Hydra API

User's Manual

Python Library for Gen6 JBOF Control
and NVMe Validation

Serial Cables, LLC

www.serialcables.com

Document Version: 1.0
December 2025

1. Overview

The Hydra API (serialcables-hydra) is the official Python library for Serial Cables' Hydra Gen6 8-bay JBOF platform. It provides complete programmatic control over the enclosure, enabling automated NVMe and CXL device validation from Python scripts, CLI, or integration with existing test frameworks.

Designed for PCIe/NVMe validation engineers, the Hydra API abstracts low-level hardware communication while exposing the full capabilities of the Hydra platform for test automation and lab integration.

1.1 Key Features

- **Slot Control** — Per-slot power sequencing, hot-plug simulation, and EDSFF presence detection
- **Thermal Management** — SSD temperature monitoring and dual-fan RPM control
- **Power Telemetry** — Real-time current draw and voltage rail monitoring
- **NVMe-MI / MCTP** — Native sideband command support for out-of-band device management
- **LED Control** — Front-panel activity, error, and pass/fail indicator management
- **Telemetry Export** — Data streaming and logging for regression analysis
- **QuarchPy Integration** — Optional power analysis with Quarch PAM/Breaker mezzanine cards

1.2 Supported Hardware

The Hydra API supports the Serial Cables Gen6 Hydra JBOF (PCI6-ENC8-E3-08) with all EDSFF form factors: E3.S 2T, E3.L, and E3.L 2T. Optional Quarch mezzanine integration requires compatible paddle cards.

2. Installation

2.1 Requirements

- Python 3.8 or higher
- USB connection to Hydra enclosure
- Serial port drivers (typically auto-installed on modern OS)

2.2 PyPI Installation

Install the latest version from PyPI:

```
pip install serialcables-hydra
```

2.3 Optional Dependencies

For Quarch mezzanine support, install QuarchPy:

```
pip install quarchpy
```

3. Quick Start

3.1 Basic Connection

Connect to a Hydra enclosure and query basic status:

```
from serialcables_hydra import Hydra

# Auto-discover and connect
```

```
hydra = Hydra.discover()

# Or connect to specific port
hydra = Hydra("/dev/ttyUSB0")

# Get enclosure information
print(hydra.info())
print(hydra.status())
```

3.2 Slot Operations

```
# Check slot presence
for slot in range(8):
    present = hydra.slot_present(slot)
    print(f"Slot {slot}: {"Populated" if present else "Empty"}")

# Power control
hydra.slot_power_off(0)
hydra.slot_power_on(0)

# Hot-plug simulation
hydra.slot_hotplug_cycle(0, delay_ms=500)
```

4. API Reference

4.1 Hydra Class

The main interface for Hydra enclosure control.

Method	Description
`discover()`	Auto-discover and connect to Hydra enclosure
`info()`	Return enclosure information (model, serial, firmware)
`status()`	Return current enclosure status summary
`slot_present(n)`	Check if device is present in slot n (0-7)
`slot_power_on(n)`	Enable power to slot n
`slot_power_off(n)`	Disable power to slot n
`slot_hotplug_cycle(n)`	Perform hot-plug removal and insertion cycle
`get_temperature(n)`	Read temperature sensor for slot n (°C)
`get_power(n)`	Read power consumption for slot n (Watts)
`set_fan_rpm(rpm)`	Set enclosure fan speed (RPM)
`set_led(n, state)`	Control slot LED (activity, error, pass, fail)
`mctp_send(n, msg)`	Send MCTP message to device in slot n
`nvme_mi_cmd(n, cmd)`	Execute NVMe-MI command on slot n

4.2 Telemetry

The Hydra API provides real-time telemetry collection for logging and analysis.

```
# Start telemetry collection
hydra.telemetry.start(interval_ms=1000)

# Export to CSV
hydra.telemetry.export("test_run_001.csv")

# Stop collection
hydra.telemetry.stop()
```

5. Integration

5.1 Atlas3 Switch Integration

The Hydra API works seamlessly with the Atlas3 PCIe switch API for complete upstream/downstream control.

```
from serialcables_atlas3 import Atlas3
from serialcables_hydra import Hydra

atlas = Atlas3.discover()
hydra = Hydra.discover()

# Coordinate switch and JBOF for test sequence
atlas.link_disable(port=4)
hydra.slot_power_off(0)
# ... perform maintenance ...
```

5.2 Hermes Redriver Integration

Combine with the Hermes Redriver API for signal integrity tuning in Gen6 test setups.

```
from serialcables_hermes import Hermes
from serialcables_hydra import Hydra

hermes = Hermes.discover()
hydra = Hydra.discover()

# Tune redriver before powering slot
hermes.set_eq(channel=0, db=12.5)
hydra.slot_power_on(0)
```

5.3 QuarchPy Integration

For Hydra units equipped with Quarch PAM or Breaker mezzanine cards, use QuarchPy alongside the Hydra API for detailed power analysis.

```
import quarchpy
from serialcables_hydra import Hydra

hydra = Hydra.discover()
pam = quarchpy.PAM.discover()

# Correlate JBOF events with power measurements
pam.start_capture()
hydra.slot_hotplug_cycle(0)
pam.stop_capture()
```

6. Command Line Interface

The Hydra API includes a CLI for quick operations without writing Python code.

```
# Show enclosure status
hydra-cli status

# Power cycle slot 3
hydra-cli slot 3 power-cycle

# Set fan speed
hydra-cli fan --rpm 2500

# Start telemetry logging
hydra-cli telemetry --output test.csv --interval 500
```

7. Safety Considerations

- **Power Sequencing** — Always follow proper power-on/off sequencing to avoid device damage
- **Thermal Limits** — Monitor SSD temperatures and ensure adequate cooling before extended tests
- **Hot-Plug Testing** — Verify host system supports surprise removal before using hot-plug functions
- **ESD Protection** — Use proper ESD precautions when handling EDSFF devices

8. Support

- **Website:** www.serialcables.com
- **Email:** support@serialcables.com
- **Sales:** sales@serialcables.com
- **Phone:** +1 303-810-5110
- **PyPI Package:** pip install serialcables-hydra