can connect without error.
- It's a small step in the process toward higher fidelity automated integration testing.

← Test IPython beamline profiles

✓ Replace Azure Pipelines with GitHub Actions for CI (#103) #22

## Summary

### Jobs

- ✓ csx-py3.10
- ✓ csx-py3.11
- ✓ csx-py3.12

### Run details

🕒 Usage

📄 Workflow file

Triggered via push last week  
thomashopkins32 pushed → a2b3708 main Status Success Total duration 6m 13s Artifacts —

### ci.yml

on: push

Matrix: testing\_beamline\_profiles

- ✓ csx-py3.10 6m 10s
- ✓ csx-py3.11 5m 40s
- ✓ csx-py3.12 5m 40s

csx-profile-collection / .github / workflows / ci.yml

mrakitin and thomashopkins32 Replace Azure Pipelines with GitHub Actions for CI (#103) ✓

Code Blame 50 lines (46 loc) · 1.59 KB

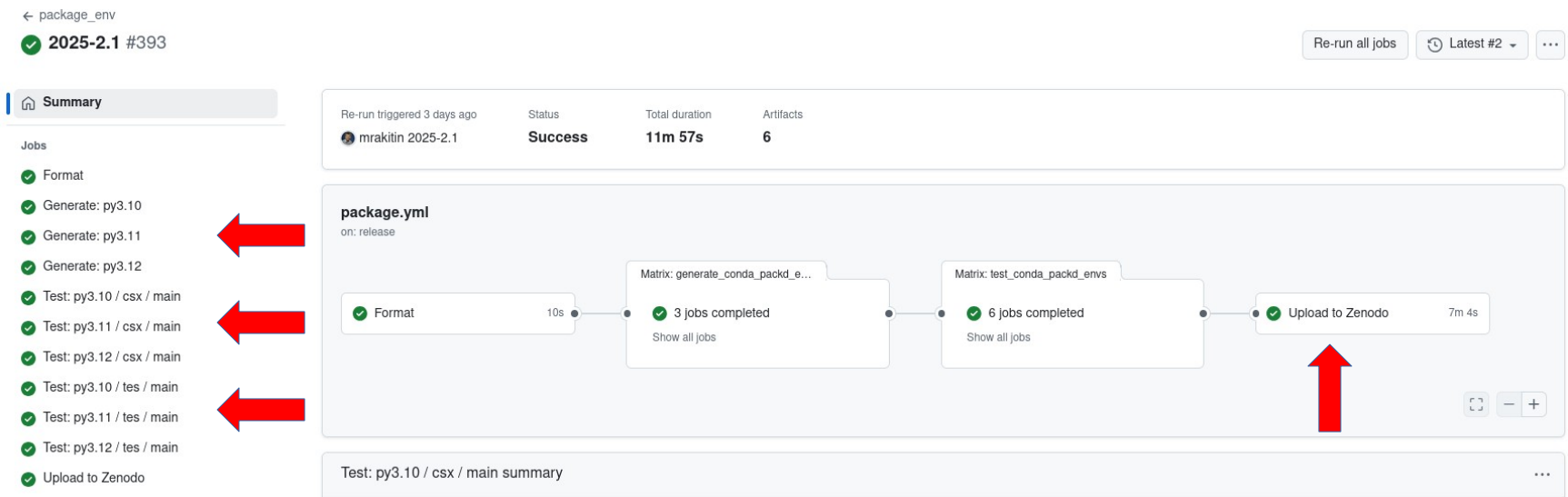
```
1 name: Test IPython beamline profiles
2
3 on:
4   push:
5     branches:
6       - main
7   pull_request:
8   workflow_dispatch:
9   workflow_call:
10  inputs:
11    version: # the variable you can use in place of a matrix
12    required: true
13    type: string
14
15 jobs:
16   testing_beamline_profiles:
17     name: ${ matrix.repos.beamline-acronym }-py${ matrix.zenodo.python }
18     strategy:
19       fail-fast: false
20     matrix:
21       repos:
22         - org: "NSLS2"
23           repo: "csx-profile-collection"
24           branch: ""
25           beamline-acronym: "csx"
26       zenodo:
27         - url: "https://zenodo.org/records/15171538/files/2025-2.0-py310-tiled.tar.gz"
28           md5: "bf2ecadce50394b4f44767fd53f76eb6"
29           python: "3.10"
30         - url: "https://zenodo.org/records/15171538/files/2025-2.0-py311-tiled.tar.gz"
31           md5: "3d705483eab31648fae7cf0ce9a77e72"
32           python: "3.11"
33         - url: "https://zenodo.org/records/15171538/files/2025-2.0-py312-tiled.tar.gz"
34           md5: "23f5634eda0e3207594c3529c2ade1cc"
35           python: "3.12"
36
37 runs-on: ubuntu-latest
38 steps:
39   - name: Checkout repository
40     uses: actions/checkout@v4
41
42   - name: Run Integration Tests Against Beamline Profiles
43     uses: NSLS2/gha-beamline-integration-test@2025-1.0-test
44     with:
45       conda_env_url: "${ matrix.zenodo.url }"
46       conda_env_md5: "${ matrix.zenodo.md5 }"
47       org: "${ matrix.repos.org }"
48       repo: "${ matrix.repos.repo }"
49       branch: "${ matrix.repos.branch }"
50       beamline-acronym: "${ matrix.repos.beamline-acronym }"
```



