ROCm SMI LIB Documentation

Release 7.8.0

Advanced Micro Devices, Inc.

Sep 17, 2025

INSTALLATION

The ROCm SMI library, is part of the ROCm software stack. It is a C++ library for Linux that provides a user space interface for applications to monitor and control GPU applications.

For more information, refer to https://github.com/ROCm/rocm_smi_lib.

Note

The AMD System Management Interface Library (AMD SMI) is a successor to ROCm SMI. It is a unified system management interface tool that provides a user space interface for applications to monitor and control GPU applications and gives users the ability to query information about drivers and GPUs on the system.

AMD SMI will replace `rocm_smi_lib` over time. We recommend that users transition to AMD SMI.

For more information, refer to https://github.com/ROCm/amdsmi and the AMD SMI documentation.

Install

- *ROCm SMI installation*

API Reference

- *Files*
- *Globals*
- *Data structures*
- *Modules*
- *Python API*

How to

- *Use C++ in ROCm SMI*
- *Use Python in ROCm SMI*

Tutorials

- *C++*
- *Python*

To contribute to the documentation, refer to Contributing to ROCm.

You can find licensing information on the Licensing page.

CHAPTER

ONE

INSTALLING ROCM SMI

1.1 Planned deprecation notice

ROCm System Management Interface (ROCm SMI) Library is planned to be ***deprecated***, and the release date will be announced soon. We recommend migration to AMD SMI.

1.2 Install amdgpu using ROCm

Use the following instructions to install AMDGPU using ROCm:

1. Install amdgpu driver. Refer to the following example, your release and link may differ. The *amdgpu-install –usecase=rocm* triggers both an amdgpu driver update and ROCm SMI packages to be installed on your device.

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/6.0.2/ubuntu/jammy/amdgpu-install_6.0.60002-
↪1_all.deb
sudo apt install ./amdgpu-install_6.0.60002-1_all.deb
sudo amdgpu-install --usecase=rocm
```

- *rocm-smi –help*

1.2.1 Building ROCm SMI

1.2.1.1 Addtional required software

To build the ROCm SMI library, the following components are required.

The following software versions are what was used in development. Earlier versions are not guaranteed to work:

- CMake (v3.14.0)
- g++ (5.4.0)

To build the latest documentation, the following are required:

- Python 3.8+
- NPM (sass)

The source code for ROCm SMI is available on Github.

After the ROCm SMI library git repository is cloned to a local Linux machine, use the following CMake build sequence to build the library. Specifically,

```
mkdir -p build
cd build
cmake ..
make -j $(nproc)
# Install library file and header; default location is /opt/rocm
make install
```

The built library will appear in the *build* folder.

To build the rpm and deb packages follow the above steps with:

```
make package
```

1.2.1.2 Building documentation

The following is an example of how to build the docs:

```
python3 -m venv .venv
.venv/bin/python3 -m pip install -r docs/sphinx/requirements.txt
.venv/bin/python3 -m sphinx -T -E -b html -d docs/_build/doctrees -D language=en docs␣
↪docs/_build/html
```

1.2.1.3 Building tests

To verify the build and capability of ROCm SMI on your system and to see an example of how ROCm SMI can be used, you may build and run the tests that are available in the repo. To build the tests, follow these steps:

```
mkdir build
cd build
cmake -DBUILD_TESTS=ON ..
make -j $(nproc)
```

To run the test, execute the program *rsmitst* that is built from the preceding steps.

CHAPTER

TWO

USING ROCM SMI WITH C++

2.1 Installation

Follow the installation procedure for rocm_smi_lib. Refer to the *installation section*.

> **Note**
>
> `hipcc` and other compilers will not automatically link in the `librocm_smi64` dynamic library. To ensure the `librocm_smi64.so` can be located, you must either set the `LD_LIBRARY_PATH` environment variable to the directory containing `librocm_smi64.so` (usually `/opt/rocm/lib`) or pass the `-lrocm_smi64` flag to the compiler.

2.2 Device indices

Many of the functions in the library take a "device index". The device index is a number greater than or equal to 0, and less than the number of devices detected, as determined by *rsmi_num_monitor_devices()*. The index is used to distinguish the detected devices from one another. It is important to note that a device may end up with a different index after a reboot, so an index should not be relied upon to be constant over reboots.

2.3 Hello ROCm SMI

The only required ROCm-SMI call for any program that wants to use ROCm-SMI is the `rsmi_init()` call. This call initializes some internal data structures that will be used by subsequent ROCm-SMI calls.

When ROCm-SMI is no longer being used, `rsmi_shut_down()` should be called. This provides a way to do any releasing of resources that ROCm-SMI may have held. In many cases, this may have no effect, but may be necessary in future versions of the library.

A simple "Hello World" type program that displays the device ID of detected devices would look like this:

```
#include <stdint.h>
#include "rocm_smi/rocm_smi.h"
int main() {
  rsmi_status_t ret;
  uint32_t num_devices;
  uint16_t dev_id;

  // We will skip return code checks for this example, but it
```

(continues on next page)

(continued from previous page)

```
  // is recommended to always check this as some calls may not
  // apply for some devices or ROCm releases

  ret = rsmi_init(0);
  ret = rsmi_num_monitor_devices(&num_devices);

  for (int i=0; i < num_devices; ++i) {
    ret = rsmi_dev_id_get(i, &dev_id);
    // dev_id holds the device ID of device i, upon a
    // successful call
  }
  ret = rsmi_shut_down();
  return 0;
}
```

CHAPTER

THREE

USING ROCM SMI WITH PYTHON

This tool acts as a command line interface for manipulating and monitoring the amdgpu kernel, and is intended to replace and deprecate the existing `rocm_smi.py` CLI tool. It uses Ctypes to call the `rocm_smi_lib` API.

To use ROCm SMI, you must have ROCm SMI library (`librocm_smi64`) installed and should have at least one AMD GPU with the ROCm driver installed.

3.1 Installation

Follow the installation procedure for `rocm_smi_lib`. Refer to *Installing ROCm SMI*.

> **Note**
>
> `hipcc` and other compilers will not automatically link in the `librocm_smi64` dynamic library. To ensure the `librocm_smi64.so` can be located, you must either set the `LD_LIBRARY_PATH` environment variable to the directory containing `librocm_smi64.so` (usually `/opt/rocm/lib` or pass the `-lrocm_smi64` flag to the compiler.

3.2 Version

The SMI will report two "versions": the `ROCM-SMI` version and the `ROCM-SMI-LIB` version.

- `ROCM-SMI` version is the CLI/tool version number with commit ID appended after + sign.
- `ROCM-SMI-LIB` version is the library package version number.

```
ROCM-SMI version: 2.0.0+8e78352
ROCM-SMI-LIB version: 6.1.0
```

3.3 Usage

For detailed and up to date usage information, consult the help.

