Hewlett Packard Enterprise

HPE Performance Cluster Manager Power Management Guide

Abstract

This publication describes the power management features included in the HPE Performance Cluster Manager 1.8 software.

Part Number: 007–6498-013
Published: September 2022
Edition: 1

Notices

Acknowledgments

Product-specific acknowledgments

Revision history

Part number	Publication date	Edition	Summary of changes
007–6498-013	September 2022	1	Supports the HPE Performance Cluster Manager 1.8 release.
007–6498-012	March 2022	1	Supports the HPE Performance Cluster Manager 1.7 release.

Table Continued

Part number	Publication date	Edition	Summary of changes
007–6498-011	September 2021	1	Supports the HPE Performance Cluster Manager 1.6 release.
007–6498-010	March 2021	1	Supports the HPE Performance Cluster Manager 1.5 release.
007–6498-009	September 2020	1	Supports the HPE Performance Cluster Manager 1.4 release.
007–6498-008	April 2020	1	Supports the HPE Performance Cluster Manager 1.3.1 release.
007–6498-007	December 2019	1	Supports the HPE Performance Cluster Manager 1.3 release.
007–6498-006	June 2019	1	Supports the HPE Performance Cluster Manager 1.2 release.
007–6498-005	April 2019	1	Supports the HPE Performance Cluster Manager 1.1 release. Includes SLES 15 and CentOS 7.6 support.
007–6498-004	December 2018	1	Supports the HPE Performance Cluster Manager 1.1 release.
007–6498-003	September 2018	1	Supports the HPE Performance Cluster Manager 1.0 release, patch 3.
007–6498-002	June 2018	1	Supports the HPE Performance Cluster Manager 1.0 release, patch 1.
007–6498-001	June 2018	1	Original publication. Supports the HPE Performance Cluster Manager 1.0 release.

Contents

HPE Performance Cluster Manager power management

The power management documentation is written for system administrators and facility managers. It assumes that you are familiar with Linux, clusters, and system administration. The documentation explains how to monitor and control power use and how to monitor the thermal health of the cluster.

The power management documentation describes the following:

- The power management components
- The power measurement points
- The data collection frequency

Power management features

The HPE Performance Cluster Manager provides the capabilities for node-level and job-level management. The cluster manager runs a service on the admin node to interact with the various power control points. The power management software performs the following functions:

- Measures, aggregates, and reports power statistics
- Sets power limits on HPE clusters

A power distribution unit (PDU) reads AC power and energy measurements on cluster rack-level power domains. The AC power measurement feature requires one or more of the PDUs listed as supported in the HPE Performance Cluster Manager release notes.

The following HPE systems support power management at the node level:

- HPE Cray EX systems.
- HPE Apollo 9000 systems and HPE Apollo 80 systems.
- HPE SGI 8600 systems.
- Systems with HPE iLO 5 node controllers or HPE iLO 4 node controllers that support the Redfish protocol. For example:
  - HPE ProLiant DL380 systems and HPE ProLiant DL360 systems
  - HPE Apollo 6500 systems, HPE Apollo 6000 systems, and HPE Apollo 2000 systems

The cluster manager power management capability requires all HPE ProLiant and HPE Apollo systems to be Gen10 systems.

HPE does not support power management on HPE Apollo 35 systems.

NOTE: For some HPE systems, an iLO license is required to be installed on each compute node in order for the `mpower` command to get limits or set limits. For information about licensing, contact your HPE representative.

Cluster manager documentation

The following list shows the HPE Performance Cluster Manager documentation:

- The release notes contain feature information, platform requirements, and other release-specific guidance. To access the release notes, follow the links on the following website:

  https://www.hpe.com/software/hpcm

  On the product media, the release notes appear in a text file in the following directory:

  `/docs`

  Hewlett Packard Enterprise strongly recommends that you read the release notes, particularly the *Known Issues* section and the *Workarounds* section.

- The following guide presents an overview of the cluster manager and explains how to attach a factory-installed cluster to your site network:

  HPE Performance Cluster Manager Getting Started Guide

- The bare-metal installation documentation is specific to each platform. These guides are as follows:

  - **HPE Performance Cluster Manager Installation Guide for Clusters With ICE Leader Nodes**
  - **HPE Performance Cluster Manager Installation Guide for Clusters With Scalable Unit (SU) Leader Nodes**
  - **HPE Performance Cluster Manager Installation Guide for Clusters Without Leader Nodes**

- The following guide explains the power management features included in the cluster manager:

  HPE Performance Cluster Manager Power Management Guide

- The following guide includes procedures and information about system-wide administration features:

  HPE Performance Cluster Manager Administration Guide

- The following guide includes procedures and information about system monitoring features:

  HPE Performance Cluster Manager System Monitoring Guide

- The following quick-start guide presents an overview of the installation process:

  HPE Performance Cluster Manager Installation Quick Start

- The following command reference shows the cluster manager commands and compares them with the commands used in the SGI Management Suite and in the HPE Insight Cluster Manager Utility:

  HPE Performance Cluster Manager Command Reference

- The following guide explains how to upgrade a cluster from an HPE Performance Cluster Manager 1.X release:

  HPE Performance Cluster Manager Upgrade Guide

After installation, the documentation resides on the system in the following directories:

- Release notes and user guides: `/opt/clmgr/doc`
- Manpages: `/opt/clmgr/man`

Feature descriptions for the HPE SGI 8600 system also apply to SGI ICE XA and SGI ICE systems.

NOTE: The cluster manager documentation includes examples where appropriate. Make sure to substitute information that pertains to your cluster when following the examples.

Using the cm commands

The following topics explain how to use the cluster manager cm commands.

Formatting cm commands and using tab completion

Many cluster manager commands are of the following form:

`cm topic [subtopic ...] action parameters`

The cm commands support tab completion for each *topic*, each *subtopic*, each *action*, and many parameters.

The cm commands implement tab completion for the `-i image` and the `--image image` parameters by comparing command input against the image names stored in the HPE Performance Cluster Manager database.

Likewise, the cm commands implement tab completion for the `-n nodes` and the `--nodes nodes` parameters by comparing command input against the node names stored in the HPE Performance Cluster Manager database.

Using wildcard characters

You can use wildcard characters in the cluster manager cm commands. If you use wildcards in the cm commands, enclose your specification in apostrophes (`' '`). The following table shows the most commonly used wildcard characters.

Wildcard	Effect
*	Matches one or more characters. For example, the following specifies all nodes in rack 1, chassis 1, tray 1 on an HPE Apollo 9000 cluster: `'r1c1t1n*'`
?	Matches exactly one character. For example, the following specifies all nodes in rack 1 that have a single-character chassis: • On an HPE Apollo 9000 cluster: `'r1c?t*n*'` • On an HPE SGI 8600 cluster: `'r1i?n*'`
[ ]	Matches any of the range of characters specified within brackets. For example, the following specifies racks 11, 12, 13, and 14: `'rack1[1-4]'`

The cluster manager includes the `--confirm` parameter, which evaluates and then displays a hostname regular expression before it runs the command. These actions let you decide whether to run the command or to halt the command so you can rewrite the command. For example:

```
# cm node show -n x3000*
x3000c0s33b1n0
x3000c0s33b2n0
x3000c0s33b3n0
x3000c0s33b4n0
# cm node show -n x3000* --confirm

This command will include the following node(s): x3000c0s33b[1-4]n0

continue [y|n]: y
```

```
x3000c0s33b1n0
x3000c0s33b2n0
x3000c0s33b3n0
x3000c0s33b4n0
```

The `--exclude` parameter lets you specify nodes to be omitted from an operation. This parameter prevents the command from running on specified nodes. When specified, the command applies the exclusion after processing all inclusions. For example:

```
# cm node show -n x3000*
x3000c0s33b1n0
x3000c0s33b2n0
x3000c0s33b3n0
x3000c0s33b4n0

# cm node show -n x3000* --exclude *b2*
x3000c0s33b1n0
x3000c0s33b3n0
x3000c0s33b4n0
```

Using of the @ symbol to specify custom groups

If you configure custom groups of nodes, you can operate on these custom node groups in a collective way with a single command. To specify a custom group on a command line, specify `@`*custom_group_name* in place of the *node* argument.

For example, the following command installs package `zlib_devel` on the SLES compute nodes in a custom group named `comp`:

```
# cm node zypper -n @comp install zlib_devel
```

Example *node* specifications

Many `cm` commands accept a `-n` *node* parameter. Generally, for *node*, you can specify one or more node hostnames. The following table shows example *node* specifications.