# Conda Environment Release Process

CSX's example of a CI/CD pipeline can be re-used to test all of the beamline profiles at NSLS-II.

- When we generate a new Conda environment, it must pass checks against a few reference beamlines before being uploaded.



# CI/CD in Ophyd

Ophyd uses ADSim to test its core AD framework support.

```
81 - name: Start Docker
82   run: |
83     sudo systemctl start docker
84     docker version
85     docker compose --version
86
87 - name: Create AD data directories
88   run: |
89     mkdir -p /tmp/ophyd_AD_test/
90     python scripts/create_directories.py /tmp/ophyd_AD_test/data1
91
92 - name: Clone epics-services-for-ophyd
93   run: |
94     git clone https://github.com/bluesky/epics-services-for-ophyd.git ~/epics-services-for-ophyd
95
96 - name: Start docker containers
97   run: |
98     source ~/epics-services-for-ophyd/environment.sh
99     docker compose -f ~/epics-services-for-ophyd/compose.yaml up -d
100
101 - name: Wait for docker containers to start
102   run: |
103     sleep 20
104
105 - name: Test with pytest
106   run: |
107     source ~/epics-services-for-ophyd/environment.sh
108     pytest -k "${TEST_CL}"
109
110 ...
```

[epics-services-for-ophyd / services / adsim\\_ioc / config / ioc.yaml](#)

jennmald move port to be consistent with other entries

Code Blame 152 lines (129 loc) · 2.88 KB

```
1 # yaml-language-server: $schema=https://github.com/epics-
2
3 ioc_name: "${ _global.get_env('IOC_NAME') }"
4 description: Example simulated camera for ophyd adsim
5
6 entities:
7   - type: devIocStats.iocAdminSoft
8     IOC: "${ ioc_name | upper }"
9
10  - type: ADSimDetector.simDetector
11    PORT: cam
12    P: "XF:31IDA-BI{Cam:Tbl}"
13    R: ":cam1:"
14    DATATYPE: 0
15    WIDTH: 1024
16    HEIGHT: 1024
17
18  - type: ADCore.NDR0I
19    PORT: ROI1
20    P: XF:31IDA-BI{Cam:Tbl}
21    R: ":ROI1:"
22    NDARRAY_PORT: cam
23
24  - type: ADCore.NDR0I
25    PORT: ROI2
26    P: XF:31IDA-BI{Cam:Tbl}
27    R: ":ROI2:"
28    NDARRAY_PORT: cam
29
30  - type: ADCore.NDR0I
31    PORT: ROI3
32    P: XF:31IDA-BI{Cam:Tbl}
33    R: ":ROI3:"
34    NDARRAY_PORT: cam
35
36  - type: ADCore.NDR0I
37    PORT: ROI4
```

# Demonstration

Showing simulated EPICS IOCs for CSX and CMS

# Future Plans

How we plan to use simulated EPICS IOCs

# Usability of the Black Hole IOC

Since the Black Hole IOC consumes all PV traffic, we have to have a way to settle conflicts between this IOC and others we plan on running

- Right now, we have a very crude way of doing this and we are working on improving this
- Suggestions are welcome!