```
/opt/rocm/bin/rocm-smi -h
```

The following is the output from the `-h` flag:

```
$ /opt/rocm/bin/rocm-smi -h
usage: rocm-smi [-h] [-V] [-d DEVICE [DEVICE ...]] [--alldevices] [--showhw] [-a] [-i] [-
↪v] [-e [EVENT [EVENT ...]]]
                [--showdriverversion] [--showtempgraph] [--showfwinfo [BLOCK [BLOCK ...
↪]]] [--showmclkrange]
                [--showmemvendor] [--showsclkrange] [--showproductname] [--showserial] [-
↪-showuniqueid]
                [--showvoltagerange] [--showbus] [--showpagesinfo] [--showpendingpages]␣
↪[--showretiredpages]
                [--showunreservablepages] [-f] [-P] [-t] [-u] [--showmemuse] [--
↪showvoltage] [-b] [-c] [-g] [-l] [-M]
                [-m] [-o] [-p] [-S] [-s] [--showmeminfo TYPE [TYPE ...]] [--showpids␣
↪[VERBOSE]]
                [--showpidgpus [SHOWPIDGPUS [SHOWPIDGPUS ...]]] [--showreplaycount]
                [--showrasinfo [SHOWRASINFO [SHOWRASINFO ...]]] [--showvc] [--
↪showxgmierr] [--showtopo]
                [--showtopoaccess] [--showtopoweight] [--showtopohops] [--showtopotype]␣
↪[--showtoponuma]
                [--showenergycounter] [--shownodesbw] [--showcomputepartition] [--
↪showmemorypartition] [-r]
                [--resetfans] [--resetprofile] [--resetpoweroverdrive] [--resetxgmierr]␣
↪[--resetperfdeterminism]
                [--resetcomputepartition] [--resetmemorypartition] [--setclock TYPE␣
↪LEVEL] [--setsclk LEVEL [LEVEL ...]]
                [--setmclk LEVEL [LEVEL ...]] [--setpcie LEVEL [LEVEL ...]] [--setslevel␣
↪SCLKLEVEL SCLK SVOLT]
                [--setmlevel MCLKLEVEL MCLK MVOLT] [--setvc POINT SCLK SVOLT] [--
↪setsrange SCLKMIN SCLKMAX]
                [--setextremum min|max sclk|mclk CLK] [--setmrange MCLKMIN MCLKMAX] [--
↪setfan LEVEL]
                [--setperflevel LEVEL] [--setoverdrive %] [--setmemoverdrive %] [--
↪setpoweroverdrive WATTS]
                [--setprofile SETPROFILE] [--setperfdeterminism SCLK]
                [--setcomputepartition {CPX,SPX,DPX,TPX,QPX,cpx,spx,dpx,tpx,qpx}]
                [--setmemorypartition {NPS1,NPS2,NPS4,NPS8,nps1,nps2,nps4,nps8}] [--
↪rasenable BLOCK ERRTYPE]
                [--rasdisable BLOCK ERRTYPE] [--rasinject BLOCK] [--gpureset] [--load␣
↪FILE | --save FILE]
                [--autorespond RESPONSE] [--loglevel LEVEL] [--json] [--csv]

AMD ROCm System Management Interface | ROCM-SMI version: 2.0.0+8e78352

optional arguments:
  -h, --help                                                    show this help␣
↪message and exit
  --gpureset                                                    Reset specified GPU␣
↪(One GPU must be specified)
  --load FILE                                                   Load Clock, Fan,␣
↪Performance and Profile settings
                                                                from FILE
  --save FILE                                                   Save Clock, Fan,␣
↪Performance and Profile settings to
                                                                FILE
```

(continues on next page)

(continued from previous page)

```
  -V, --version                                                 Show version␣
↪information

  -d DEVICE [DEVICE ...], --device DEVICE [DEVICE ...]          Execute command on␣
↪specified device

Display Options:
  --alldevices
  --showhw                                                      Show Hardware details
  -a, --showallinfo                                             Show Temperature, Fan␣
↪and Clock values

Topology:
  -i, --showid                                                  Show DEVICE ID
  -v, --showvbios                                               Show VBIOS version
  -e [EVENT [EVENT ...]], --showevents [EVENT [EVENT ...]]      Show event list
  --showdriverversion                                           Show kernel driver␣
↪version
  --showtempgraph                                               Show Temperature Graph
  --showfwinfo [BLOCK [BLOCK ...]]                              Show FW information
  --showmclkrange                                               Show mclk range
  --showmemvendor                                               Show GPU memory vendor
  --showsclkrange                                               Show sclk range
  --showproductname                                             Show SKU/Vendor name
  --showserial                                                  Show GPU's Serial␣
↪Number
  --showuniqueid                                                Show GPU's Unique ID
  --showvoltagerange                                            Show voltage range
  --showbus                                                     Show PCI bus number

Pages information:
  --showpagesinfo                                               Show retired, pending␣
↪and unreservable pages
  --showpendingpages                                            Show pending retired␣
↪pages
  --showretiredpages                                            Show retired pages
  --showunreservablepages                                       Show unreservable␣
↪pages

Hardware-related information:
  -f, --showfan                                                 Show current fan speed
  -P, --showpower                                               Show current Average␣
↪or Socket Graphics Package Power
                                                                Consumption
  -t, --showtemp                                                Show current␣
↪temperature
  -u, --showuse                                                 Show current GPU use
  --showmemuse                                                  Show current GPU␣
↪memory used
  --showvoltage                                                 Show current GPU␣
↪voltage
```

(continues on next page)

(continued from previous page)

```
Software-related/controlled information:
  -b, --showbw                                                  Show estimated PCIe␣
↪use
  -c, --showclocks                                              Show current clock␣
↪frequencies
  -g, --showgpuclocks                                           Show current GPU␣
↪clock frequencies
  -l, --showprofile                                             Show Compute Profile␣
↪attributes
  -M, --showmaxpower                                            Show maximum graphics␣
↪package power this GPU will
                                                                consume
  -m, --showmemoverdrive                                        Show current GPU␣
↪Memory Clock OverDrive level
  -o, --showoverdrive                                           Show current GPU␣
↪Clock OverDrive level
  -p, --showperflevel                                           Show current DPM␣
↪Performance Level
  -S, --showclkvolt                                             Show supported GPU␣
↪and Memory Clocks and Voltages
  -s, --showclkfrq                                              Show supported GPU␣
↪and Memory Clock
  --showmeminfo TYPE [TYPE ...]                                 Show Memory usage␣
↪information for given block(s) TYPE
  --showpids [VERBOSE]                                          Show current running␣
↪KFD PIDs (pass details to
                                                                VERBOSE for detailed␣
↪information)
  --showpidgpus [SHOWPIDGPUS [SHOWPIDGPUS ...]]                 Show GPUs used by␣
↪specified KFD PIDs (all if no arg
                                                                given)
  --showreplaycount                                             Show PCIe Replay Count
  --showrasinfo [SHOWRASINFO [SHOWRASINFO ...]]                 Show RAS enablement␣
↪information and error counts for
                                                                the specified␣
↪block(s) (all if no arg given)
  --showvc                                                      Show voltage curve
  --showxgmierr                                                 Show XGMI error␣
↪information since last read
  --showtopo                                                    Show hardware␣
↪topology information
  --showtopoaccess                                              Shows the link␣
↪accessibility between GPUs
  --showtopoweight                                              Shows the relative␣
↪weight between GPUs
  --showtopohops                                                Shows the number of␣
↪hops between GPUs
  --showtopotype                                                Shows the link type␣
↪between GPUs
  --showtoponuma                                                Shows the numa nodes
  --showenergycounter                                           Energy accumulator␣
```

(continues on next page)

(continued from previous page)

