
AMD SMI
Release 24.7.1.0

Advanced Micro Devices, Inc.

Jun 19, 2025

INSTALL

| | | |
|----------|--|------------|
| 1 | AMD SMI library and CLI tool | 3 |
| 1.1 | Requirements | 3 |
| 1.2 | Install amdgpu driver and AMD SMI with ROCm | 3 |
| 1.3 | Install AMD SMI without ROCm | 4 |
| 1.4 | Optionally enable CLI autocompletion | 4 |
| 1.5 | Install the Python library for multiple ROCm instances | 4 |
| 2 | Building AMD SMI | 7 |
| 2.1 | Required software | 7 |
| 2.2 | Build steps | 7 |
| 2.3 | Rebuild the Python wrapper | 8 |
| 2.4 | Build the tests | 8 |
| 2.5 | Build the docs | 8 |
| 3 | AMD SMI C++ library usage and examples | 9 |
| 3.1 | Device and socket handles | 9 |
| 3.2 | Hello AMD SMI | 9 |
| 4 | AMD SMI Python interface overview | 13 |
| 4.1 | Get started | 13 |
| 4.2 | Usage | 13 |
| 4.3 | Exceptions | 14 |
| 5 | AMD SMI CLI tool usage | 15 |
| 5.1 | Install the CLI Tool and Python library | 15 |
| 5.2 | Get started | 15 |
| 5.3 | Commands | 16 |
| 6 | AMD SMI C++ API reference | 33 |
| 6.1 | Modules | 33 |
| 6.2 | File List | 33 |
| 6.3 | Globals | 33 |
| 6.4 | Data Structures | 33 |
| 6.5 | Data Fields | 33 |
| 7 | AMD SMI Python API reference | 35 |
| 7.1 | API | 35 |
| 7.2 | CPU APIs | 103 |
| 8 | Changelog for AMD SMI Library | 119 |
| 8.1 | amd_smi_lib for ROCm 6.3.1 | 119 |

| | | |
|----------|----------------------------|------------|
| 8.2 | amd_smi_lib for ROCm 6.3.0 | 120 |
| 8.3 | amd_smi_lib for ROCm 6.2.1 | 136 |
| 8.4 | amd_smi_lib for ROCm 6.2.0 | 136 |
| 8.5 | amd_smi_lib for ROCm 6.1.2 | 140 |
| 8.6 | amd_smi_lib for ROCm 6.1.1 | 143 |
| 8.7 | amd_smi_lib for ROCm 6.1.0 | 148 |
| 8.8 | amd_smi_lib for ROCm 6.0.0 | 158 |
| 9 | License | 159 |

The AMD System Management Interface (AMD SMI) library offers a unified tool for managing and monitoring GPUs, particularly in high-performance computing environments. It provides a user-space interface that allows applications to control GPU operations, monitor performance, and retrieve information about the system's drivers and GPUs.

Find the source code at <https://github.com/ROCm/amdsmi>.

Note

AMD SMI is the successor to https://github.com/ROCm/rocm_smi_lib.

Install

- *Library and CLI tool installation*
- *Build from source*

How to

- *C++ library usage*
- *Python library usage*
- *CLI tool usage*

Reference

- *C++ API*
 - *Modules*
 - *Files*
 - *Globals*
 - *Data structures*
 - *Data fields*
- *Python API*

Tutorials

- *AMD SMI examples (GitHub)*
- *ROCm SMI examples (GitHub)*

To contribute to the documentation, refer to [Contributing to ROCm](#).

Find ROCm licensing information on the [Licensing](#) page.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Copyright © 2014-2024 Advanced Micro Devices, Inc. All rights reserved.

AMD SMI LIBRARY AND CLI TOOL

This section describes how to install the AMD SMI library, Python interface, and command line tool either as part of the [ROCm software stack](#) – or manually.

1.1 Requirements

The following are required to install and use the AMD SMI libraries and CLI tool.