# Minimizing Friction

Can we get to the point where spinning up a simulated beamline is as easy as clicking a button?

```
$ SIM=1 bsui
Starting simulated beamline...
Starting local tiled server...
Starting local redis server...
Starting local mongodb...
Loading IPython profile...
```

VS.

```
252 - name: Configuring Defaults (pylog, databroker, and kafka)
253 shell: bash -leo pipefail (0)
254 run: |
255   echo "::group::pylog configuration"
256   wget https://raw.githubusercontent.com/NSLS-II/profile-collection-ci/master/configs/pylog.conf -O $HOME/.pylog.conf
257   cat $HOME/.pylog.conf
258   echo "::endgroup::"
259
260   echo "::group::Classic databroker v0/v1 configuration"
261   databroker_conf_dir="$HOME/.config/databroker"
262   databroker_bl_conf="$BEAMLINE_ACRONYM.yml"
263   mkdir -p $(databroker_conf_dir)
264   wget https://raw.githubusercontent.com/NSLS-II/profile-collection-ci/master/configs/databroker.yml -O $(databroker_conf_dir)/$(databroker_bl_conf)
265   cp -v $(databroker_conf_dir)/legacy.config.yml $(databroker_conf_dir)/$(databroker_bl_conf)
266   cat $(databroker_conf_dir)/legacy.config.yml
267   cat $(databroker_conf_dir)/$(databroker_bl_conf)
268   echo "::endgroup::"
269
270   echo "::group::Tiled profile configuration"
271   tiled_profiles_dir="$HOME/.config/tiled/profiles"
272   mkdir -p $(tiled_profiles_dir)
273   cat << EOF > $(tiled_profiles_dir)/profiles.yml
274   $(BEAMLINE_ENDSTATION-$BEAMLINE_ACRONYM-local):
275     direct:
276       authentication:
277         allow_anonymous_access: true
278       trees:
279         - tree: databroker.mongo.normalized:Tree.from_uri
280           path: /
281           args:
282             uri: mongodb://localhost:27017/metadatastore-local
283             asset_registry_uri: mongodb://localhost:27017/asset-registry-local
284   EOF
285   cat $(tiled_profiles_dir)/profiles.yml
286   echo "::endgroup::"
287
288   echo "::group::Kafka configuration"
289   cat << EOF > kafka.yml
290   ...
291   abort_run_on_kafka_exception: false
292   bootstrap_servers:
293     - localhost:9092
294   runtime_producer_config:
295     security.protocol: PLAINTEXT
296   EOF
297
298   echo "$@: Placing kafka config in /etc/bluesky"
299   sudo mkdir -p /etc/bluesky
300   sudo mv -v kafka.yml /etc/bluesky/kafka.yml
301   cat /etc/bluesky/kafka.yml
302   echo "::endgroup::"
303   ...
```

```
303
304 - name: Configuring Redis
305 uses: supercharge/redis-github-action@1.8.0
306 with:
307   redis-version: latest
308
309 - name: Configuring Mongo
310 uses: supercharge/mongodb-github-action@1.11.0
311 with:
312   mongodb-version: '4.4'
313
314 - name: Add info DNS entries for local Redis and Mongo
315 shell: bash -leo pipefail (0)
316 run: |
317   echo "::group::Setting up DNS entries"
318   # Per https://stackoverflow.com/a/66982842:
319   beamline_redis_address="info.$(BEAMLINE_ACRONYM).nsls2.bnl.gov"
320   export BEAMLINE_REDIS_ADDRESS="$(beamline_redis_address)"
321   echo "BEAMLINE_REDIS_ADDRESS=$(BEAMLINE_REDIS_ADDRESS)" >> $GITHUB_ENV
322   sudo echo "127.0.0.1 $(BEAMLINE_REDIS_ADDRESS)" | sudo tee -a /etc/hosts
323   for i in {1..3}; do
324     sudo echo "127.0.0.1 mongo$(i).nsls2.bnl.gov" | sudo tee -a /etc/hosts
325   done
326   cat /etc/hosts
327   echo "::endgroup::"
328
329   echo "::group::Testing connections"
330   echo "Pinging $(BEAMLINE_REDIS_ADDRESS)..."
331   ping -c 5 $(BEAMLINE_REDIS_ADDRESS)
332   echo "Telnetting to $(BEAMLINE_REDIS_ADDRESS):6379..."
333   echo "" | telnet $(BEAMLINE_REDIS_ADDRESS) 6379
334   for i in {1..3}; do
335     echo "Pinging mongo$(i).nsls2.bnl.gov..."
336     ping -c 5 mongo$(i).nsls2.bnl.gov
337     echo "Telnetting to mongo$(i).nsls2.bnl.gov:27017..."
338     echo "" | telnet mongo$(i).nsls2.bnl.gov 27017
339   done
340   echo "::endgroup::"
341
342 - name: Insert reasonable values for cycle and data session to redis
343 shell: bash -leo pipefail (0)
344 run: |
345   echo "::group::Installing redis-cli and setting cycle and data session"
346   sudo apt install redis-tools
347   redis-cli -n $(BEAMLINE_REDIS_ADDRESS) set "$(BEAMLINE_REDIS_PREFIX)cycle" "${2025-2}"
348   redis-cli -n $(BEAMLINE_REDIS_ADDRESS) set "$(BEAMLINE_REDIS_PREFIX)data_session" "${pass-000000}"
349   redis-cli -n $(BEAMLINE_REDIS_ADDRESS) --scan
350   echo "::endgroup::"
351
352 - name: Create nsls2 tiled profile
353 shell: bash -leo pipefail (0)
354 run: |
355   echo "::group::Creating NSLS2 tiled profile"
356   conda activate $HOME/env
357   tiled_profile create --name nsls2 http://127.0.0.1:8000
358   echo "::endgroup::"
359
360 - name: Start Tiled Server
361 shell: bash -leo pipefail (0)
362 run: |
363   echo "::group::Starting Tiled server"
364   conda activate $HOME/env
365   export TILED_API_KEY=secret
366   echo "TILED_API_KEY=$(TILED_API_KEY)" >> $GITHUB_ENV
367   nohup tiled serve catalog --temp --api-key=$(TILED_API_KEY) &
368   sleep 10
369   echo "::endgroup::"
370
371 - name: Create TLA tiled profile
372 shell: bash -leo pipefail (0)
373 run: |
374   echo "::group::Creating TLA tiled profile"
375   conda activate $HOME/env
376   command=
377   from tiled.client import from_profile
378   tla = os.getenv('BEAMLINE_ACRONYM', 'xyz').lower()
379   TLA = tla.upper()
380   client = from_profile('nsls2', api_key=os.getenv('TILED_BLUESKY_WRITING_API_KEY_TLA'), 'secret')
381   client.create_container('raw', create_container('raw', specs=[CatalogDBBlueskyRuns]))
382   print(client[tla]['raw'])
383   python3 -c "$command"
384   echo "::endgroup::"
385
386 - name: Apply auto settings
387 shell: bash -leo pipefail (0)
388 run: |
389   echo "::group::Applying auto settings"
390   if [ -f "$CLONED_REPO/.ci/auto_settings.sav" ]; then
391     while IFS= read -r line; do
392       if [ -n "$line" ]; then
393         echo "Applying $line"
394         caproto-put $line
395       fi
396     done < "$CLONED_REPO/.ci/auto_settings.sav"
397   else
398     echo "No auto settings script found in $(CLONED_REPO)/.ci/auto_settings.sav"
399   fi
400   echo "::endgroup::"
401
402 - name: Activate IPython profile and run tests
```



