EXAMPLES FOR THE HYDRA COMMAND CENTER REST API · V1.0

Automation Test Script Library

23 runnable examples that turn every Command Center capability into a scripted, CI-usable test — copy one into your harness, or run it as-is.

Every script talks to the same API — base URL **http://127.0.0.1:8600**, prefix **/api/v1** — through one shared helper, **hydra_client.py**. Every response answers **{"success": true, "data": ...}** or **{"success": false, "error": ...}**, so a script that passes against the built-in simulator runs unchanged against a populated enclosure.

WHY A SCRIPT LIBRARY

Every example follows the same contract

Twenty-three scripts, ordered simplest to most advanced, all built on the same three rules — so a validation engineer can trust an unfamiliar script on sight, and a CI pipeline can treat any of them as a real gate.

Stdlib only

No **pip install**. Every script imports the shared **hydra_client.py** helper and talks to the API over **urllib**, so it runs on a bare Python 3.8+ interpreter — nothing to vet, nothing to pin.

CI-usable exit codes

0 all checks passed, **1** a real validation failure, **2** a connection or setup error — nothing was validated. A pipeline can tell "the drive failed" from "the bench wasn't wired up."

Safe by default

Destructive and disruptive scripts print a dry-run plan and exit **0** until you pass **--yes**. Anything they change is restored in a **finally** block, even on an aborted run.

QUICK START

From zero to PASS in two commands

Start the server with `hydractl serve --demo`, then run the first example against the simulator: `python examples/smoke_test/smoke_test.py --demo`. If that prints **PASS**, the API is reachable and every other example in this library will run the same way — against the simulator today, against a populated enclosure tomorrow.

23
Runnable examples, simplest to most advanced

0
pip installs required — stdlib + one shared helper

3
Exit codes: pass, fail, or setup error

5
Categories: foundational, protocol, stress, Quarch fixture, workloads & host agent

Foundational checks, protocol consoles & CI gates

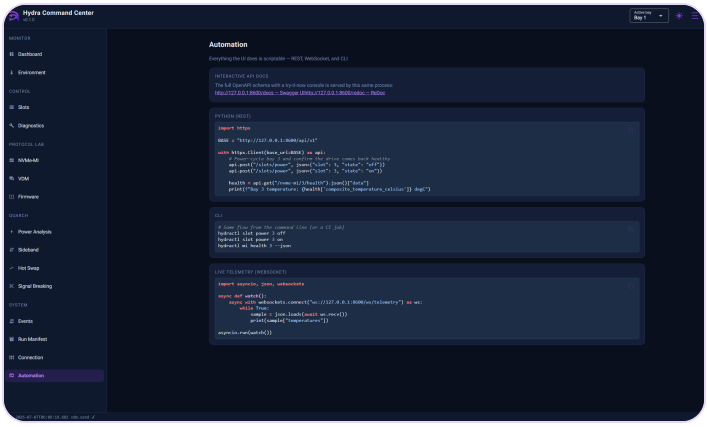
The first nine examples are non-destructive and fully demo-capable — run any of them cold, against the simulator, with no enclosure attached. They're the scripts most teams wire into a nightly job or a pre-merge gate.

FOUNDATIONAL & HEALTH CHECKS

SCRIPT	PURPOSE	DESTRUCTIVE?	DEMO-CAPABLE?	OPTIONAL HW
smoke_test	First script to run — bench-alive check: health, connection, bays, environment	No	**Yes**	—
slot_inventory	Bay population with model / serial / link and device-library match	No	**Yes**	—
link_check	Negotiated PCIe link vs. expected gen / width; catches downtraining	No	**Yes**	—
health_snapshot	NVMe-MI health + SMART threshold audit per bay	No	**Yes**	—
telemetry_capture	Live telemetry to CSV plus event-log watch	No	**Yes**	—
health_trend	Long-duration SMART trending / degradation flags	No	**Yes**	—

PROTOCOL CONSOLES & CI GATES

SCRIPT	PURPOSE	DESTRUCTIVE?	DEMO-CAPABLE?	OPTIONAL HW
validation_gates	Server-side /checks acceptance gates, JUnit output	No	**Yes**	—
cxl_device_check	CXL CCI identify / health / firmware / events	No (--clear-events needs --yes)	**Yes (slot 8)**	CXL Type-3
raw_nvme_mi	Protocol passthrough: discover / catalog / preview / raw / vdm / i2c	raw, vdm & i2c write need --yes	**Yes**	—



Same API, from the app. The in-app Automation page surfaces live links to the interactive API docs served by your own instance — Swagger UI at **/docs**, ReDoc at **/redoc**, and the raw schema at **/openapi.json** for generating a typed client.

Develop against --demo. The simulator honors the full API, so every script above is written and proven against it first — then runs unchanged the first time you point it at hardware.

Remote or HTTPS server? Pass **--host** / **--port**, or set `HCC_URL` (e.g. `https://hcc.lab:8600`). A server on a non-local bind needs a token: **--token** or `HCC_API_TOKEN`.

SHARED CLI FLAGS

Every script registers the same base flags through one shared **add_common_args()** helper, and connects automatically via **ensure_connected()** — the simulator for **--demo**, or the MCU for a given **--serial-port**.