Specification	Nodes affected
`admin`	The admin node
`n0`	`n0`
`n0,n34`	`n0` and `n34`
`node?`	All nodes that have `node` as the first four characters in the node name
`node[13]`	`node13`
`node[10-14]`	`node10` through `node14`
`node[001-022]`	`node001` through `node022`
`node[2-6,20-26,36]`	`node2` through `node6`, `node20` through `node26`, and `node36`

Table Continued

Specification	Nodes affected
`'node52*'`	`node520` through `node529`
`@gpu-nodes`	All nodes with graphics processing units (GPUs) that are configured into the custom group `gpu-nodes`

Node identification

The cluster manager recognizes distinct node hostnames for each type of cluster that it supports.

NOTE: The information in this topic shows the compute node names that the cluster manager assigns to nodes by default. This naming scheme identifies components by their location in the cluster. These names are assigned automatically when the compute nodes are configured into the cluster.

HPE Cray EX node identification

On HPE Cray EX supercomputers, the node name is in the following format:

x*CABINET*c*CHASSIS*s*SLOT*b*BLADE*n*NODE*

The variables are as follows:

Variable	Specification
CABINET	A 4-digit cabinet identifier in the range 1 <= *CABINET* <= 9999. Specific cabinet identifiers are as follows: • HPE Cray EX fluid-cooled compute: x1000 - x2999 • HPE Cray EX air-cooled I/O: x3000 - x4999 • HPE Cray EX air-cooled compute: x5000 - x5999 • HPE Cray EX TDS: x9000 • HPE Cray EX 2500: x8000 - x8999 Examples: `x1004`, `x3001`.
CHASSIS	A 1-digit chassis identifier in the range 0 <= *CHASSIS* <= 7. Examples: `c1`, `c7`.
SLOT	A 1-digit slot identifier in the range 0 <= *SLOT* <= 7. Examples: `s1`, `s4`.
BLADE	A 1-digit blade identifier in the range 0 <= *BLADE* <= 1. Examples: `b0`, `b1`.
NODE	A 1-digit node identifier in the range 0 <= *NODE* <= 1. Examples: `n0`, `n1`.

The following are node identification examples:

- `x9000c1s2b0n0` is a compute node.
- `fmn01` and `fmn02` are HPE Slingshot interconnect nodes.

HPE Cray EX switch identification

The default switch naming conventions are similar to the default node naming conventions. On HPE Cray EX supercomputers, the switch names are in the following format:

`x`*CABINET*`c`*CHASSIS*`r`*SWITCH*`b`*BMC*

The variables are as follows:

Variable	Specification
CABINET	A 4-digit rack identifier in the range 1 <= *CABINET* <= 9999. Examples: `x0046`, `x0178`.
CHASSIS	A 1-digit chassis identifier in the range 1 <= *CHASSIS* <= 4. Examples: `c1`, `c2`.
SWITCH	A 1-digit tray identifier in the range 0 <= *SWITCH* <= 7. Examples: `r5`, `r7`.
BMC	A 1-digit switch identifier in the range 0 <= *BMC* <= 1. Examples: `b0`, `b1`.

For example: `x1203c0r5b0` is a hostname for an HPE Cray EX switch controller.

HPE Apollo 9000 node identification

On HPE Apollo 9000 clusters, the node name is in one of the following formats:

`r`*RACK*`c`*CHASSIS*`t`*TRAY*`n`*NODE*

The variables are as follows:

Variable	Specification
RACK	A 3-digit rack identifier in the range 1 <= *RACK* <= 999. Examples: `r46`, `r178`.
CHASSIS	A 1-digit chassis identifier in the range 1 <= *CHASSIS* <= 4. Examples: `c1`, `c2`.
TRAY	A 1-digit tray identifier in the range 1 <= *TRAY* <= 8. Examples: `t5`, `t8`.
NODE	A 1-digit node identifier in the range 1 <= *NODE* <= 4. Examples: `n1`, `n4`.

For example: `r100c3t5n1`

HPE Apollo 9000 swich identification

The default switch naming conventions are similar to the default node naming conventions. On HPE Apollo 9000 clusters, the switch names are in the following format:

`r`*RACK*`c`*CHASSIS*`t`*TRAY*`s`*SWITCH*

The variables are as follows:

Variable	Specification
RACK	A 3-digit rack identifier in the range 1 <= *RACK* <= 999. Examples: `r46`, `r178`.
CHASSIS	A 1-digit chassis identifier in the range 1 <= *CHASSIS* <= 4. Examples: `c1`, `i2`.
TRAY	A 1-digit tray identifier in the range 1 <= *TRAY* <= 8. Examples: `t5`, `t8`.
SWITCH	A 1-digit switch identifier in the range 1 <= *SWITCH* <= 4. Examples: `s2`, `s3`.

HPE SGI 8600 node identification

On HPE SGI 8600 clusters, the node name is in the following format:

`r`*RACK*`i`*CHASSIS*`n`*NODE*

The variables are as follows:

Variable	Specification
RACK	A 3-digit rack identifier in the range 1 <= *RACK* <= 999. Examples: `r46`, `r178`.
CHASSIS	A 1-digit chassis identifier in the range 0 <= *CHASSIS* <= 3. Examples: `i1`, `i2`.
NODE	A 2-digit node identifier in the range 0 <= *NODE* <= 35. Examples: `n2`, `n33`.

For example: `r10i0n31`

HPE SGI 8600 switch identification

The default switch naming conventions are similar to the default node naming conventions. On HPE SGI 8600 clusters, the switch names are in the following format:

`r`*RACK*`i`*CHASSIS*`s`*SWITCH*

The variables are as follows:

Variable	Specification
RACK	A 3-digit rack identifier in the range 1 <= *RACK* <= 999. Examples: `r46`, `r178`.
CHASSIS	A 1-digit chassis controller identifier in the range 0 <= *CHASSIS* <= 3. Examples: `i1`, `i2`.
SWITCH	A 2-digit switch identifier in the range 0 <= *SWITCH* <= 1. Examples: `s0`, `s1`.

Cluster manager videos

The following videos show HPE Performance Cluster Manager functionality:

- **Cluster manager overview**
- **Cluster manager integration with NVIDIA DCGM**
- **Workload management using the cluster manager and Altair PBS Professional**
- **Service Infrastructure Monitoring with Grafana**
- **AIOps in production**

Identifying the cluster manager release that is installed

Procedure

1. Log into the admin node as the root user.
2. Enter one of the following commands:
   - `# cat /etc/*release`

     This command displays information about operation system distribution files and the cluster manager release.
   - `# cat /etc/sgi*release`
   - `# cadmin --version`
   - `# cadmin --ver`

Power management on HPE cluster systems

Node-level power limiting

The core of node-level power limiting is frequency control of the processors and throttling the bandwidth to memory. Every processor has various power, sleep, and turbo states. By controlling the processor frequency, either with a not-to-exceed level or a fixed range, the power is effectively limited.

CPU frequency control is accomplished out-of-band by the power service. The power service relies on a management controller on the compute node to perform the actual power limiting. It is possible to control CPU frequency in-band, while the operating system is running. This approach is not used by the power service and is beyond the scope of this manual.

Graphics processing unit (GPU) power limiting is performed entirely in-band through calls to the GPU management library that runs on the node operating system. The cluster manager supports GPU power limiting for Nvidia GPUs only.

Job power management

The power service provides an API facility to accommodate a third-party workload manager. Workload managers include products such as PBS Pro or Slurm. Additionally, the Sandia PowerAPI is a third-party power management package. The Sandia PowerAPI includes a plugin that connects to the power service API.

The workload managers collect power and energy statistics for a group of nodes designated as running a common job. They collect accurate measures of power and energy on a multinode job basis. You can use the API to apply a power limit to all nodes of a designated job.

Detecting graphics processing units (GPUs)

The cluster manager detects the presence of NVIDIA GPUs on the compute nodes when one of the following actions occurs:

- The power management service starts.
- The cluster manager performs an internal restart.

  This restart can occur, for example, when you add new nodes to the cluster. In this case, you make the configuration change, and the cluster manager updates the internal cluster management database. To detect the presence of NVIDIA GPUs, the cluster manager issues a remote `lspci` command to each compute node. This operation is performed within the power management service with the inventory monitoring task.

Configuring graphics processing units (GPUs) for power monitoring

The cluster manager installation process does not configure enable GPU power monitoring or GPU power limiting.