```
↪that stores amount of energy
                                                                consumed
  --shownodesbw                                                 Shows the numa nodes
  --showcomputepartition                                        Shows current compute␣
↪partitioning
  --showmemorypartition                                         Shows current memory␣
↪partition

Set options:
  --setclock TYPE LEVEL                                         Set Clock Frequency␣
↪Level(s) for specified clock
                                                                (requires manual Perf␣
↪level)
  --setsclk LEVEL [LEVEL ...]                                   Set GPU Clock␣
↪Frequency Level(s) (requires manual
                                                                Perf level)
  --setmclk LEVEL [LEVEL ...]                                   Set GPU Memory Clock␣
↪Frequency Level(s) (requires
                                                                manual Perf level)
  --setpcie LEVEL [LEVEL ...]                                   Set PCIE Clock␣
↪Frequency Level(s) (requires manual
                                                                Perf level)
  --setslevel SCLKLEVEL SCLK SVOLT                              Change GPU Clock␣
↪frequency (MHz) and Voltage (mV) for
                                                                a specific Level
  --setmlevel MCLKLEVEL MCLK MVOLT                              Change GPU Memory␣
↪clock frequency (MHz) and Voltage
                                                                for (mV) a specific␣
↪Level
  --setvc POINT SCLK SVOLT                                      Change SCLK Voltage␣
↪Curve (MHz mV) for a specific
                                                                 point
  --setsrange SCLKMIN SCLKMAX                                   Set min and max SCLK␣
↪speed
  --setextremum min|max sclk|mclk CLK                           Set min/max of SCLK/
↪MCLK speed
  --setmrange MCLKMIN MCLKMAX                                   Set min and max MCLK␣
↪speed
  --setfan LEVEL                                                Set GPU Fan Speed␣
↪(Level or %)
  --setperflevel LEVEL                                          Set Performance Level
  --setoverdrive %                                              Set GPU OverDrive␣
↪level (requires manual|high Perf
                                                                level)
  --setmemoverdrive %                                           Set GPU Memory␣
↪Overclock OverDrive level (requires
                                                                manual|high Perf␣
↪level)
  --setpoweroverdrive WATTS                                     Set the maximum GPU␣
↪power using Power OverDrive in
                                                                Watts
  --setprofile SETPROFILE                                       Specify Power Profile␣
```

(continues on next page)

(continued from previous page)

```
↪level (#) or a quoted string of
                                                                    CUSTOM Profile␣
↪attributes "# # # #..." (requires
                                                                    manual Perf level)
  --setperfdeterminism SCLK                                         Set clock frequency␣
↪limit to get minimal performance
                                                                    variation
  --setcomputepartition {CPX,SPX,DPX,TPX,QPX,cpx,spx,dpx,tpx,qpx}   Set compute partition
  --setmemorypartition {NPS1,NPS2,NPS4,NPS8,nps1,nps2,nps4,nps8}    Set memory partition
  --rasenable BLOCK ERRTYPE                                         Enable RAS for␣
↪specified block and error type
  --rasdisable BLOCK ERRTYPE                                        Disable RAS for␣
↪specified block and error type
  --rasinject BLOCK                                                 Inject RAS poison for␣
↪specified block (ONLY WORKS ON
                                                                     UNSECURE BOARDS)

Reset options:
  -r, --resetclocks                                                 Reset clocks and␣
↪OverDrive to default
  --resetfans                                                       Reset fans to␣
↪automatic (driver) control
  --resetprofile                                                    Reset Power Profile␣
↪back to default
  --resetpoweroverdrive                                             Set the maximum GPU␣
↪power back to the device deafult
                                                                    state
  --resetxgmierr                                                    Reset XGMI error count
  --resetperfdeterminism                                            Disable performance␣
↪determinism
  --resetcomputepartition                                           Resets to boot␣
↪compute partition state
  --resetmemorypartition                                            Resets to boot memory␣
↪partition state

Auto-response options:
  --autorespond RESPONSE                                            Response to␣
↪automatically provide for all prompts
                                                                    (NOT RECOMMENDED)

Output options:
  --loglevel LEVEL                                                  How much output will␣
↪be printed for what program is
                                                                    doing, one of debug/
↪info/warning/error/critical
  --json                                                            Print output in JSON␣
↪format
  --csv                                                             Print output in CSV␣
↪format
```

3.4 Detailed option descriptions

--setextremum <[min or max] [sclk or mclk] [value in MHz to set to]> Provided ASIC support, users can now set a maximum or minimum sclk or mclk value through our Python CLI tool (*rocm-smi –setextremum max sclk 1500*). See example below.

```
$ sudo /opt/rocm/bin/rocm-smi --setextremum max sclk 2100

========================== ROCm System Management␣
↪Interface ===========================

          ******WARNING******

          Operating your AMD GPU outside of official AMD␣
↪specifications or outside of
          factory settings, including but not limited to the␣
↪conducting of overclocking,
          over-volting or under-volting (including use of␣
↪this interface software,
          even if such software has been directly or␣
↪indirectly provided by AMD or otherwise
          affiliated in any way with AMD), may cause damage␣
↪to your AMD GPU, system components
          and/or result in system failure, as well as cause␣
↪other problems.
          DAMAGES CAUSED BY USE OF YOUR AMD GPU OUTSIDE OF␣
↪OFFICIAL AMD SPECIFICATIONS OR
          OUTSIDE OF FACTORY SETTINGS ARE NOT COVERED UNDER␣
↪ANY AMD PRODUCT WARRANTY AND
          MAY NOT BE COVERED BY YOUR BOARD OR SYSTEM␣
↪MANUFACTURER'S WARRANTY.
          Use this utility with caution.