- Python 3.6.8+ (64-bit)
- amdgpu driver must be loaded for `amdsmi_init()` to work.

1.1.1 Supported platforms

At initial release, the AMD SMI library will support Linux bare metal and Linux virtual machine guest for AMD GPUs. In a future release, the library will be extended to support AMD EPYC™ CPUs.

AMD SMI library can run on AMD ROCm supported platforms, refer to [System requirements \(Linux\)](#) for more information.

To run the AMD SMI library, the amdgpu driver and the amd_hsmpt driver need to be installed. Optionally, libdrm can be installed to query firmware information and hardware IPs.

1.2 Install amdgpu driver and AMD SMI with ROCm

1. Get the amdgpu-install installer following the instructions for your Linux distribution at [rocm-install-on-linux:install/amdgpu-install](#).

See the following example; your desired ROCm release and install URL may be different.

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/6.2.2/ubuntu/noble/amdgpu-install_6.2.
↳60202-1_all.deb
sudo apt install ./amdgpu-install_6.2.60202-1_all.deb
```

2. Use amdgpu-install to install the amdgpu driver and ROCm packages with AMD SMI included.

```
sudo amdgpu-install --usecase=rocm
```

The `amdgpu-install --usecase=rocm` option triggers both an amdgpu driver update and AMD SMI packages to be installed on your device.

3. Verify your installation.

```
amd-smi --help
```

1.3 Install AMD SMI without ROCm

The following are example steps to install the AMD SMI libraries and CLI tool on Ubuntu 22.04.

1. Install the library.

```
sudo apt install amd-smi-lib
```

2. Add the installation directory to your PATH. If installed with ROCm, ignore this step.

```
export PATH="${PATH:+${PATH}:}~/opt/rocm/bin"
```

3. Verify your installation.

```
amd-smi --help
```

1.4 Optionally enable CLI autocompletion

The `amd-smi` CLI application supports autocompletion. If `argcomplete` is not installed and enabled already, do so using the following commands.

```
python3 -m pip install argcomplete
activate-global-python-argcomplete --user
# restart shell to enable
```

1.5 Install the Python library for multiple ROCm instances

If multiple ROCm versions are installed and you are not using `pyenv`, uninstall previous versions of AMD SMI before installing the desired version from your ROCm instance.

1.5.1 Manually install the Python library

The following are example AMD SMI installation steps on Ubuntu 22.04 without ROCm.

1. Remove previous AMD SMI installation.

```
python3 -m pip list | grep amd
python3 -m pip uninstall amdsmi
```

2. Install the AMD SMI Python library from your target ROCm instance.

```
apt install amd-smi-lib
cd /opt/rocm/share/amd_smi
python3 -m pip install --upgrade pip
python3 -m pip install --user .
```

3. You should now have the AMD SMI Python library in your Python path:

```
~$ python3
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import amdsmi
>>>
```


BUILDING AMD SMI

This section describes the prerequisites and steps to build AMD SMI from source.

2.1 Required software

To build the AMD SMI library, the following components are required. Note that the software versions specified were used during development; earlier versions are not guaranteed to work.

- CMake (v3.14.0 or later) – `python3 -m pip install cmake`
- g++ (v5.4.0 or later)

In order to build the AMD SMI Python package, the following components are required:

- Python (3.6.8 or later)
 - prerequisite modules:
 - * `python3-wheel`
 - * `python3-setuptools`
- `virtualenv - python3 -m pip install virtualenv`

2.2 Build steps

1. Clone the AMD SMI repository to your local Linux machine.