For information about these procedures, see the following:

HPE Performance Cluster Manager Administration Guide

Graphics processing unit (GPU) power reporting and GPU power limiting

GPU power reporting includes the GPU power, in watts, consumed by all GPUs on a compute node. For example, assume that a compute node includes four GPUs. Each GPU might consume 50 watts of power. The cluster manager reports this as 200 watts.

GPU power limiting applies equally to all the GPUs included on a node. For example, assume that a node includes two 2 GPUs. If you set a GPU power limit of 200 watts, the cluster manager sets the power limit for each GPU to 200 watts.

The GPU power limit allowable range varies by the particular GPU model. When you apply a power limit to a GPU, if the limit is outside of the allowable range, the system returns an error.

Viewing power data in the cluster manager GUI

Cluster manager commands such as `mpower` return power management information at the command line.

You can also view power management information within the cluster manager GUI. The `cm monitoring` command enables monitoring within the cluster manager. Some power monitoring capabilities require you to configure settings in the `cmuserver.conf` file or the `ActonsAndAlertsFile` file.

For information about viewing power data within the GUI, see the following:

HPE Performance Cluster Manager Administration Guide

Configuring the power management software for rack-level, continuous AC power monitoring

For systems with power distribution units (PDUs), the cluster manager continuously collects, aggregates, and stores AC power and energy data from all racks in round-robin databases (RRDs). In addition, the cluster manager makes the data accessible to you through graphical displays or other analysis.

The following procedure explains how to configure the power management software.

Procedure

1. (Conditional) Determine whether the PDUs are configured in the cluster database.

   Complete this step only on clusters without leader nodes. Do not complete this step on clusters with leader nodes. On clusters with leader nodes, the installer configures the PDUs automatically, and the software manages the power metering components automatically.

   Enter the following command:

   ```
   # cm node show -t system pdu
   ```

   If the command output lists your PDUs, proceed to the next step in this procedure.

   If the command output does not list your PDUs, add the PDUs to the cluster database, as follows:

   a. Create a new, scratch, cluster definition file for the PDUs. Add PDU entries in the file.

      The following is an example of an entry in example cluster definition file `my-pdus` for PDU 0:

      ```
      internal_name=pdu0,mgmt_bmc_net_name=head-bmc,mgmt_bmc_net_macs=00:0a:9c:54:72:98,
      pdu_protocol=snmp,geolocation="aisle 4 rack 2",hostname1=pdu0
      ```

   b. Use the `cm node add` command and your file as input to configure the PDUs.

      For example, if you place all your PDU entries in scratch file `my-pdus`, enter the following command to configure the PDUs:

      ```
      # cm node add -c my-pdus
      ```

   c. Enter the following command to verify that the PDUs are configured:

      ```
      # cm node show -t system pdu
      ```

   For more information about configuring PDUs, see the installation guide for your platform. For links to the installation guides, see the following:

   Cluster manager documentation

2. (Conditional) Use the `cm node show` command and the `cm node` command to set PDU attributes.

   Complete this step only on clusters without leader nodes. Do not complete this step on clusters with leader nodes. On clusters with leader nodes, the cluster manager sets the PDU attributes automatically.

   The following examples show the attributes that you can set and how you can retrieve information about the attributes:

- The following commands rename a PDU and then display the new name:

  ```
  cm node set -n pdu_hostname -name new_pdu_hostname
  cm node show -n pdu_hostname
  ```

- The following commands set the physical location of a PDU and then display the location:

  ```
  cm node set --geolocation 'location' -n pdu_hostname
  cm node show --geolocation -n pdu_hostname
  ```

- The following commands set the protocol for a PDU and then display the PDU protocol:

  ```
  cm node set -n pdu_hostname --pdu-protocol protocol
  cm node show -B -n pdu_hostname
  ```

  The variables in the preceding commands are as follows:

Variable	Value
pdu_hostname	Name of the PDU at this time.
new_pdu_hostname	Name that you want to assign to the PDU.
"*location*"	Location of the PDU in a rack. For example, "`aisle 3, rack 5`"
protocol	Specify `modbus` or `snmp`.

  Example:

```
# cm node show -B -t system pdu
NODE                CARDIPADDRESS      CARDMACADDRESS       CARDTYPE          PROTOCOL
pdu1                172.24.0.4         00:9c:02:99:1b:3e    none              none
```

3. On the admin node, open the following file:

   `/opt/clmgr/etc/clmgr-power.conf`

4. To configure the `clmgr-power` service on the admin node for AC input power monitoring, change the `acpower_monitoring_active` setting to `True`.

   For example:

   ```
   acpower_monitoring_active = True
   ```

5. To configure the cluster manager to export power monitoring data to Elasticsearch, change the `acpower_to_mqtt` setting to `True`.

   For example:

   ```
   acpower_to_mqtt = True
   ```

6. Save and close the `clmgr-power.conf` file.

7. Enter the following command to restart the `clmgr-power` service on the admin node:

   ```
   # systemctl restart clmgr-power
   ```

8. Use the `acpower` command to view AC power and energy use at the rack level or at the cluster level.

   The `acpower` command provides power information at the rack level or at the cluster level.

On clusters without leader nodes, energy reading is available only when AC rack meters are configured.

Enter the following command to determine whether PDU power monitoring is enabled, and take appropriate actions.

```
# acpower system get_power_snmp
```

The command returns one of the following:

- If the `acpower` command is unsuccessful, it returns the following:

  ```
  failure :: {u'info': 'AcPowerService not running'}
  ```

  In this case, enter the following command to verify whether the `pdu-collect` service is running:

  ```
  # systemctl status pdu-collect
  ```

  If the `pdu-collect` service is not running, enter the following command to start the service:

  ```
  # systemctl restart pdu-collect
  ```

- If the `acpower` command is successful, and the `pdu-collect` service is running, the command returns the following:

  ```
  ● pdu-collect.service - pdu power
     Loaded: loaded (/usr/lib/systemd/system/pdu-collect.service; enabled; vendor preset>
     Active: active (exited) since Fri 20XX-07-17 10:02:23 CDT; 24min ago
  ```

  If the `pdu-collect` service is not running, enter the following command to start the service:

  ```
  # systemctl restart pdu-collect
  ```

For more information, see the following manpages:

- `acpower`
- `pdu-collect`

Setting power limits and retrieving power information with the `mpower` command

You can use the `mpower` command to accomplish the following:

- Set power limits on nodes
- Retrieve power readings from nodes
- Retrieve information about node power limits

The `mpower` command performs the following operations:

- At the node level, the command operates on the specific node or set of nodes specified on the command line.
- At the graphics processing unit (GPU) level, you can query compute nodes with GPUs to obtain the GPU power level. You can also set or get GPU power limits.

NOTE: To retrieve online help information for the `mpower` command, enter the following on the command line:

```
# mpower -h
```

`mpower` command format

The `mpower` command has the following formats:

- `mpower node` *`action target`*
- `mpower GPU` *`action target`*

The variables are as follows:

Variable	Specification
action	Specify one of the following for *action*: • `get_limit` Displays the power limits set for the designated target. • `set_limit` *watts* Sets a power limit for the designated *target*. To set a power limit, use the `set_limit` action as follows: For *watts*, specify the power limit, in watts, to be enforced on the *target*. Specify an integer number of watts in the range 100-1500, inclusive. ICE compute nodes typically cannot limit power under 100 watts. The maximum power for an ICE compute node varies by processor and memory configuration but typically cannot reach or exceed 450 watts. Compute nodes typically cannot limit power under 150 watts. The maximum power for a compute node also varies by configuration (processor/memory/onboard disks) and can exceed 600 watts. For nodes with graphics processing units (GPUs), the power consumption level can be considerably higher. However, it is not possible to use out-of-band power management to limit GPU power. This limiting must be done in-band and is beyond the scope of this documentation. The power limit set on any compute node is kept in the Node Manager NVRAM on the compute node. Hence, the limit is saved across reboots and power cycles. • `get_power` Displays the current power readings for the designated targets. The power readings include data from the last time the power was reset.
target	For *target*, specify one or more node hostnames. If you specify multiple hostnames, use a comma (`,`) to separate the hostnames. Example: `r1i0n0,r2i0n0` To display a list of node names, use the `cm node show` command. For information about the wildcard characters you can use in the *target* specification, see the following: **Wildcard characters for `mpower` command *target* specifications**

Wildcard characters for `mpower` command *target* specifications

You can use pattern-matching expressions (wildcards) in *target* specifications. Enclose the *target* in quotation mark characters (") if it contains wildcard characters. The `mpower` command supports globbing expressions. The following table shows the most commonly used expressions:

Wildcard	Effect
*	Matches one or more characters. For example: "n*"
?	Matches exactly one character. For example, the following specifies all nodes in rack 1 of an HPE SGI 8600 system that have a single-character chassis: "r1i?n*"
[ ]	Matches any of the range of characters specified within brackets. For example: "n[1-4]"

mpower command examples -- setting limits and retrieving power consumption information

Using mpower to obtain power usage for one ICE compute node on an HPE SGI 8600 system

To display the power usage for ICE compute node r1i1n2, enter the following command:

```
# mpower node get_power r1i1n2
r1i1n2 126W
```

This command displays the power usage for the node. The power reading returned is a moving average. This command fails if no power policy exists.