Do you accept these terms? [y/N] y
================================= Set Valid sclk Extremum␣
↪==================================
GPU[0]          : Successfully set max sclk to 2100(MHz)
GPU[1]          : Successfully set max sclk to 2100(MHz)
GPU[2]          : Successfully set max sclk to 2100(MHz)
GPU[3]          : Successfully set max sclk to 2100(MHz)
================================= End of ROCm SMI Log␣
↪==================================
```

--setsclk, --setmclk <# [# # . . .]> This allows you to set a mask for the levels. For example, if a GPU has 8 clock levels, you can set a mask to use levels 0, 5, 6 and 7 with –setsclk 0 5 6 7 . This will only use the base level, and the top 3 clock levels. This will allow you to keep the GPU at base level when there is no GPU load, and the top 3 levels when the GPU load increases.

> **Note**
>
> The clock levels will change dynamically based on GPU load based on the default Compute and Graphics profiles. The thresholds and delays for a custom

mask cannot be controlled through the SMI tool.

This flag automatically sets the Performance Level to `manual` as the mask is not applied when the Performance level is set to `auto`.

--setfan <LEVEL> This sets the fan speed to a value ranging from 0 to maxlevel, or from 0%-100%.

If the level ends with a %, the fan speed is calculated as pct*maxlevel/100 (maxlevel is usually 255, but is determined by the ASIC).

Note

While the hardware is usually capable of overriding this value when required, it is recommended to not set the fan level lower than the default value for extended periods of time.

--setperflevel <LEVEL> This lets you use the pre-defined Performance Level values for clocks and power profile, which can include:

- `auto` (Automatically change values based on GPU workload)
- `low` (Keep values low, regardless of workload)
- `high` (Keep values high, regardless of workload)
- `manual` (Only use values defined by `--setsclk` and `--setmclk`)

Warning

DEPRECATED IN NEWER KERNEL VERSIONS. Use `--setslevel` or `--setmlevel` instead.

This sets the percentage above maximum for the max Performance Level. For example, `--setoverdrive` 20 will increase the top sclk level by 20%, similarly `--setmemoverdrive` 20 will increase the top mclk level by 20%. Thus if the maximum clock level is 1000MHz, then `--setoverdrive` 20 will increase the maximum clock to 1200MHz.

Note

This option can be used in conjunction with the `--setsclk`/`--setmclk` mask.

Operating the GPU outside of specifications can cause irreparable damage to your hardware Observe the warning displayed when using this option.

This flag automatically sets the clock to the highest level, as only the highest level is increased by the *OverDrive* value.

--setoverdrive, --setpoweroverdrive <#>, --resetpoweroverdrive <#> This allows users to change the maximum power available to a GPU package. The input value is in Watts. This limit is enforced by the hardware, and some cards allow users to set it to a higher value than the default that ships with the GPU. This Power OverDrive mode allows the GPU to run at higher frequencies for longer periods of time, though this

may mean the GPU uses more power than it is allowed to use per power supply specifications. Each GPU has a model-specific maximum Power OverDrive that is will take; attempting to set a higher limit than that will cause this command to fail.

> **Note**
>
> Operating the GPU outside of specifications can cause irreparable damage to your hardware. Observe the warning displayed when using this option.

--setprofile <SETPROFILE> The Compute Profile accepts 1 or n parameters, either the Profile to select (see `--showprofile` for a list of preset Power Profiles) or a quoted string of values for the CUSTOM profile. These values can vary based on the ASIC, and may include:

Setting	Description
SCLK_PROFILE_ENABLE	Whether or not to apply the following SCLK settings (0=disable,1=enable)
	NOTE: This is a hidden field. If set to 0, the following 3 values are displayed as '-'
SCLK_UP_HYST	Delay before sclk is increased (in milliseconds)
SCLK_DOWN_HYST	Delay before sclk is decresed (in milliseconds)
SCLK_ACTIVE_LEVEL	Workload required before sclk levels change (in %)
MCLK_PROFILE_ENABLE	Whether or not to apply the following MCLK settings (0=disable,1=enable)
	NOTE: This is a hidden field. If set to 0, the following 3 values are displayed as '-'
MCLK_UP_HYST	Delay before mclk is increased (in milliseconds)
MCLK_DOWN_HYST	Delay before mclk is decresed (in milliseconds)
MCLK_ACTIVE_LEVEL	Workload required before mclk levels change (in %)

3.4.1 Other settings

Setting	Description
BUSY_SET_POINT	Threshold for raw activity level before levels change
FPS	Frames Per Second

(continues on next page)

(continued from previous page)

