

Contents

power-hmc-exporter	1
Requirements	1
Building	1
Configuration	2
Running	2
Prometheus scrape configuration	3
Metrics	3
Example PromQL	5
Grafana dashboards	5
Diagnostics	6
Known characteristics	6
Architecture	6
Installing from the release tarball	6
Building a release	7
Editions	7
About	7

power-hmc-exporter

A Prometheus exporter for IBM Power Hardware Management Consoles (HMC), written in Go. It polls the HMC REST API and exposes managed-system, LPAR and VIOS metrics derived from the Performance and Capacity Monitoring (PCM) facility — including processor and memory utilization, per-adapter network counters, Shared Ethernet Adapters, physical Fibre Channel adapters and NPIV virtual FC adapters with cross-reference labels for capacity analysis.

Requirements

- An IBM Power HMC with the REST API enabled and PCM aggregation turned on for the managed systems of interest. Verified against **HMC V10 R3 SP1063**.
- TCP **12443** reachable from the exporter host to the HMC.
- A Linux x86_64 host. RHEL 9.x verified. The exporter binary is statically linked, no runtime dependencies on the target.
- **Go 1.25+** on the build host.

Building

```
make build      # statically linked binary in bin/power-hmc-exporter
make test
make all        # vet + test + build (default target)
```

Configuration

Configuration is read from environment variables prefixed PHMC_. A KEY=VALUE configuration file (systemd EnvironmentFile syntax) can be passed with -config <path>; variables already present in the environment take precedence over the file, so you can point -config at the same file systemd uses and still override individual settings ad hoc.

Variable	Required	Default	Description
PHMC_HMC_HOST	yes	—	HMC hostname or IP.
PHMC_HMC_USER	yes	—	HMC user (typically hscroot).
PHMC_HMC_PASSWORD	yes ¹	—	HMC user password.
PHMC_HMC_PASSWORD_FILE	no ¹	—	Path to a file holding the password (preferred over PHMC_HMC_PASSWORD).
PHMC_HMC_PORT	no	12443	HMC REST API port.
PHMC_INSECURE_SKIP_VERIFY	no	false	Skip TLS verification (HMC usually has a self-signed cert).
PHMC_LISTEN_ADDR	no	:9876	Exporter HTTP listen address.
PHMC_POLL_INTERVAL	no	60s	Interval between HMC poll cycles.
PHMC_REQUEST_TIMEOUT	no	30s	Timeout for each individual HMC API call.
PHMC_MAX_CONCURRENCY	no	6	Maximum number of concurrent HMC requests within a poll cycle. Lower to 1 for fully sequential polling.

¹ Either PHMC_HMC_PASSWORD or PHMC_HMC_PASSWORD_FILE is required.

Running

```
power-hmc-exporter -h          # show help, flags and all PHMC_ variables
power-hmc-exporter -version    # print the build's VCS revision
```

Manually, sourcing the environment file into the shell:

```
set -a; . /etc/sysconfig/power-hmc-exporter; set +a
./bin/power-hmc-exporter
```

...or letting the binary read that same file directly:

```
./bin/power-hmc-exporter -config /etc/sysconfig/power-hmc-exporter
```

HTTP endpoints: - GET /metrics — Prometheus metrics - GET /healthz — liveness probe

systemd deployment

A unit file and an environment file template are provided under `deploy/`.

```
# binary
install -m 755 bin/power-hmc-exporter /usr/local/bin/power-hmc-exporter

# service user (no shell, no home)
useradd --system --no-create-home --shell /sbin/nologin power-hmc-exporter

# configuration file – edit credentials before starting.
# The shipped unit reads it via -config, so it must be readable by the
# service account: mode 0640, group power-hmc-exporter.
install -m 640 -o root -g power-hmc-exporter \
    deploy/power-hmc-exporter.env.example \
    /etc/sysconfig/power-hmc-exporter
vi /etc/sysconfig/power-hmc-exporter

# systemd unit
install -m 644 deploy/systemd/power-hmc-exporter.service \
    /etc/systemd/system/
systemctl daemon-reload
systemctl enable --now power-hmc-exporter
journalctl -u power-hmc-exporter -f
```

To keep the secrets file strictly root-only (0600 root:root), drop the `-config` argument from the unit's `ExecStart` and add `EnvironmentFile=/etc/sysconfig/power-hmc-exporter` instead — `systemd` then reads the file as root and injects the variables.