Using mpower to obtain the power limit for one compute node

To display the power limit for compute node r1c2t3n4, enter the following command:

```
# mpower node get_limit r1c2t3n4
r1c2t3n4 1000W
```

Using mpower to set a power limit on one ICE compute node on an HPE SGI 8600 system

To set a power limit of 200 watts for ICE compute node r1i1n2, enter the following command:

```
# mpower node set_limit r1i1n2 200
```

Using mpower to set a power limit on one compute node

To set a power limit of 300 watts for compute node n1, enter the following command:

```
# mpower node set_limit n1 300
```

Using `mpower` to obtain the power limit on one compute node

To display the power limit for compute node `n1`, enter the following command:

```
# mpower node get_limit n1
n1 1000W
```

Using `mpower` to obtain the power limit on one graphics processing unit (GPU)

To display the power limit for the GPU attached to compute node `r1c2t3n4`, enter the following command:

```
# mpower gpu get_power r1c2t3n4
r1c2t3n4 500W
```

Configuring node-level power monitoring on HPE SGI 8600 clusters

When you configure node-level power monitoring, the cluster manager is enabled with the following new capabilities:

- The cluster manager can send power data to Elasticsearch.
- You can use the job-level API routines.
- You can use the Sandia National Laboratory Power API plugin.

  The Sandia National Laboratory Power API is a suite of APIs for high performance power measurement and control. HPE includes the plugin with the cluster manager installation files.

Complete the following procedures:

Procedure

1. **Installing the Power API plugin on HPE SGI 8600 clusters**
2. **Enabling power monitoring on HPE SGI 8600 clusters**

Installing the Power API plugin on HPE SGI 8600 clusters

Procedure

1. Download and install the Power API onto one of the compute nodes that you have reserved for user services.

   This procedure refers to a compute node with a services role as a **service node**.

   For example, install the Power API into `/opt/pwrapi` on the compute node named `n1`.
2. Select one of the service nodes to host the RPM, and use the `cm node` command to install the RPM on that node.

   For example, the following commands install the RPM on node `n1`:

   - RHEL 9.X, RHEL 8.X:

     ```
     # cm node dnf -n n1 install power-8600-v2.0-1.0
     ```

   - RHEL 7.X:

     ```
     # cm node yum -n n1 install power-8600-v2.0-1.0
     ```

   - SLES:

     ```
     # cm node zypper -n n1 install power-8600-v2.0-1.0
     ```

3. Log into the service node through an `ssh` connection.

   For example, log into `n1`.
4. Change to the directory where the RPM is installed.

   For example:

```
# cd /opt/hpe/power/8600/v2.0
```

5. Run the `install.sh` script, and specify the full path to the Power API.

   For example, if the Power API is installed in `/opt/pwrapi`, enter the following command:

   ```
   # ./install.sh /opt/pwrapi
   ```

6. Use the `cd` command to change to the *`install_path`*`/lib` directory.
7. Verify that the software is installed properly.

   Use the `ls` command, and verify that the following files reside in the *`install_path`*`/lib` directory:

   - `libpwr_hpe_power_manager_dev.la`
   - `libpwr_hpe_power_manager_dev.so`
   - `libpwr_hpe_power_manager_dev.so.0`
   - `libpwr_hpe_power_manager_dev.so.0.1.0`

Enabling power monitoring on HPE SGI 8600 clusters

Prerequisites
Installing the Power API plugin on HPE SGI 8600 clusters

Procedure

1. Use the `ssh` command to connect to the admin node.
2. Open the following file in a text editor:

   `/opt/clmgr/etc/clmgr-power.conf`
3. Edit the file as follows:

   - Enable power monitoring by editing the monitoring setting in the file to read as follows:

     `node_power_monitoring_active = True`
   - If your system has AC PDUs, set the following:

     `acpower_monitoring_active = True`

     To determine if your system has AC PDUs, enter the following command and analyze the output:

     ```
     # cm node show -t system pdu
     ```
4. Save and close the file.
5. Enter the following command to restart the cluster manager power service:

   ```
   # systemctl restart clmgr-power
   ```
6. Log back into the service node, and access the `README` file for more information about the Power API plugin.

   The `README` file is in the following location:

   `/opt/hpe/power/8600/v2.0`

The `README` file contains information about the following:

- The helper script that generates XML, `hpe_power_manager_xml.py`.
- The sample file, `hpe_power_manager.xml`.

Managing power consumption at the job level

Python client modules

The power management API consists of the following Python modules:

- The primary module is `hpe_clmgr_power_api.py`. This module contains calls to the cluster manager power infrastructure. These calls perform measure power. limit power, and group nodes.

  The `hpe_clmgr_power_api.py` module is intended to be imported by client Python code. This module supports one connection only to the cluster manager infrastructure. This module is not subprocess executable. This module requires Python 2.5 or greater. Function calls of this API to the cluster manager infrastructure block the connection and subsequent return values. In error conditions, function calls throw exceptions.
- The secondary Python module is `remlib.py`. The `hpe_clmgr_power_api.py` module imports the `remlib.py` module. The secondary module is a wrapper for the XML RPC remote library.

The Python modules reside in the compute node images in the following directory:

`/opt/clmgr/power-service`

For information about the API function calls, see the following:

API functions

For more in-depth information, see the module itself, which is annotated for each function call.

Defining and using custom groups within a job

Job-level power management uses the concept of a **custom group**. A custom group is an arbitrary list of nodes with a unique name to reference that list of nodes.

You can create and manage additional custom groups in one of the following ways:

- By using the power management API.
- By using the cluster manager GUI.
- The `cm group custom add` command creates a custom group. Other cluster manager commands enable you to manage custom groups.
- By using the Slurm connector.

The preceding methods for creating and managing custom groups are equivalent. The power management features are equally usable regardless of how you created the custom group.

For example, a workload manager might choose a group of nodes to run a user job with a job name defined. You can create a custom group that contains this same list of nodes. As a convenience, you can use the same name for both the job name and the custom group name.

This guide describes how to use the power management API to create and manage custom groups. For information on the other methods in the preceding list, see the following:

HPE Performance Cluster Manager Administration Guide

Power profiles

The API has a list of power profiles available for setting power limits. The list consists of power-level options ranging from 100 watts to 1500 watts and `NONE` (meaning no power limit). These profile options become the power limit value for each node in a job while the job is underway.

For diskless ICE compute nodes, the 100-watt value is the lowest realistic limiting level. For other compute servers with disks, the lowest realistic power limit is approximately 150 watts. Upper limits vary by server type, processor, memory type, and configuration. If you set a power limit above the absolute peak of a compute node, effectively, there is no power limiting.

The API provides the `ListAvailableProfiles()` function call to list all power profiles available for the cluster. You can use the `MonitorStart()` function to apply a power profile to a job at its start time. You can use the `MonitorAdjust()` function to establish a new power profile while the job is underway.

Running a job with power monitoring and power limiting

The following procedure shows how to run a job with power limiting and power monitoring. The procedure is written in general terms. You can customize the procedure for your job.

Procedure

1. Use the following function call to create a custom group:

   `NodesetCreate('custom_group_name',['node_hostname','node_hostname',...])`

   Enclose the list of hostnames in brackets. To specify multiple hostnames, use commas to separate them. Enclose each node hostname in apostrophes.

   For example:

   `NodesetCreate('mynodeset',['r1i0n0','r1i0n1','r1i0n2'])`

2. Call `MonitorStart('custom_group_name','profile')` to initialize the monitoring and, optionally, enable power limiting.

   To run the job without power limiting, set *profile* to `NONE`.