```
↪       |
| USE_RLC_BUSY     | When set to 1, DPM is switched up as long as RLC busy message is␣
↪received |
| MIN_ACTIVE_LEVEL | Workload required before levels change (in %)                  ␣
↪       |
```

Note

When a compute queue is detected, the COMPUTE Power Profile values will be automatically applied to the system, provided that the Perf Level is set to `auto`.

The CUSTOM Power Profile is only applied when the Performance Level is set to `manual` so using this flag will automatically set the performance level to `manual`.

It is not possible to modify the non-CUSTOM Profiles. These are hard-coded by the kernel.

-P, --showpower Show average or instantaneous socket graphics package power consumption.

"Graphics Package" refers to the GPU plus any HBM (High-Bandwidth memory) modules, if present.

-M, --showmaxpower Show the maximum Graphics Package power that the GPU will attempt to consume.

This limit is enforced by the hardware.

--loglevel This will allow the user to set a logging level for the SMI's actions. Currently this is only implemented for `sysfs` writes, but can easily be expanded upon in the future to log other things from the SMI.

--showmeminfo This allows the user to see the amount of used and total memory for a given block (`vram`, `vis_vram`, `gtt`). It returns the number of bytes used and total number of bytes for each block `all` can be passed as a field to return all blocks, otherwise a quoted-string is used for multiple values (such as `vram vis_vram`).

`vram` refers to the Video RAM, or graphics memory, on the specified device `vis_vram` refers to Visible VRAM, which is the CPU-accessible video memory on the device `gtt` refers to the Graphics Translation Table

-b, --showbw This shows an approximation of the number of bytes received and sent by the GPU over the last second through the PCIe bus. Note that this will not work for APUs since data for the GPU portion of the APU goes through the memory fabric and does not 'enter/exit' the chip via the PCIe interface, thus no accesses are generated, and the performance counters can't count accesses that are not generated.

Note

It is not possible to easily grab the size of every packet that is transmitted in real time, so the kernel estimates the bandwidth by taking the maximum payload size (mps), which is the max size that a PCIe packet can be. and multiplies it by the number of packets received and sent. This means that the SMI will report the maximum estimated bandwidth, the actual usage could (and likely will be) less.

--showrasinfo This shows the RAS information for a given block. This includes enablement of

the block (currently GFX, SDMA and UMC are the only supported blocks) and the number of errors.

- `ue` - Uncorrectable errors
- `ce` - Correctable errors

3.5 Clock type descriptions

Clock type	Description
DCEFCLK	DCE (Display)
FCLK	Data fabric (VG20 and later) - Data flow from XGMI, Memory, PCIe
SCLK	GFXCLK (Graphics core)
	Note - SOCCLK split from SCLK as of Vega10. Pre-Vega10 they were both controlled by SCLK
MCLK	GPU Memory (VRAM)
PCLK	PCIe bus
	Note - This gives 2 speeds, PCIe Gen1 x1 and the highest available based on the hardware
SOCCLK	System clock (VG10 and later) - Data Fabric (DF), MM HUB, AT HUB, SYSTEMHUB, OSS, DFD
	Note - DF split from SOCCLK as of Vega20. Pre-Vega20 they were both controlled by SOCCLK

--gpureset This flag will attempt to reset the GPU for a specified device. This will invoke the GPU reset through the kernel debugfs file `amdgpu_gpu_recover`. Note that GPU reset will not always work, depending on the manner in which the GPU is hung.

--showdriverversion This flag will print out the AMDGPU module version for amdgpu-pro or ROCm kernels. For other kernels, it will simply print out the name of the kernel (`uname -r`).

--showserial This flag will print out the serial number for the graphics card.

> **Note**
>
> This is currently only supported on Vega20 server cards that support it. Consumer cards and cards older than Vega20 will not support this feature.

--showproductname This uses the `pci.ids` file to print out more information regarding the GPUs on the system. `update-pciids` may need to be executed on the machine to get the latest PCI ID snapshot, as certain newer GPUs will not be present in the stock `pci.ids` file, and the file may even be absent on certain OS installation types.

--showpagesinfo, --showretiredpages, --showpendingpages, --showunreservablepages These flags display the different "bad pages" as reported by the kernel. The three types of pages are:

- Retired pages (reserved pages) - These pages are reserved and are unable to be used.
- Pending pages - These pages are pending for reservation, and will be reserved/retired.
- Unreservable pages - These pages are not reservable for some reason.

--showmemuse, --showuse, --showmeminfo `--showuse` and `--showmemuse` are used to indicate how busy the respective blocks are. For example, for `--showuse (gpu_busy_percent sysfs file)`, the SMU samples every ms or so to see if any GPU block (RLC, MEC, PFP, CP) is busy. If so, that's 1 (or high). If not, that's 0 (low). If we have 5 high and 5 low samples, that means 50% utilization (50% GPU busy, or 50% GPU use). The windows and sampling vary from generation to generation, but that is how GPU and VRAM use is calculated in a generic sense. `--showmeminfo` (and VRAM% in concise output) will show the amount of VRAM used (visible, total, GTT), as well as the total available for those partitions. The percentage shown there indicates the amount of used memory in terms of current allocations.

3.6 OverDrive settings

Enabling OverDrive requires both a card that support OverDrive and a driver parameter that enables its use. Because OverDrive features can damage your card, most workstation and server GPUs cannot use OverDrive. Consumer GPUs that can use OverDrive must enable this feature by setting bit 14 in the amdgpu driver's ppfeaturemask module parameter

For OverDrive functionality, the OverDrive bit (bit 14) must be enabled (by default, the OverDrive bit is disabled on the ROCK and upstream kernels). This can be done by setting amdgpu.ppfeaturemask accordingly in the kernel parameters, or by changing the default value inside amdgpu_drv.c (if building your own kernel).

As an example, if the `ppfeaturemask` is set to `0xffffbfff` (11111111111111110111111111111111), then enabling the OverDrive bit would make it `0xffffffff` (11111111111111111111111111111111).

These are the flags that require OverDrive functionality to be enabled for the flag to work:

- `--showclkvolt`
- `--showvoltagerange`
- `--showvc`
- `--showsclkrange`
- `--showmclkrange`
- `--setslevel`
- `--setmlevel`
- `--setoverdrive`
- `--setpoweroverdrive`
- `--resetpoweroverdrive`
- `--setvc`
- `--setsrange`

- `--setmrange`

CHAPTER

FOUR

FILE LIST

CHAPTER

FIVE

GLOBALS

CHAPTER

SIX

DATA STRUCTURES

CHAPTER

SEVEN

MODULES

CHAPTER

EIGHT

PYTHON API REFERENCE

This section describes the ROCm SMI Python module API.

8.1 Functions

rocm_smi.**checkAmdGpus**(*deviceList*)

Check if there are any AMD GPUs being queried, return False if there are none

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**checkIfSecondaryDie**(*device*)

Checks if GCD(die) is the secondary die in a MCM. MI200 device specific feature check. The secondary dies lacks power management features.

Parameters
device – The device to check

rocm_smi.**check_runtime_status**() → bool

Check the runtime status of all AMD GPU devices managed by the amdgpu driver.

This function scans the directories under the specified path to verify the runtime power management status of each device. It checks the "runtime_status" file for each device to determine if the device is in an "active" state. If any device is not in an "active" state it returns False. If the file is inaccessible, this may be due to a system that does not support runtime power management. Some GPUs support runtime power management, while others may not. This is why the default status is set to True.

bool: False if any device is not in "active" state, True otherwise.

rocm_smi.**component_str**(*component*)

Returns the component String value

Parameters
component – Component (currently only driver)

rocm_smi.**confirmChangingMemoryPartitionAndReloadingAMDGPU**(*autoRespond*)

Print the warning for running outside of specification and prompt user to accept the terms.

Parameters
autoRespond – Response to automatically provide for all prompts

rocm_smi.**confirmOutOfSpecWarning**(*autoRespond*)

Print the warning for running outside of specification and prompt user to accept the terms.

Parameters
autoRespond – Response to automatically provide for all prompts

rocm_smi.**doesDeviceExist**(*device*)

Check whether the specified device exists

Parameters
device – DRM device identifier

rocm_smi.**driverInitialized**()

Returns true if amdgpu is found in the list of initialized modules

rocm_smi.**findFirstAvailableTemp**(*device*)

Discovers the first available device temperature to display

Returns a tuple of (temp_type, temp_value) for the device specified

Parameters
device – DRM device identifier

rocm_smi.**formatCsv**(*deviceList*)

Print out the JSON_DATA in CSV format

rocm_smi.**formatJson**(*device*, *log*)

Print out in JSON format

Parameters

- **device** – DRM device identifier
- **log** – String to parse and output into JSON format

rocm_smi.**formatMatrixToJSON**(*deviceList*, *matrix*, *metricName*)

Format symmetric matrix of GPU permutations to become JSON print-ready.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **metricName** – Title of the item to print to the log
- **matrix** – symmetric matrix full of values of every permutation of DRM devices.

Matrix example:

$$\begin{bmatrix} & GPU0 & GPU1 \\ GPU0 & 0 & 40 \\ GPU1 & 40 & 0 \end{bmatrix}$$

Where matrix content is: [[0, 40], [40, 0]]

rocm_smi.**getAllocatedMemoryPercent**(*device*)

Return dictionary of allocated memory (VRAM) of a given device
Response of allocated_memory_vram dictionary:

```
{
    'value': float allocated vram memory (floor of %) or 'N/A' (for rsmi_status_
↪t.RSMI_STATUS_NOT_SUPPORTED),
    'unit': %,
    'combined': string (eg. '30%') or 'N/A' (for rsmi_status_t.RSMI_STATUS_NOT_
```

(continues on next page)

(continued from previous page)

```
↪SUPPORTED)
    'ret': rsmi_status_t.RSMI_STATUS_NOT_SUPPORTED or rsmi_status_t.RSMI_STATUS_
↪NOT_SUPPORTED
}
```

Parameters
device – DRM device identifier

rocm_smi.**getBus**(*device*, *silent=False*)

Return the bus identifier of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getCoarseGrainUtil**(*device*, *typeName=None*)

Find Coarse Grain Utilization
If typeName is not given, will return array with of all available sensors, where sensor type and value could be addressed like this:

```
for ut_counter in utilization_counters:
    printLog(device, utilization_counter_name[ut_counter.type], ut_
↪counter.val)
```

Parameters

- **device** – DRM device identifier
- **typeName** – 'GFX Activity', 'Memory Activity'

rocm_smi.**getComputePartition**(*device*, *silent=True*)

Return the current compute partition of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is on.

rocm_smi.**getDRMDeviceId**(*device*, *silent=False*)

Return the hexadecimal value of a device's ID

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getDeviceName**(*device*, *silent=False*)

Return the uint64 value of device's name
reported by KFD

Parameters

- **device** – DRM device identifier

- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getFanSpeed**(*device*, *silent=True*)

Return a tuple with the fan speed (value,%) for a specified device, or (None,None) if either current fan speed or max fan speed cannot be obtained

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is on.

rocm_smi.**getGUID**(*device*, *silent=False*)

Return the uint64 value of device's GUID, also referred as GPU ID - reported by KFD.

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getGpuUse**(*device*, *silent=False*)

Return the current GPU usage as a percentage

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getGraphColor**(*percentage*)

rocm_smi.**getMaxPower**(*device*, *silent=False*)

Return the maximum power cap of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getMemInfo**(*device*, *memType*, *silent=False*)

Returns a tuple of (memory_used, memory_total) of
the requested memory type usage for the device specified

Parameters

- **device** – DRM device identifier
- **type** – [vram|vis_vram|gtt] Memory type to return
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off, which exposes any issue accessing the different memory types.

rocm_smi.**getMemoryPartition**(*device*, *silent=True*)

Return the current memory partition of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is on.

rocm_smi.**getMemoryPartitionCapabilities**(*device*, *silent=True*)

Return the current memory partition capablities of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is on.

rocm_smi.**getNodeId**(*device*, *silent=False*)

Return the uint32 value of device's node id reported by KFD.

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getPartitionId**(*device*, *silent=False*)

Return the partition identifier of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getPerfLevel**(*device*, *silent=False*)

Return the current performance level of a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getPid**(*name*)

Get the process id of a specific application

Parameters

name – Process name of a program to be parsed

rocm_smi.**getPidList**()

Return a list of KFD process IDs

rocm_smi.**getPower**(*device*)

Return dictionary of power responses.

Response power dictionary:

```
{
    'power': string wattage response or 'N/A' (for not RSMI_STATUS_SUCCESS),
    'power_type': power type string - 'Current Socket' or 'Average',
    'unit': W (Watt)
    'ret': response of rsmi_dev_power_get(device, byref(power), byref(power_
↪type))
}
```

Parameters

device – DRM device identifier

rocm_smi.**getPowerLabel**(*deviceList*)

Discovers the the first identified power label

Returns a string label value

Parameters
device – DRM device identifier

rocm_smi.**getProcessName**(*pid*)

Get the process name of a specific pid

Parameters
pid – Process ID of a program to be parsed

rocm_smi.**getRasEnablement**(*device*, *block*, *silent=True*)

Return RAS enablement state for a given device

Parameters

- **device** – DRM device identifier
- **block** – RAS block identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is on.

rocm_smi.**getRev**(*device*, *silent=False*)

Return the hexadecimal value of a device's Revision

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getSubsystemId**(*device*, *silent=False*)

Return the a device's subsystem id

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getTargetGfxVersion**(*device*, *silent=False*)

Return the uint64 value of device's target graphics version as reported by KFD

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getTemp**(*device*, *sensor*, *silent=True*)

Display the current temperature from a given device's sensor

Parameters

- **device** – DRM device identifier
- **sensor** – Temperature sensor identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is on.

rocm_smi.**getTemperatureLabel**(*deviceList*)

Discovers the the first identified power label
Returns a string label value

Parameters
device – DRM device identifier

rocm_smi.**getVbiosVersion**(*device*, *silent=False*)
Returns the VBIOS version for a given device

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getVendor**(*device*, *silent=False*)
Return the a device's vendor id

Parameters

- **device** – DRM device identifier
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**getVersion**(*deviceList*, *component*, *silent=False*)
Return the software version for the specified component

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **component** – Component (currently only driver)
- **silent** – Turn on to silence error output (you plan to handle manually). Default is off.

rocm_smi.**initializeRsmi**()
initializes rocmsmi if the amdgpu driver is initialized

rocm_smi.**isAmdDevice**(*device*)
Return whether the specified device is an AMD device or not

Parameters
device – DRM device identifier

rocm_smi.**isConciseInfoRequested**(*args*)

rocm_smi.**isRasControlAvailable**(*device*)
Check if RAS control is available for a specified device.

Parameters
device – DRM device identifier

rocm_smi.**listDevices**()
Returns a list of GPU devices

rocm_smi.**load**(*savefilepath*, *autoRespond*)
Load clock frequencies and fan speeds from a specified file.

Parameters

- **savefilepath** – Path to the save file

- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**padHexValue**(*value*, *length*)

Pad a hexadecimal value with a given length of zeros

Parameters

- **value** – A hexadecimal value to be padded with zeros
- **length** – Number of zeros to pad the hexadecimal value

rocm_smi.**print2DArray**(*dataArray*)

Print 2D Array with uniform spacing

rocm_smi.**printEmptyLine**()

Print out a single empty line

rocm_smi.**printErrLog**(*device*, *err*, *is_warning=False*)

Print out an error to the SMI log

Parameters

- **device** – DRM device identifier
- **err** – Error string to print

rocm_smi.**printEventList**(*device*, *delay*, *eventList*)

Print out notification events for a specified device

Parameters

- **device** – DRM device identifier
- **delay** – Notification delay in ms
- **eventList** – List of event type names (can be a single-item list)

rocm_smi.**printInfoLog**(*device*, *metricName*, *value*)

Print out an info line to the SMI log

Parameters

- **device** – DRM device identifier
- **metricName** – Title of the item to print to the log
- **value** – The item's value to print to the log

rocm_smi.**printListLog**(*metricName*, *valuesList*)

Print out to the SMI log for the lists

Parameters

- **metricName** – Title of the item to print to the log
- **valuesList** – The item's list of values to print to the log

rocm_smi.**printLog**(*device*, *metricName*, *value=None*, *extraSpace=False*, *useItalics=False*, *xcp=None*)

Print out to the SMI log

Parameters

- **device** – DRM device identifier
- **metricName** – Title of the item to print to the log
- **value** – The item's value to print to the log

rocm_smi.**printLogSpacer**(*displayString=None*, *fill='='*, *contentSizeToFit=0*)

Prints [name of the option]/[name of the program] in the spacer to explain data below

If no parameters are given, a default fill of the '=' string is used in the spacer

Parameters

- **displayString** – name of item to be displayed inside of the log spacer
- **fill** – padding string which surrounds the given display string
- **contentSizeToFit** – providing an integer > 0 allows ability to dynamically change output padding/fill based on this value instead of appWidth. Handy for concise info output.

rocm_smi.**printSysLog**(*SysComponentName*, *value*)

Print out to the SMI log for repeated features

Parameters

- **SysComponentName** – Title of the item to print to the log
- **value** – The item's value to print to the log

rocm_smi.**printTableLog**(*column_headers*, *data_matrix*, *device=None*, *tableName=None*, *anchor='>'*, *v_delim=' '*)

Print out to the SMI log for the lists

Parameters

- **column_headers** – Header names for each column
- **data_matrix** – Matrix of values
- **device** – DRM device identifier
- **tableName** – Title of the table to print to the log
- **anchor** – Alignment direction of the print output
- **v_delim** – Boundary String delimiter for the print output

rocm_smi.**printTableRow**(*space*, *displayString*, *v_delim=' '*)

Print out a line of a matrix table

Parameters

- **space** – The item's spacing to print
- **displayString** – The item's value to print
- **v_delim** – Boundary String delimiter for the print output

rocm_smi.**printTempGraph**(*deviceList*, *delay*, *temp_type*)

rocm_smi.**profileString**(*profile*)

rocm_smi.**progressbar**(*it*, *prefix=''*, *size=60*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>*)

rocm_smi.**relaunchAsSudo**()

Relaunch the SMI as sudo

To use rocm_smi_lib functions that write to sysfs, the SMI requires root access Use execvp to relaunch the script with sudo privileges

rocm_smi.**resetClocks**(*deviceList*)

Reset clocks to default

Reset clocks to default values by setting performance level to auto, as well as setting OverDrive back to 0

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**resetFans**(*deviceList*)

Reset fans to driver control for a list of devices.

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**resetGpu**(*device*)

Perform a GPU reset on the specified device

Parameters
device – DRM device identifier

rocm_smi.**resetPerfDeterminism**(*deviceList*)

Reset Performance Determinism

Parameters
deviceList – Disable Performance Determinism for these devices

rocm_smi.**resetPowerOverDrive**(*deviceList*, *autoRespond*)

Reset Power OverDrive to the default value

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**resetProfile**(*deviceList*)

Reset profile for a list of a devices.

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**resetXgmiErr**(*deviceList*)

Reset the XGMI Error value

Parameters
deviceList – Reset XGMI error count for these devices

rocm_smi.**rsmi_ret_ok**(*my_ret*, *device=None*, *metric=None*, *silent=False*)

Returns true if RSMI call status is 0 (success)

If status is not 0, error logs are written to the debug log and false is returned

Parameters

- **device** – DRM device identifier
- **my_ret** – Return of RSMI call (rocm_smi_lib API)
- **metric** – Parameter of GPU currently being analyzed
- **silent** – Echo verbose error response. True silences err output, False does not silence err output (default).

rocm_smi.**save**(*deviceList*, *savefilepath*)

Save clock frequencies and fan speeds for a list of devices to a specified file path.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **savefilepath** – Path to use to create the save file

rocm_smi.**setClockExtremum**(*deviceList*, *level*, *clkType*, *clkValue*, *autoRespond*)

Set the range for the specified clktype in the PowerPlay table for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **level** – [min|max] Minimum value or Maximum value
- **clktype** – [sclk|mclk] Which clock type to apply the range to
- **clkValue** – clock value to apply to the level
- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**setClockOverDrive**(*deviceList*, *clktype*, *value*, *autoRespond*)

Set clock speed to OverDrive for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **type** – [sclk|mclk] Clock type to set
- **value** – [0-20] OverDrive percentage
- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**setClockRange**(*deviceList*, *clkType*, *minvalue*, *maxvalue*, *autoRespond*)

Set the range for the specified clktype in the PowerPlay table for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **clktype** – [sclk|mclk] Which clock type to apply the range to
- **minvalue** – Minimum value to apply to the clock range
- **maxvalue** – Maximum value to apply to the clock range
- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**setClocks**(*deviceList*, *clktype*, *clk*)

Set clock frequency levels for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **clktype** – [validClockNames] Clock type to set
- **clk** – Clock frequency level to set

rocm_smi.**setComputePartition**(*deviceList*, *computePartitionType*)

Sets compute partitioning for a list of device

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **computePartition** – Compute Partition type to set as

rocm_smi.**setFanSpeed**(*deviceList*, *fan*)

Set fan speed for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **level** – [0-255] Fan speed level

rocm_smi.**setMemoryPartition**(*deviceList*, *memoryPartition*, *autoRespond*)

Sets memory partition (memory partition) for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **memoryPartition** – Memory Partition type to set as

rocm_smi.**setPerfDeterminism**(*deviceList*, *clkvalue*)

Set clock frequency level for a list of devices to enable performance determinism.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **value** – Clock frequency level to set

rocm_smi.**setPerformanceLevel**(*deviceList*, *level*)

Set the Performance Level for a specified device.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **level** – Performance Level to set

rocm_smi.**setPowerOverDrive**(*deviceList*, *value*, *autoRespond*)

Use Power OverDrive to change the the maximum power available power available to the GPU in Watts. May be limited by the maximum power the VBIOS is configured to allow this card to use in OverDrive mode.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **value** – New maximum power to assign to the target device, in Watts
- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**setPowerPlayTableLevel**(*deviceList*, *clkType*, *point*, *clk*, *volt*, *autoRespond*)

Set clock frequency and voltage for a level in the PowerPlay table for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **clktype** – [sclk|mclk] Which clock type to apply the range to
- **point** – Point on the voltage curve to modify
- **clk** – Clock speed specified for this curve point
- **volt** – Voltage specified for this curve point
- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**setProfile**(*deviceList*, *profile*)

Set Power Profile, or set CUSTOM Power Profile values for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **profile** – Profile to set

rocm_smi.**setRas**(*deviceList*, *rasAction*, *rasBlock*, *rasType*)

Perform a RAS action on the devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **rasAction** – [enable|disable|inject] RAS Action to perform
- **rasBlock** – [$validRasBlocks] RAS block
- **rasType** – [ce|ue] Error type to enable/disable

rocm_smi.**setVoltageCurve**(*deviceList*, *point*, *clk*, *volt*, *autoRespond*)

Set voltage curve for a point in the PowerPlay table for a list of devices.

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **point** – Point on the voltage curve to modify
- **clk** – Clock speed specified for this curve point
- **volt** – Voltage specified for this curve point
- **autoRespond** – Response to automatically provide for all prompts

rocm_smi.**showAccessibleTopology**(*deviceList*)

Display the HW Topology Information based on link accessibility

This reads the HW Topology file and displays the matrix for the nodes

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showAllConcise**(*deviceList*)

Display critical info for all devices in a concise format

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showAllConciseHw**(*deviceList*)

Display critical Hardware info

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showBus**(*deviceList*)

Display PCI Bus info

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showClocks**(*deviceList*)

Display all available clocks for a list of devices

Current clocks marked with a '*' symbol

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showComputePartition**(*deviceList*)

Returns the current compute partitioning for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showCurrentClocks**(*deviceList*, *clk_defined=None*, *concise=False*)

Display all clocks for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **clk-type** – Clock type to display

rocm_smi.**showCurrentFans**(*deviceList*)

Display the current fan speed for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showCurrentTemps**(*deviceList*)

Display all available temperatures for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showDriverVersion**(*deviceList*, *component*)

Display the software version for the specified component

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **component** – Component (currently only driver)

rocm_smi.**showEnergy**(*deviceList*)

Display amount of energy consumed by device until now

Default counter value is 10000b, indicating energy status unit is 15.3 micro-Joules increment. :param deviceList: List of DRM devices (can be a single-item list)

rocm_smi.**showEvents**(*deviceList*, *eventTypes*)

Display a blocking list of events for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **eventTypes** – List of event type names (can be a single-item list)

rocm_smi.**showFwInfo**(*deviceList*, *fwType*)

Show the requested FW information for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **fwType** – [$validFwBlocks] FW block version to display (all if left empty)

rocm_smi.**showGPUMetrics**(*deviceList*)

Returns the gpu metrics for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showGpuUse**(*deviceList*)

Display GPU use for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showGpusByPid**(*pidList*)

Show GPUs used by a specific Process ID (pid)

Print out the GPU(s) used by a specific KFD process If pidList is empty, print all used GPUs for all KFD processes

Parameters
pidList – List of PIDs to check

rocm_smi.**showHopsTopology**(*deviceList*)

Display the HW Topology Information based on number of hops

This reads the HW Topology file and displays the matrix for the nodes

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showHwTopology**(*deviceList*)

Display the HW Topology Information based on weight/hops/type

This reads the HW Topology file and displays the matrix for the nodes

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showId**(*deviceList*)

Display the device IDs for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showMaxPower**(*deviceList*)

Display the maximum Graphics Package Power that this GPU will attempt to consume before it begins throttling performance

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showMemInfo**(*deviceList*, *memType*)

Display Memory information for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **memType** – [$validMemTypes] Type of memory information to display

rocm_smi.**showMemUse**(*deviceList*)

Display GPU memory usage for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showMemVendor**(*deviceList*)

Display GPU memory vendor for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showMemoryPartition**(*deviceList*)

Returns the current memory partition for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showNodesBw**(*deviceList*)

Display max and min bandwidth between nodes. Currently supports XGMI only. This reads the HW Topology file and displays the matrix for the nodes :param deviceList: List of DRM devices (can be a single-item list)

rocm_smi.**showNumaTopology**(*deviceList*)

Display the HW Topology Information for numa nodes

This reads the HW Topology file and display the matrix for the nodes

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showOverDrive**(*deviceList*, *odtype*)

Display current OverDrive level for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **odtype** – [sclk|mclk] OverDrive type

rocm_smi.**showPcieBw**(*deviceList*)

Display estimated PCIe bandwidth usage for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showPcieReplayCount**(*deviceList*)

Display number of PCIe replays for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showPerformanceLevel**(*deviceList*)

Display current Performance Level for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showPids**(*verbose*)

Show Information for PIDs created in a KFD (Compute) context

rocm_smi.**showPower**(*deviceList*)

Display Current (also known as instant) Socket or Average
Graphics Package Power Consumption for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showPowerPlayTable**(*deviceList*)
Display current GPU Memory clock frequencies and voltages for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showProduct**(*deviceList*)
Show the requested product information for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showProfile**(*deviceList*)
Display available Power Profiles for a list of devices.

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showProgressbar**(*title=''*, *timeInSeconds=13*)

rocm_smi.**showRange**(*deviceList*, *rangeType*)
Show the range for either the sclk or voltage for the specified devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **rangeType** – [sclk|voltage] Type of range to return

rocm_smi.**showRasInfo**(*deviceList*, *rasType*)
Show the requested RAS information for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **rasType** – [$validRasBlocks] RAS counter to display (all if left empty)

rocm_smi.**showRetiredPages**(*deviceList*, *retiredType='all'*)
Show retired pages of a specified type for a list of devices

Parameters

- **deviceList** – List of DRM devices (can be a single-item list)
- **retiredType** – Type of retired pages to show (default = all)

rocm_smi.**showSerialNumber**(*deviceList*)
Display the serial number for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showTempGraph**(*deviceList*)

rocm_smi.**showTypeTopology**(*deviceList*)

Display the HW Topology Information based on link type

This reads the HW Topology file and displays the matrix for the nodes

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showUId**(*deviceList*)

Display the unique device ID for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showVbiosVersion**(*deviceList*)

Display the VBIOS version for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showVersion**(*isCSV=False*)

rocm_smi.**showVoltage**(*deviceList*)

Display the current voltage (in millivolts) for a list of devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showVoltageCurve**(*deviceList*)

Show the voltage curve points for the specified devices

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showWeightTopology**(*deviceList*)

Display the HW Topology Information based on weights

This reads the HW Topology file and displays the matrix for the nodes

Parameters
deviceList – List of DRM devices (can be a single-item list)

rocm_smi.**showXgmiErr**(*deviceList*)

Display the XGMI Error status

This reads the XGMI error file, and interprets the return value from the sysfs file

Parameters
deviceList – Show XGMI error state for these devices

CHAPTER

NINE

ROCM SMI C++ API TUTORIAL