```
git clone https://github.com/ROCm/amdsmi.git
```

2. The default installation location for the library and headers is `/opt/rocm`. Before installation, any old ROCm directories should be deleted:
 - `/opt/rocm`
 - `/opt/rocm-<version_number>`
3. Build the library by following the typical CMake build sequence (run as root user or use `sudo` before `make install` command); for instance:

```
mkdir -p build
cd build
cmake ..
make -j $(nproc)
make install
```

The built library is located in the `build/` directory. To build the rpm and deb packages use the following command:

```
make package
```

2.3 Rebuild the Python wrapper

The Python wrapper for the AMD SMI library is found in the *auto-generated file* `py-interface/amdsmi_wrapper.py`. It is essential to regenerate this wrapper whenever there are changes to the C++ API. It is not regenerated automatically.

To regenerate the wrapper, use the following command.

```
./update_wrapper.sh
```

After this command, the file in `py-interface/amdsmi_wrapper.py` will be updated on compile.

Note

You need Docker installed on your system to regenerate the Python wrapper.

2.4 Build the tests

To verify the build and capabilities of AMD SMI on your system, as well as to see practical examples of its usage, you can build and run the available [tests in the repository](#). Follow these steps to build the tests:

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

2.4.1 Run the tests

Once the tests are *built*, you can run them by executing the `amdsmitst` program. The executable can be found at `build/tests/amd_smi_test/`.

2.5 Build the docs

To build the documentation, follow the instructions at [Building documentation](#).

AMD SMI C++ LIBRARY USAGE AND EXAMPLES

This section presents a brief overview and some basic examples on the AMD SMI library's usage. Whether you are developing applications for performance monitoring, system diagnostics, or resource allocation, the AMD SMI C++ library serves as a valuable tool for leveraging the full potential of AMD hardware in your projects.

Note

`hipcc` and other compilers will not automatically link in the `libamd_smi` dynamic library. To compile code that uses the AMD SMI library API, ensure the `libamd_smi.so` can be located by setting the `LD_LIBRARY_PATH` environment variable to the directory containing `librocm_smi64.so` (usually `/opt/rocm/lib`) or by passing the `-lamd_smi` flag to the compiler.

See also

Refer to the *C++ library API reference*.

3.1 Device and socket handles

Many functions in the library take a *socket handle* or *device handle*. A *socket* refers to a physical hardware socket, abstracted by the library to represent the hardware more effectively to the user. While there is always one unique GPU per socket, an APU may house both a GPU and CPU on the same socket. For MI200 GPUs, multiple GCDs may reside within a single socket

To identify the sockets in a system, use the `amdsmi_get_socket_handles()` function, which returns a list of socket handles. These handles can then be used with `amdsmi_get_processor_handles()` to query devices within each socket. The device handle is used to differentiate between detected devices; however, it's important to note that a device handle may change after restarting the application, so it should not be considered a persistent identifier across processes.

The list of socket handles obtained from `amdsmi_get_socket_handles()` can also be used to query the CPUs in each socket by calling `amdsmi_get_processor_handles_by_type()`. This function can then be called again to query the cores within each CPU.

3.2 Hello AMD SMI

An application using AMD SMI must call `amdsmi_init()` to initialize the AMD SMI library before all other calls. This call initializes the internal data structures required for subsequent AMD SMI operations. In the call, a flag can be passed to indicate if the application is interested in a specific device type.

amdsmi_shut_down() must be the last call to properly close connection to driver and make sure that any resources held by AMD SMI are released.

1. A simple “Hello World” type program that displays the temperature of detected devices looks like this:

```
#include <iostream>
#include <vector>
#include "amd_smi/amdsmi.h"

int main() {
    amdsmi_status_t ret;

    // Init amdsmi for sockets and devices. Here we are only interested in AMD_GPUS.
    ret = amdsmi_init(AMDSMI_INIT_AMD_GPUS);

    // Get all sockets
    uint32_t socket_count = 0;

    // Get the socket count available in the system.
    ret = amdsmi_get_socket_handles(&socket_count, nullptr);

    // Allocate the memory for the sockets
    std::vector<amdsmi_socket_handle> sockets(socket_count);
    // Get the socket handles in the system
    ret = amdsmi_get_socket_handles(socket_count, &sockets[0]);

    std::cout << "Total Socket: " << socket_count << std::endl;