   To set a power limit on each node in the custom group, set *profile* to a value other than `NONE`. For example, to set the power limit for each node to 100 watts, set *profile* to `'100W'`.

   The job scheduler plugins use `MonitorReport()` to get the power and energy data periodically until `MonitorStop()` is called.

3. Start the job.

   The cluster manager gathers power statistics readings from every compute node of the custom group and maintains an energy accumulator.

4. At the end of the job, call `MonitorStop('custom_group_name')`.

   The cluster manager completes the monitoring and reports the energy used by that job.

More power management workflows

The API provides the following additional capabilities for obtaining energy use statistics and power limiting:

- While a job is underway, you can obtain energy statistics for the job or for nodes from the job custom group.
- While a job is underway, you can change the power profile of the custom group
- You can reset the energy accumulators for a custom group that is running a monitoring job at any time.

For example, to obtain immediate energy values, a workflow can complete the following tasks:

- Create a custom group for a job
- Issue the following API function:

  `MonitorStart('custom_group_name','profile')`

- At some later time during job execution, enter the following API function:

  `MonitorReport('custom_group_name')`

A workload manager decision can be made to make the following changes:

- Change the power profile
- Speed up the job (less power limiting)
- Reduce the power consumption (more power limiting)

This sampling and profile adjustment can take place once or repetitively as required. At job completion, the final energy usage is reported.

CAUTION: Do not set up a script that uses the `mpower` command in a tight loop. The power service polls all compute nodes in the cluster and performs various housekeeping tasks. Such a script can cause severe disruptions of the following:

- The power service
- The power management hardware

More unintended problems can also occur.

API connectivity

The use of the power management API is flexible within the system management infrastructure. The controlling entity of a workload manager can reside on a designated server that is separate from the underlying compute nodes. The controlling entity can provide all aspects of workload management from that designated server. The workload manager on the dedicated server can import the `hpe_clmgr_power_api.py` module and connect to the power management infrastructure directly.

Some workload managers offer an agent on each compute node. Any of these agents can act as a job leader. With appropriate credentials, the `hpe_clmgr_power_api.py` module can be used in precisely the same fashion. Stated differently, the API is able to reach the power management infrastructure flexibly from various points in the cluster.

API functions

The power management software includes many API functions. The API documentation uses the following format to represent the API syntax:

`name ('arg1'[,'arg2',...],command)`

In the preceding format, brackets (`[ ]`) indicate optional items that can appear on the call. For example, `NodePowerControl` has the following syntax statement:

`NodePowerControl('node'[,'node',...],command`

The format indicates that one *node* hostname is required. However, you can supply more *node* hostnames. In addition, notice that all *node* hostnames must be enclosed in apostrophes (`' '`).

For example, the following is a correct `NodePowerControl` call:

```
NodePowerControl('n01','n02','n03',off)
```

For more information about the function call parameters and return values, see the annotations in the `hpe_clmgr_power_api.py` module.

NOTE: Complete the following procedure before you use the API functions to monitor power at the node level:

Configuring node-level power monitoring on HPE SGI 8600 clusters

---

ChassisEnergyReport('*chassis*')

For a single chassis, returns `[ energy, time ]`, which are as follows:

- *energy* : float

  Total energy use for the specified chassis in watt hours.

- *time* : int

  Time in seconds since activate or reset was last called.

For *chassis*, specify the hostname of a cluster chassis.

For information about how to obtain a list of chassis hostnames, see the following:

InventoryRackChassis('*rack*')

ChassisPowerReport('*chassis*')

For a single chassis, returns `[ power, time ]`, which are as follows:

- *power* : float

  Power in watts.

- *time* : int

  Time in seconds since activate or reset was last called.

For *chassis*, specify the hostname of a cluster chassis.

For information about how to obtain a list of chassis hostnames, see the following:

InventoryRackChassis('*rack*')

GpuPowerGetLimit('*node*')

For a single node, returns the following metrics for all the graphics processing units (GPUs) of that node:

`[ power_limit, minpwr, maxpwr, default]`

For *node*, specify the hostname of a node.

On output, the *minpwr* and *maxpwr* values are notable because the `GpuPowerLimit` function sets limit values that fall within the range of *minpwr* and *maxpwr* found here.

The return values are as follows:

- *power_limit*: float

  The power limit, in watts, set equally to all GPUs on the specified node.
- *minpwr*: int

  The minimum power, in watts, for all GPUs on the specified node.
- *maxpwr*: int

  The maximum power, in watts, for all GPUs on the specified node.
- *default*

  The default power limit setting for each GPU on the specified node.

For information about how to retrieve a list of node hostnames, see the following:

InventoryNodes()

GpuPowerLimit('*node*','*limit*')

Sets the power limit for each graphics processing unit (GPU) of a single node. The cluster manager applies the same limit value equally to each GPU.

The variables are as follows:

Variable	Specification
node	The hostname of a node. For information about how to retrieve a list of node hostnames, see the following: **InventoryNodes()**
limit	Specify the power limit, in watts, to be enforced for all GPUs on the node. Specify an integer number in the range 100-500, inclusive.

GpuPowerReport('*node*')

For a single node, returns the following:

`[ energy, time, avepwr, instpwr, minpwr, maxpwr ]`

For *node*, specify the hostname of a node. If the node is not found, the function returns an empty list.

The return values are as follows:

- *energy* : float

Total energy use for all graphics processing units (GPUs) of the specified node in watt hours. The energy calculation is based on the start time of the power service monitoring and is not associated with any job time.

- *time* : int

  Time in seconds since activate or reset was last called. This time is Universal Time Coordinated (UTC).

- *avepwr* : int

  The average power, in watts, for the GPUs on the specified node.

- *instpwr* : int

  The instant power, in watts, for the GPUs on the specified node.

- *minpwr* : int

  The minimum power, in watts, for the GPUs on the specified node.

- *maxpwr* : int

  The maximum power, in watts, for the GPUs on the specified node.

For information about how to retrieve a list of node hostnames, see the following:

`InventoryNodes()`

`InventoryChassis('chassis')`

Returns the list of nodes in a chassis on a cluster managed by ICE leader nodes or SU leader nodes.

For *chassis*, specify the hostname of a chassis.

For information about how to obtain a list of chassis hostnames, see the following:

`InventoryRackChassis('rack')`

`InventoryMgmtNodes()`

Returns a list of admin nodes and leader nodes in the cluster.

`InventoryNodes()`

Returns the list of ICE compute nodes and compute nodes in the system in no particular order.

`InventoryPdus()`

Returns a list of the power distribution unit (PDU) meter names.

Not valid on clusters with leader nodes. On a cluster with leader nodes, if you call this function, the call returns an empty list.

`InventoryRackChassis('rack')`

Returns the list of chassis controllers in a rack. The returned list includes the hostname for each chassis controller.

For *rack*, specify the name of a rack in the cluster. For example, specify `rack1`.

For information about how to obtain a list of rack names, see the following:

`InventoryRackList()`

This function applies only to clusters with ICE leader nodes.

InventoryRackList()

Returns the list of racks.

InventoryRackPdus('*rack*')

Returns the list of power distribution unit (PDU) meters in a rack.

For *rack*, specify the name of the rack. For example, `rack1`.

For information about how to obtain a list of rack names, see the following:

InventoryRackList()

ListAvailableProfiles()

Lists the available power profiles.

Returns [*profile_list*]

MonitorAdjust('*custom_group*','*profile*')

Changes the power limit for the custom group.

The variables are as follows:

Variable	Specification
custom_group	A predefined custom group name. For information about how to obtain a list of custom groups, see the following: **NodesetList()**
profile	A predefined profile name or `NONE`. For example, `100W`. For information about how to obtain a list of profiles, see the following: **ListAvailableProfiles()**

MonitorReport('*custom_group*')

Generates a power/energy report for a custom group.

For *custom_group*, specify a predefined custom group name.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

Returns [*time*, *energy_stats*]:

- *time* : int

  Time in seconds since monitoring was activated or last reset.

- *energy* : float

  Total energy use for all nodes in custom group in kilowatt hours.

- *avepwr* : int

  A sum of the average power for all nodes in the custom group.

- *instpwr* : int

  A sum of the instant power for all nodes in the custom group.

- *minpwr* : int

  A sum of the minimum power for all nodes in the custom group.

- *maxpwr* : int

  A sum of the maximum power for all nodes in the custom group.

MonitorReportError('*custom_group*')

Reports monitoring errors.

