3-ID-C BITS

5-minute introduction

Presented 2026-06-01

What is this?

- A **Bluesky Instrument (BITS)** package for beamline 3-ID-C
- Python package name: `id3c`
- Built on `apsbits` (the APS BITS framework)
- Provides SPEC-style command-line scanning with **Bluesky**

What's installed today

- **Motors:** `sample_stage` (xprime/base_y/zprime/omega), `detector_stage` (det_x/eiger_y/eiger_z), `laser_optics` (us/ds)
- **Shutter:** `shutter` -- A-station PSS
- **Detector:** `eiger2` -- Eiger2 500k (HDF5 plugin pending)
- **Simulators:** `sim_motor`, `sim_det` for verification
- **Interlock:** `omega` <-> `laser_optics` (Python-session only)

The one rule

Plans go through `RE(...)`. Direct ophyd calls don't.

```
RE(bps.mv(sample_stage.xprime, 12.3))   # plan -- use RE
sample_stage.xprime.position             # data -- no RE
laser_optics.is_out                   # data -- no RE
RE(laser_optics.move_out())           # plan method -- use RE
```

Forgetting `RE(...)` silently does nothing. Our plans print a warning shortly after you press Enter, so you'll know to retype.

Where to learn more

- **Docs site:** https://bcda-aps.github.io/3idc-bits/
- **Cheat sheet:** `reference/cheat_sheet.md`
- **From SPEC:** `tutorials/spec_to_bluesky.md`
- **From EPICS:** `tutorials/epics_to_ophyd.md`
- **Bluesky Office Hours:** Every Wednesday, 2-3 pm on Teams

Try it

```
conda activate 3idc-bits
ipython
```

These plans use simulators. They do not use the 3-ID-C hardware.

```
from id3c.startup import *
RE(sim_print_plan())
RE(sim_count_plan())
RE(sim_rel_scan_plan())
```

Questions: https://github.com/BCDA-APS/3idc-bits/issues