Prometheus scrape configuration

```
scrape_configs:
  - job_name: vhmcc
    scrape_interval: 1m
    static_configs:
      - targets: ['<exporter-host>:9876']
```

A scrape interval matching `PHMC_POLL_INTERVAL` is recommended; shorter does not yield fresher data.

Metrics

Names below; consult `/metrics` for full descriptions. All metrics are gauges unless stated otherwise.

Exporter health - `power_hmc_up` - `power_hmc_exporter_build_info`{version, go_version} - `power_hmc_poll_duration_seconds` - `power_hmc_last_poll_timestamp_seconds` - `power_hmc_errors_total` (counter) - `power_hmc_system_pcm_sample_age_seconds`{system}

Inventory - `power_hmc_managed_systems` - `power_hmc_managed_system_info`{name, machine_type, model, machine, serial, state, processor_mode, firmware_level} — machine is the machine_type-model form (e.g. 9009-22A, matching IBM `prtconf`) - `power_hmc_lpar_info`{system, lpar, lpar_id, type, state, os_type, os_version, logical_serial, processor_mode, sharing_mode, migration_state, boot_mode, keylock_position, system_name, service_partition, bootable, remote_restart_capable} - `power_hmc_lpar_uptime_seconds`{system,

lpar} — wall-clock seconds since the LPAR booted - power_hmc_vios_info{system, vios, vios_id, state}

Managed system PCM (per system) - power_hmc_system_processor_units_total / _utilized / _available / _configurable - power_hmc_system_memory_total_megabytes / _available_megabytes / _configurable_megabytes / _assigned_to_lpars_megabytes

LPAR PCM (per system, lpar) - power_hmc_lpar_processor_units_utilized / _entitled / _max / _idle / _utilized_capped / _utilized_uncapped - power_hmc_lpar_virtual_processors - power_hmc_lpar_memory_logical_megabytes / _backed_physical_megabytes

LPAR network (per system, lpar, adapter, vlan_id, vios_id, vios, sea) - power_hmc_lpar_network_receive / _sent_packets / _received_bytes / _sent_bytes / _dropped_packets

vios is the serving VIOS name (resolved from vios_id), for filtering and grouping by VIOS without an id→name join.

LPAR NPIV VFC (per system, lpar, vfc, wwpn, vios_id, physical_port_wwpn, vios, fc) - power_hmc_lpar_vfc_reads / _writes / _read_bytes / _write_bytes / _transmitted_bytes / _running_speed_gbps

vios is the serving VIOS name and fc its physical FC device (e.g. fcs0), resolved from physical_port_wwpn, so NPIV traffic can be grouped per VIOS and per physical FC port without a join.

Note: on HMC V10 the VFC transmittedBytes field is reported as 0; use read_bytes + write_bytes for NPIV throughput.

LPAR VSCSI client (per system, lpar, vhost, vios_id, location) - power_hmc_lpar_vscsi_reads / _writes / _read_bytes / _write_bytes / _transmitted_bytes

Emitted for LPARs using virtual SCSI (boot disks, vSCSI-attached storage). The vhost label carries the serving VIOS-side vhost name (e.g. vhost0), matching the vhost label on power_hmc_vios_vscsi_* — together with vios_id, this is the join key for LPAR ↔ VIOS vhost cross-reference.

VIOS PCM (per system, vios) - power_hmc_vios_processor_units_utilized / _entitled / _max / _idle / _utilized_capped / _utilized_uncapped - power_hmc_vios_virtual_processors

Note: _max is the partition profile's maximum processing units (the DLPAR ceiling for entitled capacity). For an uncapped shared VIOS the real consumption cap is the virtual processor count (power_hmc_vios_virtual_processors), since a partition cannot consume more processing units than it has virtual processors. The HMC reports currentVirtualProcessors as 0 for VIOS partitions, so that metric falls back to maxVirtualProcessors. - power_hmc_vios_memory_assigned_megabytes / _utilized_megabytes

VIOS network — generic adapters (per system, vios, adapter, ent, type) - power_hmc_vios_network_receive / _sent_packets / _received_bytes / _sent_bytes / _dropped_packets

VIOS network — Shared Ethernet Adapters (per system, vios, sea, adapter, vlans) - power_hmc_vios_sea_received_packets / _sent_packets / _received_bytes / _sent_bytes / _dropped_packets - power_hmc_vios_sea_bridged_adapters_count (no vlans label)

The vlans label is the sorted, space-separated list of VLAN IDs carried by the SEA (e.g. "10 191 250"). It is derived from the VIOS virtual ethernet adapters (virtualEthernetAdapters[].vlanId) matched to the SEA by adapter slot — a stable PCM source, so the label does not flap.