For *custom_group*, specify the name of a predefined custom group.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

Returns [(*hostname*, *error_string*), ...]

MonitorReportForNode('*custom_group*','*node*')

Generates a power and energy report for a specified node. Specify both a custom group name and a node name.

The variables are as follows:

Variable	Specification
custom_group	A predefined custom group name. For information about how to obtain a list of custom groups, see the following: **NodesetList()**
node	The hostname of the node for which you want the report. For information about how to retrieve a list of node hostnames, see the following: **InventoryNodes()**

Returns [*time*, *energy_stats*]

- *time* : int

  Time in seconds since activate or reset was last called

- *energy* : float

  Total energy use for the specified node in kilowatt hours

- *avepwr* : int

  The average power for the specified node in the custom group

- *instpwr* : int

The instant power for the specified node in the custom group

- *minpwr* : int

  The minimum power for the specified node in the custom group

- *maxpwr* : int

  The maximum power for the specified node in the custom group

MonitorReset('*custom_group*')

Resets the power usage accumulators for a custom group.

For *custom_group*, specify a predefined custom group name.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

Returns all pending reports. Same as MonitorReport().

MonitorStart('*custom_group*','*profile*')

Activates the power profile and starts monitor for total power usage.

The variables are as follows:

Variable	Specification
custom_group	Specify a predefined custom group name. For information about how to obtain a list of custom groups, see the following: **NodesetList()**
profile	Specify a predefined profile name or NONE. No argument or NONE sets the profile to full power, 1000W. For information about how to obtain a list of profiles, see the following: **ListAvailableProfiles()**

MonitorStop('*custom_group*')

Stops the monitoring job.

For *custom_group*, specify a predefined custom group name.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

MonitorStopAll()

Stops all monitoring jobs.

NodeBiosSettings('*node*'[, '*node*', ...])

Returns the BIOS settings for one or more nodes that support HPE iLO.

For *node*, specify the hostname of a node. Use a comma to separate multiple hostnames. For information about how to obtain a list of nodes, see the following:

InventoryNodes()

NodePowerControl('*node*'[,'*node*',...],*command*)

Turns the power on or off for one or more nodes. Upon success, the command returns nothing.

The variables are as follows:

Variable	Specification
node	The hostname of a node. Use a comma to separate multiple hostnames. For information about how to obtain a list of nodes, see the following: **InventoryNodes()**
command	`on` or `off`. This parameter is not case-sensitive.

The command returns a `TAUserError` under the following circumstances:

- You accidentally enter something other than `on` or `off` for command.
- The command cannot find the node.
- The power operation fails.

Examples:

```
NodePowerControl('n01','n02',off)
NodePowerControl('n10',ON)
```

NodePowerGetLimit('*node*')

Returns the power limit for a node.

For *node*, specify the hostname of a node.

For information about how to retrieve a list of node hostnames, see the following:

InventoryNodes()

NodePowerLimit('*node*','*limit*')

Sets the power limit for a single node.

The variables are as follows:

Variable	Specification
node	The hostname of a node. For information about how to retrieve a list of node hostnames, see the following: **InventoryNodes()**
limit	Specify the power limit, in watts, to be enforced on the node. Specify an integer number in the range 100-500, inclusive.

This function overrides limits set using the following functions:

- `MonitorAdjust`
- `MonitorStart`

Likewise, if you use `NodePowerLimit` to set a power limit, you can use `MonitorAdjust` or `MonitorStart` to override that limit later

NodePowerReport('*node*')

For a single node, returns the following:

`[ hostname, energy, time, avepwr, instpwr, minpwr, maxpwr ]`

The return values are as follows:

- *hostname* : string

  The hostname of the node in the report.
- *energy* : float

  Total energy use for the specified node in kilowatt hours.
- *time* : int

  Time in seconds since activate or reset was last called.
- *avepwr* : int

  The average power for the specified node.
- *instpwr* : int

  The instant power for the specified node.
- *minpwr* : int

  The minimum power for the specified node.
- *maxpwr* : int

  The maximum power for the specified node.

For *node*, specify the hostname of a node.

If the node is not found, the function returns an empty list.

For information about how to retrieve a list of node hostnames, see the following:

InventoryNodes()

NodePowerStatus('*node*'[,'*node*',...])

Returns a list of nodes and node status readings in the following format:

```
{'node':'status'}
{'node':'status'}
.
.
.
```

Possible *status* readings are as follows:

- ON
- OFF
- BOOTED
- UNAVAILABLE

For *node*, specify the hostname of a node. Use a comma to separate multiple node hostnames.

For information about how to retrieve a list of node hostnames, see the following:

InventoryNodes()

This function generates TAUserError if the node is not found or if a status cannot be retrieved.

NodeSensorList('*node*')

Returns the sensor list for a node.

For *node*, specify the hostname of a node.

For information about how to retrieve a list of node hostnames, see the following:

InventoryNodes()

NodeThermalReport('*node*')

Returns the following values for a node:

```
[ hostname, thermal_sensor, thermal_sensor, ... ]
```

These values are as follows:

- *hostname* : string

  The hostname of the node.

- *thermal_sensor* : integer

  The temperature of the node inlet temperature sensor in degrees Celsius.

For *node*, specify the hostname of a node.

For information about how to retrieve a list of node hostnames, see the following:

InventoryNodes()

NodeThermalReportSensor('*node*','*sensor*')

Returns sensor data for the specified sensor.

The variables are as follows:

Variable	Specification
node	The hostname of a node. For information about how to retrieve a list of node hostnames, see the following: **InventoryNodes()**
sensor	The string name of a sensor. For information about how to retrieve a list of sensor names, see the following: **NodeSensorList('*node*')**

NodesetCreate('*custom_group*',['*node*','*node*',...])

Creates a custom group. Overwrites an existing custom group with the same name.

The variables are as follows:

Variable	Specification
custom_group	A name for the custom group.
node	One or more nodes. The resulting custom group includes the nodes specified. Separate the hostnames with commas. Enclose each hostname in apostrophes. For information about how to retrieve a list of node hostnames, see the following: **InventoryNodes()**

For example:

```
NodesetCreate('mycustomgroup',['r1i0n0','r1i0n1','r1i0n2'])
```

NodesetDelete('*custom_group*')

Deletes a custom group from the database.

Returns `True`.

For *custom_group*, specify a predefined custom group name.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

NodesetDetail('*custom_group*',['*node*','*node*',...])

Lists the nodes in the specified custom group.

Returns the list of node hostnames for the specified custom group.

The variables are as follows:

Variable	Specification
custom_group	The name of a predefined custom group. For information about how to obtain a list of custom groups, see the following: **NodesetList()**
node	One or more nodes. Separate the hostnames with commas. Enclose each hostname in apostrophes. For information about how to retrieve a list of node hostnames, see the following: **InventoryNodes()**

NodesetGpuReport('*custom_group*')

For a single custom group, returns [*instpwr*, *energy*], which are as follows:

- *instpwr* : int

  The instant power for the specified node in the custom group.

- *energy* : float

  Total energy use for the specified custom group in kilowatt hours. The energy calculation is based on the start time of the power management service, not the start time of any job.

For *custom_group*, specify the name of a predefined custom group.

If the custom group is not found, the function returns an empty list.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

NodesetList()

Lists the existing custom group names. You can specify a custom group name in the *custom_group* input field of the power APIs.

NodesetPowerControl('*custom_group*',*command*)

Turns the power on or off for one custom group. Upon success, the command returns nothing.

The variables are as follows:

Variable	Specification
custom_group	The name of a custom group. For information about how to obtain a list of custom groups, see the following: **NodesetList()**
command	`on` or `off`. This parameter is not case-sensitive.

The command returns a `TAUserError` under the following circumstances:

- You accidentally enter something other than `on` or `off` for command.
- The command cannot find the custom group.
- The power operation fails.

NodesetPowerLimit('*custom_group*','*limit*')

Sets the power limit for a custom group.

The variables are as follows:

Variable	Specification
custom_group	The name of a custom group. For information about how to obtain a list of custom groups, see the following: **NodesetList()**
limit	The power limit, in watts, to be enforced on the custom group. Specify an integer number in the range 100-500, inclusive.