`--HOST` `--PORT` `--TOKEN` `--SERIAL-PORT COM3` `--DEMO` `--JSON` `--YES`

Stress, endurance, firmware & Quarch-fixture scripts

These nine examples change enclosure state — fans, bay power, rails, firmware — or drive an optional fixture. Each one dry-runs by default and restores what it touched, even on an aborted run.

STRESS, ENDURANCE & FIRMWARE

SCRIPT	PURPOSE	DESTRUCTIVE?	DEMO-CAPABLE?	OPTIONAL HW
self_test_campaign	Device Self-Test launch + poll across bays	Disruptive	**Yes (flow)**	—
smart_delta_stress	SMART before/after delta across a stress window	Disruptive (optional)	**Yes (flow)**	—
thermal_soak	Fan-commanded thermal stress with abort ceiling	Disruptive	Limited	Real HW intended
hotplug_endurance	Bay power-cycle endurance loop	**Destructive**	Limited	Real HW intended
firmware_update	Validate → download → commit → verify FW state machine	**Destructive**	**Yes (full state machine)**	—

QUARCH FIXTURE & I3C

SCRIPT	PURPOSE	DESTRUCTIVE?	DEMO-CAPABLE?	OPTIONAL HW
quarch_power_margin	Rail margining / glitch injection	Disruptive	Simulated	Quarch module
quarch_bay_bringup	Breaker-paddle pin profile + drive power/health verify	Disruptive	Simulated	Quarch breaker
quarch_stream_capture	High-rate QIS power + sideband capture to CSV	No	**Yes**	Quarch module
i3c_ccc_probe	I3C target identification with per-step health net	Disruptive	**Yes**	Quarch optional (recovery)

SAFETY CONTRACT

Nothing destructive without --yes

Every destructive or disruptive script prints its plan and exits **0** untouched until you pass **--yes** — you see exactly what a run would change before it changes anything. Everything it does change — fan commands, bay power, rail voltages — is restored in a **finally** block, so a crashed or aborted run leaves the enclosure in its starting state rather than mid-stress.

QUARCH FIXTURE STATUS

Windows ready, Linux documented

WINDOWS · VENDOR USB DRIVER

The four Quarch-fixture scripts run in simulated mode with no module attached. On real hardware the work to date has run on the Windows USB driver. On Linux the module needs a one-time udev rule (`quarchpy.run debug --fixusb`) — documented from Quarch's own FAQ, not yet validated on a Linux HCC host.

THE CATALOG · PART 3 OF 3 · NEW IN V1.0

Workloads & in-band host-agent scripts

Five examples script the two newest Command Center features: the **Workloads** engine, and the **Host Agent** that joins what the Linux host sees in-band to each bay's out-of-band view. The host scripts need no hardware to try — the demo host agent answers every job, and bay 3 is deliberately the "sick" bay.

WORKLOADS & HOST AGENT

SCRIPT	PURPOSE	DESTRUCTIVE?	DEMO-CAPABLE?	OPTIONAL HW
workload_run	Run a saved workload (preset or custom) end to end — arm, start, follow the log, exit by verdict; Ctrl-C stops early through the server	As the template (armed only with --yes)	**Yes**	Quarch for the margin presets
host_inband_check	Host-side link / AER / SMART vs. the out-of-band view, per bay	No	**Yes**	Enrolled hcc-agent
host_fio_stream	Canned fio profile on the host, live 1 Hz stream to CSV	Disruptive (write profiles gated)	**Yes**	hcc-agent + fio
host_aer_watch	AER counter + kernel-log watch over a window	No	**Yes**	Enrolled hcc-agent
host_reset_recovery	Host-initiated controller / link / bus reset with recovery timing	Disruptive	**Yes**	hcc-agent with allow_writes

EXIT CODES FOR THESE SCRIPTS

SCRIPT	0	1	2
workload_run	Passed or review	Failed	Incomplete, or setup error
host_inband_check	Every checked bay passed	A host gate is BAD, or the host reports AER errors	No online host agent, or a bay has no host binding
host_fio_stream	Job done (and above --min-iops)	Job failed / timed out, or below --min-iops	No host sees the bay, write without --confirm, or HCC unreachable
host_aer_watch	No AER counter changed	At least one counter incremented	No online host sees the bay, or HCC unreachable
host_reset_recovery	Recovered, link no worse, no AER counter moved	Not recovered, link degraded, or AER counters incremented	No online host, host not opted in, or HCC unreachable

A **review** verdict from a workload means "look at the counters" — errors are counted and shown, never auto-failed — and exits 0 by design.

HOST_RESET_RECOVERY · THREE INTERLOCKS

A host-side reset never happens by accident

1. HCC requires `confirm=true` (the script passes it; the UI asks). **2.** The host admin must opt in with **allow_writes** in the agent's config — without it the agent refuses whatever HCC sent. **3.** The agent fails closed: a mounted or held namespace is never reset or retrained, and a bus reset that would hit other functions under the same port is refused.

TALK TO US

Bring your validation harness, we'll bring the API

Serial Cables, LLC · Englewood, CO · HYDRA 8-Bay Gen6 JBOF | serialcables.com