    // For each socket, get identifier and devices
    for (uint32_t i=0; i < socket_count; i++) {
        // Get Socket info
        char socket_info[128];
        ret = amdsmi_get_socket_info(sockets[i], 128, socket_info);
        std::cout << "Socket " << socket_info << std::endl;

        // Get the device count for the socket.
        uint32_t device_count = 0;
        ret = amdsmi_get_processor_handles(sockets[i], &device_count, nullptr);

        // Allocate the memory for the device handlers on the socket
        std::vector<amdsmi_processor_handle> processor_handles(device_count);
        // Get all devices of the socket
        ret = amdsmi_get_processor_handles(sockets[i],
            &device_count, &processor_handles[0]);

        // For each device of the socket, get name and temperature.
        for (uint32_t j=0; j < device_count; j++) {
            // Get device type. Since the amdsmi is initialized with
            // AMD_SMI_INIT_AMD_GPUS, the processor_type must be AMDSMI_PROCESSOR_TYPE_
            ↪AMD_GPU.
            processor_type_t processor_type;
            ret = amdsmi_get_processor_type(processor_handles[j], &processor_type);
            if (processor_type != AMDSMI_PROCESSOR_TYPE_AMD_GPU) {
                std::cout << "Expect AMDSMI_PROCESSOR_TYPE_AMD_GPU device type!\n";
            }
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

    return 1;
}

// Get device name
amdsmi_board_info_t board_info;
ret = amdsmi_get_gpu_board_info(processor_handles[j], &board_info);
std::cout << "\tdevice "
          << j << "\n\t\tName:" << board_info.product_name << std::endl;

// Get temperature
int64_t val_i64 = 0;
ret = amdsmi_get_temp_metric(processor_handles[j], AMDSMI_TEMPERATURE_TYPE_
↳EDGE,
                          AMDSMI_TEMP_CURRENT, &val_i64);
std::cout << "\t\tTemperature: " << val_i64 << "C" << std::endl;
}
}

// Clean up resources allocated at amdsmi_init. It will invalidate sockets
// and devices pointers
ret = amdsmi_shut_down();

return 0;
}

```

2. A sample program that displays the power of detected CPUs looks like this:

```

#include <iostream>
#include <vector>
#include "amd_smi/amdsmi.h"

int main(int argc, char **argv) {
    amdsmi_status_t ret;
    uint32_t socket_count = 0;

    // Initialize amdsmi for AMD CPUs
    ret = amdsmi_init(AMDSMI_INIT_AMD_CPUS);

    ret = amdsmi_get_socket_handles(&socket_count, nullptr);

    // Allocate the memory for the sockets
    std::vector<amdsmi_socket_handle> sockets(socket_count);

    // Get the sockets of the system
    ret = amdsmi_get_socket_handles(&socket_count, &sockets[0]);

    std::cout << "Total Socket: " << socket_count << std::endl;

    // For each socket, get cpus
    for (uint32_t i = 0; i < socket_count; i++) {
        uint32_t cpu_count = 0;

```

(continues on next page)

(continued from previous page)

```
// Set processor type as AMDSMI_PROCESSOR_TYPE_AMD_CPU
processor_type_t processor_type = AMDSMI_PROCESSOR_TYPE_AMD_CPU;
ret = amdsmi_get_processor_handles_by_type(sockets[i], processor_type, &
↪ nullptr, &cpu_count);

// Allocate the memory for the cpus
std::vector<amdsmi_processor_handle> plist(cpu_count);