VIOS storage — physical FC HBA ports (per system, vios, fc, wwpn, location) - power_hmc_vios_fc_reads / _writes / _read_bytes / _write_bytes / _transmitted_bytes / _running_speed_gbps / _ports

VIOS storage — VSCSI vhost adapters (per system, vios, vhost, location) - power_hmc_vios_vscsi_reads / _writes / _read_bytes / _write_bytes / _transmitted_bytes

Example PromQL

```
# Top 10 LPARs consuming a given physical FC port (NPIV)
topk(10, sum by (lpar) (
  power_hmc_lpar_vfc_read_bytes{physical_port_wwpn="<wwpn>"}
+ power_hmc_lpar_vfc_write_bytes{physical_port_wwpn="<wwpn>"}
))

# Top 10 LPARs behind a given Shared Ethernet Adapter
topk(10, sum by (lpar) (
  power_hmc_lpar_network_received_bytes{sea="ent11"}
+ power_hmc_lpar_network_sent_bytes{sea="ent11"}
))

# Aggregate SEA throughput per VIOS
sum by (system, vios, sea) (
  power_hmc_vios_sea_received_bytes + power_hmc_vios_sea_sent_bytes
)

# LPAR CPU consumption vs entitlement (>1 = borrowing uncapped)
power_hmc_lpar_processor_units_utilized
/ power_hmc_lpar_processor_units_entitled

# Recommended freshness alerts
time() - power_hmc_last_poll_timestamp_seconds > 180
power_hmc_system_pcm_sample_age_seconds > 900
power_hmc_poll_duration_seconds > 50
```

Grafana dashboards

Three dashboards are provided under `deploy/grafana/`. Import each via *Dashboards* → *Import* and select your Prometheus datasource at the prompt:

- `power-hmc-overview.json` — estate-wide overview.
- `power-hmc-per-system.json` — per managed-system drill-down (VIOS, SEA, physical FC, NPIV by LPAR).
- `power-hmc-per-lpar.json` — per-LPAR drill-down.

Optional: VIOS FC errors (requires aix-exporter)

The per-system dashboard has a *Fibre Channel* — *VIOS FC errors* section that charts per-VIOS Fibre Channel link- and resource-error counters (link failures, loss of sync/signal, CRC/error/dumped frames, DMA/command resource exhaustion, LIP/NOS events). These come from a separate **aix-exporter** (`aix_fcstat_*`) scraped into the *same* Prometheus, not from this exporter — the HMC PCM facility does not expose FC port error counters. The panels correlate the two exporters on the AIX node:

```
rate(aix_fcstat_LinkFailureCount[5m]) and on (instance) aix_lpar_lcpus{lpar="$viosname"}
```

and `on(instance)` keeps only the `aix_fcstat_*` series whose AIX node reports the selected VIOS partition name — `aix_lpar_*` carries an `lpar` label equal to the HMC partition name (`$viosname`), so the join needs no environment-specific hostname mapping and is value-preserving. Where no `aix-exporter` runs on a VIOS, these panels simply show *No data*; the rest of the dashboard is unaffected.

Diagnostics

A standalone tool `pcmdump` ships with the exporter for diagnosing the PCM payload directly:

```
go build -o bin/pcmdump ./cmd/pcmdump
./bin/pcmdump system <managed-system-uuid>           # inspect system PCM
./bin/pcmdump probe <managed-system-uuid>            # probe LPAR endpoints
./bin/pcmdump lpar <system-uuid> <lpar-uuid>          # inspect LPAR PCM
```

It uses the same `PHMC_*` environment variables.

