Cnuas Facility Twin, Datasheet

Campus digital twin, OpenUSD scene generation, power model and live ORV3 telemetry

Document DS-CNU-030 • **Revision** B • **Issued** 23 August 2026 • **Status** Published

Item	Value
Part	`cnuas-facility`
Type	Facility digital twin and power model
Package	`cnuas-facility`, version 0.1.0
Version	`5c3d075`
Repo	PacketFive/cnuas

1. Overview

The facility twin generates a hyperscale AI and HPC data centre campus as an OpenUSD scene, computes its electrical and thermal load, and animates the scene from live readings taken off the emulated Open Rack V3 power shelf. It is what connects a rack of emulated hardware to the site scale question every operator eventually asks: how much power does this draw, and what has to be built to supply it.

The scene opens in any OpenUSD capable viewer, including NVIDIA Omniverse and Isaac Sim.

Key features

- Declarative campus specification, from site dimensions down to rack rows.
- Strict user JSON configuration with template generation and spatial validation before OpenUSD authoring.
- Three building footprints, the I, H and E shapes hyperscale operators build.
- Data halls with rack rows, cold and hot aisle spacing, and support rooms.
- Site features: yards, roads, plant compounds, groves and lakes.
- Power roll-up from rack to hall to building to campus, with plant sizing.
- Live sampling of the emulated Open Rack V3 shelf over Modbus, written back onto the scene as typed attributes.

2. Module map

Module	Responsibility
`spec.py`	Declarative campus, building and hall specification, and the reference campus
`config.py`	JSON template, strict loader, custom rack catalogue and site validation
`catalog.py`	Equipment catalogue, dimensions and power rating per rack and plant type
`plan.py`	Resolves a specification into placed rooms, racks, plant and roads
`layout.py`	Rectangle mathematics, building footprints, rack row layout
`geom.py`	Geometry and material primitives
`usd.py`	Writes the campus OpenUSD scene graph
`schema.py`	The `cnuas:` attribute namespace carried on the scene
`power.py`	Load roll-up and plant sizing

Module	Responsibility
`orv3.py`	Modbus client and register decoding for the power shelf
`live.py`	Samples the shelf and animates the scene
`units.py`	Named dimensional constants
`cli.py`	Command line front end

3. Command reference

Command	Purpose
`build`	Write the campus OpenUSD stage
`power`	Print the power roll-up
`rooms`	Print the room programme
`live`	Sample the Open Rack V3 shelf and animate the stage
`template`	Write the complete reference campus as editable JSON
`validate`	Validate a custom JSON campus without writing OpenUSD

4. Scene specification

Parameter	Reference value
Campus name	Cnuas Ridge AI Campus
Site footprint	1000 m by 800 m
Buildings	3, one each of the I, H and E footprints
Data halls	16 across the three buildings
Design power usage effectiveness	1.15
Target campus load	Close to 300 MW
Stage root	`/World`
Scene branches	`/World/Looks`, `/World/Site`, `/World/Yards`, `/World/Buildings`, `/World/Prototypes`
Default output format	`.usda`, with `.usdc` and `.usd` also supported

5. Equipment catalogue

Entry	Rated load
Cnuas Open Rack V3, 24 OpenU	6 kW
GB200 NVL72 class rack	132 kW
Storage rack	12 kW
Network rack	15 kW
Medium voltage switchgear, transformer, generator, coolant distribution unit, fan wall, indirect evaporative cooler, dry cooler	Sized by the power model

6. Power model

Quantity	Definition
Information technology load	Sum of rack ratings in scope
Facility load	Information technology load multiplied by the power usage effectiveness
Overhead	Facility load less information technology load
Heat rejected	Equal to the information technology load
Density	Load per unit floor area

Roll-up functions run at hall, building and campus scope. Plant counts are derived from the resulting load.

Plant	Unit size used for sizing
Generator	3000 kW
Cooler	3500 kW
Coolant distribution unit	2000 kW

7. Live telemetry

Item	Value
Source	Emulated Open Rack V3 power shelf
Protocol	Modbus remote terminal unit framing
Transport	Serial line or a TCP endpoint, by default port 3485 on the loopback address
Decoded from the power supply	Voltage, current, power, fan speed, temperature
Decoded from the battery backup	State of charge and unit state
Write back	Typed attributes in the `cnuas:` namespace on the corresponding scene primitives

The shelf itself is emulated by the BMC stack, which runs the power shelf firmware and presents the RS-485 segment.

8. Validation

Area	Test	Cases
Power roll-up and plant sizing	`tests/test_power.py`	8
Footprints, rectangles and rack rows	`tests/test_layout.py`	17
Scene graph generation, including custom-configuration read-back	`tests/test_usd.py`	14
Shelf sampling and scene animation	`tests/test_live.py`	21
Custom JSON round trip, validation, automatic plant sizing and CLI	`tests/test_config.py`	6

All 66 cases pass. See the Validation Matrix.

9. Operating notes

- The twin models geometry, load and telemetry. It does not perform computational fluid dynamics, electrical fault studies or structural analysis, and the plant counts are sizing estimates rather than engineering designs.
- Building footprints are limited to the I, H and E shapes in this revision.
- Custom configurations can add rack types, but plant item types remain the built-in electrical and mechanical catalogue in this revision.

10. Integration information

Item	Value
Repo	`PacketFive/cnuas`
Source	`facility/src/cnuas_facility/`
Entry point	`cnuas-facility`
Python	3.10 or later
Dependencies	OpenUSD Python bindings
Viewers	Any OpenUSD viewer, including NVIDIA Omniverse and Isaac Sim

11. Revision history

Revision	Notes
A	First publication. Reference campus, catalogue ratings and plant unit sizes read from the source.
B	Added the beginner workflow and strict custom-campus JSON template, validation, custom rack types, configurable roads and automatic per-building plant counts.