# Rolling Out CI/CD for Beamlines

We plan to use the Black Hole IOC as a starting point for CI/CD testing for all beamline IPython profiles.

- This will enable us to perform very simple runtime checks

Next, we plan on integrating ADSim and MotorSim to automatically test actual Bluesky plans.

# More Simulators for Testing

Can we develop higher fidelity simulators for CI/CD testing?

- Instead of having ADSim that tries to simulate area detectors generally, could we implement an ADPilatusSim that is closer to how the Pilatus series of detectors works?
- Some detectors have additional functionality beyond what ADSim provides that could be useful to simulate

# How you can help

- Use Git and push your changes to GitHub
  - If you work on any beamline software, keep your repository up-to-date following Git/GitHub best practices
  - Your code should soon be automatically tested
- Share simulators with us if you have them!
  - We can help integrate them into automated testing pipelines
- Use these tools for development and give us feedback on what works well and what doesn't
  - Where is the most friction? Where is this not useful yet?

# References

- EPICS Homepage: <https://epics-controls.org/>
- Caproto Homepage: <https://caproto.github.io/caproto/v1.2.0/>
- Black Hole IOC:  
[https://github.com/caproto/caproto/blob/main/caproto/ioc\\_examples/pathological/spoof\\_beamline.py](https://github.com/caproto/caproto/blob/main/caproto/ioc_examples/pathological/spoof_beamline.py)
- ADSimDetector: <https://github.com/areaDetector/ADSimDetector>
- MotorSim: <https://github.com/epics-motor/motorMotorSim>
- Beamline Integration Test GitHub Action: <https://github.com/NSLS2/gha-beamline-integration-test>
- EPICS Containers: <https://github.com/epics-containers/>
- EPICS Collaboration Meeting on EPICS Containers:  
[https://controlssoftware.sns.ornl.gov/training/2024\\_EPICS/](https://controlssoftware.sns.ornl.gov/training/2024_EPICS/)

Thank you to all of the contributors who make projects like this possible! This is a combination of a lot of software that took many years to put together.

# Questions?