Known characteristics

- **PCM Processed sample latency on HMC V10 ~6-7 minutes.** This is the HMC's intrinsic processing lag, not an exporter bug. Set staleness alerts at ~15 minutes (`power_hmc_system_pcm_sample_age` > 900). For near real-time data, the `RawMetrics/LongTermMonitor` endpoint (30s granularity) would be needed.
- **Per-LPAR ProcessedMetrics endpoint is nested under ManagedSystem** on HMC V10 (`/rest/api/pcm/ManagedSystem/{sys}/LogicalPartition/{lpar}/...`); the non-nested path documented for Power8 returns HTTP 500 on V10.
- **VFC transmittedBytes is reported as 0** in NPIV; rely on `read_bytes` + `write_bytes` for throughput.
- **Poll cycle is concurrent and not capped at the poll interval.** Each PCM document is a 2-step retrieval (Atom feed → JSON), so an estate with many LPARs would otherwise run hundreds of sequential calls and exceed the interval, truncating the cycle and leaving gaps in the graphs. The exporter fetches up to `PHMC_MAX_CONCURRENCY` requests in parallel and lets a busy cycle stretch the cadence rather than drop data. Watch `power_hmc_poll_duration_seconds`: it should stay comfortably below `PHMC_POLL_INTERVAL`. If residual single-sample gaps remain, enable *Connect null values* (`spanNulls`) on the Grafana time-series panels.

Architecture

- `cmd/power-hmc-exporter/` — main exporter binary.
- `cmd/pcmdump/` — PCM diagnostic CLI.
- `internal/config/` — environment-based configuration loader.
- `internal/hmc/` — HMC REST client (login, UOM `ManagedSystem/LogicalPartition`, PCM `ProcessedMetrics`).
- `internal/collector/` — periodic poller and Prometheus metric publication.

Installing from the release tarball

The shipped release `power-hmc-exporter-vX.Y.Z-linux-amd64.tar.gz` contains the statically-linked binary, the systemd unit and env template, the three Grafana dashboards, the documentation (README in `.md`, `.txt`, `.pdf`), this changelog, the EULA and the third-party notices, plus an installer:

```
tar xzf power-hmc-exporter-vX.Y.Z-linux-amd64.tar.gz
cd power-hmc-exporter-vX.Y.Z-linux-amd64/
sudo ./install.sh
sudo vi /etc/sysconfig/power-hmc-exporter
sudo systemctl enable --now power-hmc-exporter
```

`install.sh` creates the `power-hmc-exporter` service account, places the binary at `/usr/local/bin/`, installs the systemd unit and (only if not already present) the `env` file template. The Grafana dashboards under `grafana/` are imported via *Dashboards* → *Import* in your Grafana instance — they are self-contained and only require a Prometheus datasource at import time.

Override install paths with `./install.sh --prefix=... --sysconfdir=... --unitdir=...` if your distribution uses non-default locations.

Building a release

```
make release VERSION=vX.Y.Z
```

This cross-compiles `linux/amd64` (CGO disabled), renders `README.txt` and `README.pdf` from `README.md` via `pandoc`, assembles the install tree and tarballs it into `dist/`. **pandoc and a LaTeX engine (xelatex / TeX Live) are required on the build host** for PDF rendering.

`make release` builds **both editions** and writes two tarballs to `dist/`: the full `power-hmc-exporter-<version>-linux-amd64.tar.gz` and the trial `power-hmc-exporter-trial-<version>-linux-amd64.tar.gz`. Use `make release-full` or `make release-trial` to build a single one.

Editions

`power-hmc-exporter` ships in two editions, built from the same source via a link-time flag (`-X main.edition=...`):

- **Full** — monitors every managed system on your HMC(s). This is the licensed product (per managed system, perpetual).
- **Trial** — identical, but limited to a **single managed system**, at full depth (pools, LPARs, VIOS, SEA, NPIV, VSCSI, all three dashboards). Select the system with `PHMC_TRIAL_SYSTEM` (defaults to the first by name). The build reports its edition via `-version` (a `(trial)` suffix) and `power_hmc_exporter_build_info{edition="full|trial"}`, and the number of withheld systems via `power_hmc_trial_systems_gated`. See `TRIAL.md`, shipped in the trial tarball.

About

power-hmc-exporter is an AIXWatch product by **Strava Ltd**, distributed under a commercial per-managed-system perpetual license. See `LICENSE` for the full End-User License Agreement and `THIRD-PARTY-NOTICES` for the open-source components it incorporates.

- Website: <https://aixwatch.com>
- Support: support@aixwatch.com
- License inquiries: legal@aixwatch.com