```cpp
#include <stdint.h>
#include "rocm_smi/rocm_smi.h"
int main() {

rsmi_status_t ret;
uint32_t num_devices;
uint16_t dev_id;

// We will skip return code checks for this example, but it
// is recommended to always check this as some calls may not
// apply for some devices or ROCm releases

ret = rsmi_init(0);
ret = rsmi_num_monitor_devices(&num_devices);

for (int i=0; i < num_devices; ++i) {
    ret = rsmi_dev_id_get(i, &dev_id);
    // dev_id holds the device ID of device i, upon a
    // successful call
}
ret = rsmi_shut_down();
return 0;
}
```

For more examples please check the C++ example or tests.

CHAPTER

TEN

ROCM SMI PYTHON API TUTORIAL

```
import sys
sys.path.append("/opt/rocm/libexec/rocm_smi/")
try:
    import rocm_smi
except ImportError:
    raise ImportError("Could not import /opt/rocm/libexec/rocm_smi/rocm_smi.py")

class prof_utils:
    def __init__(self, mode) -> None:
        rocm_smi.initializeRsmi()

    def getPower(self, device):
        return rocm_smi.getPower(device)

    def listDevices(self):
        return rocm_smi.listDevices()

    def getMemInfo(self, device):
        (memUsed, memTotal) = rocm_smi.getMemInfo(device, "vram")
        return round(float(memUsed)/float(memTotal) * 100, 2)
```

CHAPTER

ELEVEN

LICENSE

PYTHON MODULE INDEX

r

INDEX

R

S