3-ID-C BITS

Bluesky Instrument for beamline 3-ID-C

15-minute introduction

Presented 2026-06-01

Agenda

1. What is BITS, what is in this repo
2. Why Bluesky (vs SPEC, vs bare EPICS)
3. The one rule: `RE(plan(...))` vs direct calls
4. What's installed today
5. The omega <-> laser_optics interlock
6. Where the docs live, where to get help

What is BITS

- **Bluesky Instrument** -- a deployable Python package built on the `apsbits` framework
- Provides a **command-line scanning environment**
- Each beamline gets its own BITS package; ours is `id3c`
- Repo: https://github.com/BCDA-APS/3idc-bits

This deck is for the 3-ID-C team coming on-board to Bluesky. You already know your hardware; this is about the new software wrapper.

Why Bluesky?

You get...	...in exchange for
Structured metadata (UID, scan_id, run docs)	More verbose syntax than SPEC
Pause / resume mid-scan	A learning curve
Live plots + tables for free	New mental model (plans, RE)
Tiled-backed catalog of all runs	A new IPython session per shift
Generic suspenders (beam dump etc.)	A larger software stack

The trade is worth it for reproducibility, recovery, and post-experiment data access. We will not pretend it's "easier" than SPEC -- it is more *capable*.

The mental model

In SPEC:

```
SPEC> mv samx 5
```

In Bluesky:

```
RE(bps.mv(sample_stage.xprime, 5))
```

`bps.mv(...)` returns a **description** (a generator). `RE(...)` **runs** the description. If you type just `bps.mv(...)`, the motor does not move.

The one rule

Wrap in `RE(...)`	Call directly
`bps.mv(motor, 5)`	`motor.position`
`bp.scan([det], motor, 0, 10, 11)`	`motor.user_readback.get()`
`laser_optics.move_out()`	`laser_optics.is_out`
any `@plan`-decorated function	`motor.read()`
	`shutter.open()`
	`cat[-1].primary.read()`

Rule: returns a *generator* -> use `RE`; returns *data* -> call directly.

"Did nothing" -- the most common bug

```
sim_print_plan()            # WRONG -- no RE -- nothing happens
```

Our plans are decorated with `bluesky.utils.plan`. This makes the bare call print a warning shortly after you press Enter:

```
RuntimeWarning: plan `sim_print_plan` was never iterated,
                did you mean to use `yield from`?
```

That warning is your cue to retype with `RE(...)`.

What's installed today

```
%wa motors          # list every motor by label
%wa baseline        # list devices in the baseline stream
%wa                 # list everything
```

- `sample_stage`: xprime, base_y, zprime, **omega** (interlocked)
- `detector_stage`: det_x, eiger_y, eiger_z
- `laser_optics`: us, ds (interlocked)
- `shutter`: 3ida:shutterC (A-station PSS shutter)
- `eiger2`: Eiger2 500k area detector
  (HDF5 file plugin pending; see `devices.yml` FIXMEs)
- `sim_motor`, `sim_det`: simulators for verification

The omega <-> laser_optics interlock

Bidirectional, Python-session-only:

- `sample_stage.omega` is blocked unless `laser_optics.is_out`.
- `laser_optics.us / .ds` are blocked while `omega` is moving.

Raises a `MotionInterlock` exception before any CA put, or stops the motion in flight if the condition changes mid-move.

Scope: Python only in this session. Does not disable EPICS PVs. Does not protect against MEDM jogs, `caput`, other Bluesky sessions, or a Python crash. IOC-level interlock is a separate (and welcome) future improvement.

A first session

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

```
from id3c.startup import *
%wa motors

RE(bps.mv(sample_stage.xprime, 0))                  # move
RE(bp.count([sim_det], num=5))                    # count
RE(bp.scan([sim_det], sim_motor, -5, 5, 11))  # scan

run = cat[-1]                                     # most recent run
run.primary.read()                                # xarray Dataset
```

Where the docs live

`docs/source/` is a Sphinx site, organized by Diátaxis:

- **tutorials/** -- *learning* (start here if new)
- **how_to/** -- *task* ("how do I add a device?")
- **reference/** -- *lookup* (cheat sheet, quick reference)
- **explanation/** -- *understanding* (why `RE`, why `yield from`, interlocks)

Build locally: `cd docs && make html`. Auto-deployed from `main` to `https://bcda-aps.github.io/3idc-bits/`.

What's *not* here yet

- **Queueserver** workflow -- the host scripts exist (`scripts/id3c_qs_host.sh`) but not yet documented for users.
- **Validated HDF5 image readback** -- the Eiger2 master-file + external-link path needs setup and end-to-end testing.
- **Diffraction tools** (`hklpy2` package) -- planned, not configured.
- **Custom plans for beamline workflows** -- to be added as needs emerge.

Getting help

- **Issues:** https://github.com/BCDA-APS/3idc-bits/issues
- **Cheat sheet:** keep a tab on `reference/cheat_sheet.md`
- **From SPEC:** the cross-walk in `tutorials/spec_to_bluesky.md`
- **From EPICS:** `tutorials/epics_to_ophyd.md`
- **Conventions for contributing:** `AGENTS.md` at the repo root
- **Bluesky Office Hours:** Every Wednesday, 2-3 pm on Teams

Questions now?