This function overrides limits set using the following functions:

- `MonitorAdjust`
- `MonitorStart`

Likewise, if you use `NodePowerLimit` to set a power limit, you can use `MonitorAdjust` or `MonitorStart` to override that limit later

NodesetPowerReport('*custom_group*')

For a single custom group, returns [*energy*, *instpwr*, *time*], which are as follows:

- *energy* : float

  Total energy use for the specified custom group in kilowatt hours.
- *instpwr* : int

  The instant power for the specified node in the custom group.
- *time* : int

  Time in seconds since activate or reset was last called.

For *custom_group*, specify the name of a predefined custom group.

If the custom group is not found, the function returns an empty list.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

NodesetPowerStatus('*custom_group*')

Returns a list of custom groups and custom group status readings in the following format:

```
{'custom_group':'status'}
{'custom_group':'status'}
.
.
.
```

Possible *status* readings are as follows:

- ON
- OFF
- BOOTED
- UNAVAILABLE

For *custom_group*, specify the name of a custom group.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

This function generates TAUserError if the node is not found or if a status cannot be retrieved.

NodesetThermalReport('*custom_group*')

Returns the average inlet temperature of all nodes in the custom group. The returned value excludes information about nonreporting nodes.

For *custom_group*, specify the name of a custom group.

For information about how to obtain a list of custom groups, see the following:

NodesetList()

PduEnergy('*pdu*')

Returns the power distribution unit (PDU) meter energy reading for *pdu*.

For *pdu*, specify the hostname of a PDU.

For information about how to obtain a list of PDU hostnames, see the following:

InventoryPdus()

The return value format is as follows:

```
[ energy, timestamp ]
```

The *energy* and *timestamp* values are floating-point values.

PduPower('*pdu*')

Returns the power distribution unit (PDU) meter power reading for *pdu*.

For *pdu*, specify the hostname of a PDU.

The return value format is as follows:

```
[ power, timestamp ]
```

The *power* and *timestamp* values are floating-point values.

For information about how to obtain a list of PDU hostnames, see the following:

InventoryPdus()

RackPduEnergy('*rack*')

Returns the sum of all power distribution unit (PDU) meter measurements in a cluster with leader nodes.

For *rack*, specify the name of one of the racks. For example, `rack1`.

The return format is as follows:

`[ energy, timestamp ]`

The *energy* and *timestamp* values are both floating-point values.

For information about how to obtain a list of rack names, see the following:

InventoryRackList()

RackPduPower('*rack*')

Returns the sum of all power distribution unit (PDU) meter measurements in a rack in a cluster with leader nodes.

For *rack*, specify the name of one of the racks. For example, `rack1`.

The returned format is as follows:

`[ power, timestamp ]`

The *power* and *timestamp* values are floating-point values.

For information about how to obtain a list of rack names, see the following:

InventoryRackList()

VerifyConnection()

Tests the connection from the client module to the cluster admin node.

Python session example

The following procedure explains how to start an interactive Python session that uses the main job-level power management API calls.

Procedure

1. Open a Python session and import the Python `sys` module.

   Set up the path environment to include the location of the power management software, which is `/opt/clmgr/power-service`. Then, import the job power API.

   For example:

   ```
   cluster:~ # python
   Python 2.7.9 (default, Jan 21 20XX, 11:02:59) [GCC] on linux2
   Type "help", "copyright", "credits" or "license" for more information.
   >>> import sys
   >>> sys.path.append("/opt/clmgr/power-service")
   >>> import hpe_clmgr_power_api as api
   ```

NOTE: The `import` command in this step renamed the `hpe_clmgr_power_api` to `api`, and the remaining steps in this procedure use that shorthand reference.

2. Enter the following API call to list the predefined custom groups:

```
>>> api.NodesetList()
['allleader', 'allservice', 'ice-compute']
```

The `allleader`, `allservice`, and `ice-compute` custom groups are the power management software default, predefined custom groups.

3. Enter the following API call to create an additional custom group called `mycustomgroup`:

```
>>> api.NodesetCreate('mycustomgroup', ['r1i1n0', 'r1i1n1', r1i1n2'])
```

Notice that the API call arguments in this step are, in fact, Python data types. Enclose each name in apostrophes (`' '`). Use the brackets (`[ ]`) as indicated in this step.

4. Enter the following API call to list the custom groups again:

```
>>> api.NodesetList()
['allleader', 'allservice', 'ice-compute', 'mycustomgroup']
```

5. Enter the following API call to list the content of a custom group:

```
>>> api.NodesetDetail('mycustomgroup')
['r1i1n0', 'r1i1n1', r1i1n2'])
```

6. Enter the following `MonitorStart` API call, which specifies a 500-watt power limit for each node:

```
>>> api.MonitorStart('mycustomgroup','500W')
```

Notice that a sufficiently high wattage value, or no value, has no effect on node power limits.

After `MonitorStart` commences, the `clmgr-power` service starts sampling power on each node and calculating the energy. This action continues indefinitely.

You can issue an API call at any time to display the current power and energy statistics. For example:

```
>>> api.MonitorReport('mycustomgroup') ['total_energy', 0.09220277777777777, 1488549378.816054, 8899, 8987,  8899, 9539]
```

The monitoring statistics are detailed in the documentation string of each API call. For example:

```
>>> print api.MonitorReport.__doc__
nodeset_name:  name of previously defined Nodeset

return: [ 'total_energy', energy, time, avepwr, instpwr, minpwr, maxpwr ]
  energy : float( total energy usage for all nodes in nodeset, in Kilowatt
Hours ),`
  time   : int( time in seconds since 'activate' or 'reset' last called ),
  avepwr : int( a sum of the average power stat for all nodes in the nodeset),
  instpwr: int( a sum of the instant power stat for all nodes in the nodeset),
  minpwr : int( a sum of the minimum power stat for all nodes in the nodeset),
  maxpwr : int( a sum of the maximum power stat for all nodes in the nodeset)

  throws TAUserError if nodeset_name not found

  nodeset has directed monitoring to stop (see MonitorStop).  >>>
```

In this example, the return values from the `MonitorStart` call are as follows:

- Energy
- Time stamp
- Average power
- Instantaneous power
- Minimum power
- Maximum power

7. Enter the following API call to return information about a single node in `mycustomgroup`:

```
 >>> api.MonitorReportForNode('mycustomgroup', 'r1i1n0')
[0.0015852777777777777, 1488549401.0, 143, 143, 135, 143]
```

This command is similar to the `MonitorReport` command.

8. Enter the following API call to adjust the power limit of each node:

```
>>> api.MonitorAdjust('ice-compute','400W')
[[True, None], [True, None]]
```

Alternatively, you could perform this step at the time of the `MonitorStart` call or while the custom group is being monitored using the `MonitorAdjust` call.

Notice that the return values for the command in this step are not meaningful.

9. Enter the following API call to check for nodes in the custom group that are encountering errors specific to collecting power-related information:

```
>>> api.MonitorReportError('ice-compute')
[]
```

This call returns a Python empty list `[]` because there are no errors. If actual node errors existed, the call returns a Python list of error nodes and messages.

10. Enter the following API call to stop monitoring the custom group:

```
>>> api.MonitorStop('ice-compute')
['total_energy', 4.6485797222222205, 1488549503.820233, 8615, 8660,
8615, 9993]
```

Notice that the output includes final power and energy statistics.

11. Enter the following API call to verify that the `clmgr-power` service is active and responsive:

```
>>> api.VerifyConnection()
'connection verified'
```

Troubleshooting the power commands

Under certain conditions, the power commands might return unexpected values or no values at all.

`Power service not available` message

The command you entered could not connect to the `clmgr-power` service.

On a cluster without leader nodes, ensure that the power service is running on the admin node.

On a cluster with leader nodes, ensure that the power service is running on both the admin node and on the leader nodes.

The following shows a check for the `clmgr-power` service running on the admin node:

```
# systemctl status clmgr-power
  clmgr-power.service - clmgr power
     Loaded: loaded (/usr/lib/systemd/system/clmgr-power.service; enabled; vendor preset: disabled)
     Active: active (running) since Fri 202X-07-15 17:51:11 IST; 1h 48min ago
   Main PID: 24365 (clmgr-power /op)
      Tasks: 14
     CGroup: /system.slice/clmgr-power.service
             └─24365 clmgr-power /opt/clmgr/power-service/tlib/twistd -o -n --pidfile= -y /opt/clmgr/power-service/sacmain.tac
```

The following shows a check for the `clmgr-power` service running on a leader node:

```
# ssh leader1 systemctl status clmgr-power
  clmgr-power.service - clmgr-power service
   Loaded: loaded (/usr/lib/systemd/system/clmgr-power.service; enabled;
vendor preset: enabled)
   Active: active (running) since Wed 202X-07-13 16:15:41 CDT; 1 day 16h ago
 Main PID: 25972 (platform-python)
    Tasks: 6 (limit: 2471484)
   Memory: 66.9M
   CGroup: /system.slice/clmgr-power.service
           ├─25972 clmgr-power /opt/clmgr/power-service/tlib/twistd --
originalname -o -n --pidfile= -y /opt/clmgr/power-service/rlcmain.tac
           └─26154 socat TCP-LISTEN:8888,fork TCP:admin:8888
```

If you do not retrieve similar output, including the entries ending with `sacmain.tac` or `rlcmain.tac`, see the following:

Configuring the power management software for rack-level, continuous AC power monitoring

NOTE: The output in this topic was wrapped for inclusion in this documentation.

`mpower` command troubleshooting - no response and no prompt, or no information returned but prompt appears

The information in this topic explains what to do when one of the following occurs:

- You enter an `mpower` command, but the system returns nothing. The system does not respond, nor does the system display the command prompt.
- You enter an `mpower` command, and the system response includes the typical list of statistics, but all the statistics have zero values.

In this case, complete the following troubleshooting steps:

1. Ensure that the `clmgr-power` service is running on the admin node and the leader nodes.

Check the leader nodes only on a cluster with leader nodes.

For information about how to complete this step, see the following:

`Power service not available` message

2. Check for any errors logged to `/opt/clmgr/log/clmgr-power.log` on the admin node and on any leader nodes.

`mpower` command troubleshooting - information is returned but no data

In the following example, the `mpower` command is run on a cluster with leader nodes. The command returns information, but no data:

```
# mpower system get_power
System Power Stats:
Instant : 124.00
Minimum during sampling period : 0.00
Maximum during sampling period : 0.00
Average during sampling period : 0.00
KwH during sampling period : 0.00
```

In this case, the `clmgr-power` service is not running on a leader node. To verify the situation, see the following log file on the admin node:

```
/opt/clmgr/log/clmgr-power.log
```

On the admin node, notice that the preceding log file includes the following entry:

```
20XX-01-26 13:50:36+0000 [LBroker,0,172.23.0.2] r1lead disconnected
```

After the `clmgr-power` service resumes on the leader node, the following entry appears in the log file:

```
20XX-01-26 13:52:22+0000 [LBroker,1,172.23.0.2] registered leader reference for r1lead
```

In this case, the `clmgr-power` service might be fully operational. However, the chassis controller might be having a problem generating power statistics.

Command options for online help, verbose output, and node-by-node thermal information

The `mpower` command accepts the following additional options for special circumstances:

- `mpower`

  `mpower -h`

  `mpower --help`

  These command formats return help information.

- `mpower -v` *`target_type action target_list value`*

  This command format returns verbose output for reports and errors.

- `mpower -m` *`target_type action target_list value`*

  This command format returns detailed, node-by-node information rather than a summary report. Specify `-m` only when issuing a request for thermal information.

Billing grade and power management

Power measurement as it relates to **billing grade** refers to a level of reliable and trustworthy measurement precision that is sufficient to be used for commercial energy billing. ANSI standard C12.20 and its equivalent IEC standard describe billing grade precision. The standard defines the billing grade accuracy and billing grade performance, as follows:

- The billing grade accuracy level must be no greater than 1%. More often, the billing grade standard is 0.05%.

  The power measurement capability for clusters with leader nodes meets the billing grade standard for accuracy.
- Billing grade performance requires that measurement are taken on a continuous, reliable basis.

  The clusters with leader nodes do not meet the billing grade performance standard.

Measuring AC power and energy

To measure AC power and energy, clusters use a measurement appliance or meter. The devices measure each AC feed coming into an E rack. The meters meet the ANSI C12.20-2010 standard and are billing-grade devices. For information about supported measurement appliances or meters, see the HPE Performance Cluster Manager release notes.

The measurement appliances are connected through Ethernet to the rack management infrastructure. General system management operations use the rack management infrastructure for operations such as the following:

- Reboots
- Resets
- Firmware upgrades

The preceding events can halt the reporting functions between the measurement appliances and the data collection infrastructure. This loss can last a few seconds or, in some cases, minutes.

Much of this disruption is due to planned system restarts or upgrades. The brief loss of system management capability does not interfere with the ongoing energy measurement performed by the measurement appliance. Only current power readings are affected.

Node power

Node power measurements are accomplished by several means, depending on the hardware type. Node power monitoring is not capable of performing sampling more than once per second and is not billing grade accurate.

Support and other resources

Accessing Hewlett Packard Enterprise Support

- For live assistance, go to the Contact Hewlett Packard Enterprise Worldwide website:

  https://www.hpe.com/info/assistance

- To access documentation and support services, go to the Hewlett Packard Enterprise Support Center website:

  https://www.hpe.com/support/hpesc

Information to collect

- Technical support registration number (if applicable)
- Product name, model or version, and serial number
- Operating system name and version
- Firmware version
- Error messages
- Product-specific reports and logs
- Add-on products or components
- Third-party products or components

Accessing updates

- Some software products provide a mechanism for accessing software updates through the product interface. Review your product documentation to identify the recommended software update method.
- To download product updates:

  Hewlett Packard Enterprise Support Center

  https://www.hpe.com/support/hpesc

  Hewlett Packard Enterprise Support Center: Software downloads

  https://www.hpe.com/support/downloads

  My HPE Software Center

  https://www.hpe.com/software/hpesoftwarecenter

- To subscribe to eNewsletters and alerts:

  https://www.hpe.com/support/e-updates

- To view and update your entitlements, and to link your contracts and warranties with your profile, go to the Hewlett Packard Enterprise Support Center **More Information on Access to Support Materials** page:

  https://www.hpe.com/support/AccessToSupportMaterials

IMPORTANT: Access to some updates might require product entitlement when accessed through the Hewlett Packard Enterprise Support Center. You must have an HPE Onepass set up with relevant entitlements.

Remote support

Remote support is available with supported devices as part of your warranty or contractual support agreement. It provides intelligent event diagnosis, and automatic, secure submission of hardware event notifications to Hewlett Packard Enterprise, which initiates a fast and accurate resolution based on the service level of your product. Hewlett Packard Enterprise strongly recommends that you register your device for remote support.

If your product includes additional remote support details, use search to locate that information.

HPE Get Connected

https://www.hpe.com/services/getconnected

HPE Pointnext Tech Care

https://www.hpe.com/services/techcare

HPE Complete Care

https://www.hpe.com/services/completecare

Customer self repair

Hewlett Packard Enterprise customer self repair (CSR) programs allow you to repair your product. If a CSR part needs to be replaced, it will be shipped directly to you so that you can install it at your convenience. Some parts do not qualify for CSR. Your Hewlett Packard Enterprise authorized service provider will determine whether a repair can be accomplished by CSR.

For more information about CSR, contact your local service provider.

Warranty information

To view the warranty information for your product, see the links provided below:

HPE ProLiant and IA-32 Servers and Options

https://www.hpe.com/support/ProLiantServers-Warranties

HPE Enterprise and Cloudline Servers

https://www.hpe.com/support/EnterpriseServers-Warranties

HPE Storage Products

https://www.hpe.com/support/Storage-Warranties

HPE Networking Products

https://www.hpe.com/support/Networking-Warranties

Regulatory information

To view the regulatory information for your product, view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products*, available at the Hewlett Packard Enterprise Support Center:

https://www.hpe.com/support/Safety-Compliance-EnterpriseProducts

Additional regulatory information

Hewlett Packard Enterprise is committed to providing our customers with information about the chemical substances in our products as needed to comply with legal requirements such as REACH (Regulation EC No 1907/2006 of the European Parliament and the Council). A chemical information report for this product can be found at:

https://www.hpe.com/info/reach

For Hewlett Packard Enterprise product environmental and safety information and compliance data, including RoHS and REACH, see:

https://www.hpe.com/info/ecodata

For Hewlett Packard Enterprise environmental information, including company programs, product recycling, and energy efficiency, see:

https://www.hpe.com/info/environment

Documentation feedback

Hewlett Packard Enterprise is committed to providing documentation that meets your needs. To help us improve the documentation, use the **Feedback** button and icons (located at the bottom of an opened document) on the Hewlett Packard Enterprise Support Center portal (**https://www.hpe.com/support/hpesc**) to send any errors, suggestions, or comments. All document information is captured by the process.