// Get the cpus for each socket
ret = amdsmi_get_processor_handles_by_type(sockets[i], processor_type, &
↪ plist[0], &cpu_count);

for (uint32_t index = 0; index < plist.size(); index++) {
    uint32_t socket_power;
    std::cout<<"CPU "<<index<<"\t"<< std::endl;
    std::cout<<"Power (Watts): ";

    ret = amdsmi_get_cpu_socket_power(plist[index], &socket_power);
    if(ret != AMDSMI_STATUS_SUCCESS)
        std::cout<<"Failed to get cpu socket power"<<"["<<index<<"] , Err["<
↪ <ret<<"] "<< std::endl;

    if (!ret) {
        std::cout<<static_cast<double>(socket_power)/1000<<std::endl;
    }
    std::cout<<std::endl;
}

// Clean up resources allocated at amdsmi_init
ret = amdsmi_shut_down();

return 0;
}
```

AMD SMI PYTHON INTERFACE OVERVIEW

The AMD SMI Python interface provides a convenient way to interact with AMD hardware through a simple and accessible *API*. Compatible with Python 3.6 and higher, this library requires the AMD driver to be loaded for initialization – review the *prerequisites*.

4.1 Get started

To get started, the `amdsmi` folder should be copied and placed next to the importing script. Import it as follows:

```
from amdsmi import *

try:
    amdsmi_init()

    # amdsmi calls ...

except AmdSmiException as e:
    print(e)
finally:
    try:
        amdsmi_shut_down()
    except AmdSmiException as e:
        print(e)
```

4.1.1 Folder structure

| File name | Description |
|----------------------------------|--------------------------------------|
| <code>__init__.py</code> | Python package initialization file |
| <code>amdsmi_interface.py</code> | Amdsmi library Python interface |
| <code>amdsmi_wrapper.py</code> | Python wrapper around amdsmi binary |
| <code>amdsmi_exception.py</code> | Amdsmi <i>exceptions</i> Python file |

4.2 Usage

 Note

hipcc and other compilers will not automatically link in the libamd_smi dynamic library. To compile code that uses the AMD SMI library API, ensure the libamd_smi.so can be located by setting the LD_LIBRARY_PATH environment variable to the directory containing librocm_smi64.so (usually /opt/rocm/lib) or by passing the -lamd_smi flag to the compiler.

➔ See also

Refer to the *Python library API reference*.

An application using AMD SMI must call `amdsmi_init()` to initialize the AMD SMI library before all other calls. This call initializes the internal data structures required for subsequent AMD SMI operations. In the call, a flag can be passed to indicate if the application is interested in a specific device type.

`amdsmi_shut_down()` must be the last call to properly close connection to driver and make sure that any resources held by AMD SMI are released.

4.3 Exceptions

All exceptions are in `amdsmi_exception.py` file.

Exceptions that can be thrown by AMD SMI are:

- `AmdSmiException`: base `amdsmi` exception class
- `AmdSmiLibraryException`: derives base `AmdSmiException` class and represents errors that can occur in `amdsmi-lib`. When this exception is thrown, `err_code` and `err_info` are set. `err_code` is an integer that corresponds to errors that can occur in `amdsmi-lib` and `err_info` is a string that explains the error that occurred.

For example:

```
try:
    num_of_GPUs = len(amdsmi_get_processor_handles())
    if num_of_GPUs == 0:
        print("No GPUs on machine")
except AmdSmiException as e:
    print("Error code: {}".format(e.err_code))
    if e.err_code == amdsmi_wrapper.AMDSMI_STATUS_RETRY:
        print("Error info: {}".format(e.err_info))
```

- `AmdSmiRetryException`: Derives `AmdSmiLibraryException` class and signals device is busy and call should be retried.
- `AmdSmiTimeoutException`: Derives `AmdSmiLibraryException` class and represents that call had timed out.
- `AmdSmiParameterException`: Derives base `AmdSmiException` class and represents errors related to invalid parameters passed to functions. When this exception is thrown, `err_msg` is set and it explains what is the actual and expected type of the parameters.
- `AmdSmiBdfFormatException`: Derives base `AmdSmiException` class and represents invalid bdf format.

AMD SMI CLI TOOL USAGE

This tool is a command line interface (CLI) for manipulating and monitoring the `amdgpu` kernel; it is intended to replace and deprecate the existing `rocm_smi` CLI tool and `gpubv-smi` tool. The AMD SMI CLI tool uses Ctypes to call the `amd_smi_lib` API.

When using the CLI tool, you should have at least one AMD GPU and the driver installed.

Disclaimer

The AMD SMI CLI tool is provided as an example code to aid the development of telemetry tools. The *Python* or *C++ library* is recommended as a robust data source.

5.1 Install the CLI Tool and Python library

Refer to the *installation instructions*.

5.2 Get started

The `amd-smi` command provides system management and monitoring capabilities for AMD hardware. When run without arguments, it reports the version and platform detected:

```
~$ amd-smi
usage: amd-smi [-h] ...

AMD System Management Interface | Version: 24.7.1.0 | ROCm version: 6.3.0 | Platform: Linux Baremetal

options:
  -h, --help            show this help message and exit

AMD-SMI Commands:
  version              Descriptions:
                       Display version information
  list                 List GPU information
  static               Gets static information about the specified GPU
  firmware (ucode)    Gets firmware information about the specified GPU
  bad-pages            Gets bad page information about the specified GPU
  metric               Gets metric/performance information about the specified GPU
  process              Lists general process information running on the specified GPU
  event                Displays event information for the given GPU
```

(continues on next page)

(continued from previous page)

| | |
|----------------|--|
| topology | Displays topology information of the devices |
| set | Set options for devices |
| reset | Reset options for devices |
| monitor (dmon) | Monitor metrics for target devices |
| xgmi | Displays xgmi information of the devices |

Example commands:

```
amd-smi static --gpu 0
amd-smi metric
amd-smi process --gpu 0 1
amd-smi reset --gpureset --gpu all
```

Note

For command-specific help, use `amd-smi [command] --help` for see more detailed usage information. See *Commands*.

For more detailed version information, use `amd-smi version`.

5.3 Commands

The following are the help output for each command, providing quick reference details for usage.

5.3.1 amd-smi list

Lists GPU information.

```
~$ amd-smi list --help
usage: amd-smi list [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                  [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]]

Lists all the devices on the system and the links between devices.
Lists the BDF, UUID, KFD_ID, and NODE_ID for each GPU and/or CPUs.
In virtualization environments, it can also list VFs associated to each
GPU with some basic information for each VF.

options:
  -h, --help                show this help message and exit
  -g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
                             ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
                             ↪7e20764e2714
                             ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
                             ↪7c8cefe1f084
                             ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
                             ↪25d815189854
                             ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                             ↪4c2be5e66537
                             all | Selects all devices
  -U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                             ID: 0
```

(continues on next page)

(continued from previous page)

```

                                ID: 1
                                ID: 2
                                ID: 3
                                all | Selects all devices
-0, --core CORE [CORE ...] Select a Core ID from the possible choices:
                                ID: 0 - 95
                                all | Selects all devices

Command Modifiers:
--json                        Displays output in JSON format (human readable by default).
--csv                         Displays output in CSV format (human readable by default).
--file FILE                   Saves output into a file on the provided path (stdout by
↪ default).
--loglevel LEVEL              Set the logging level from the possible choices:
                                DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.2 amd-smi static

Gets static information about the specified GPU. See the *sample output* for `amd-smi static`.

```

~$ amd-smi static --help
usage: amd-smi static [-h] [-g GPU [GPU ...] | -U CPU [CPU ...]] [-a] [-b] [-V] [-d] [-v]
                    [-c] [-B] [-R] [-r] [-p] [-l] [-P] [-x] [-u] [-s] [-i]
                    [--json | --csv] [--file FILE] [--loglevel LEVEL]

If no GPU is specified, returns static information for all GPUs on the system.
If no static argument is provided, all static information will be displayed.

Static Arguments:
-h, --help                show this help message and exit
-g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
↪ 7e20764e2714            ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
↪ 7c8cefe1f084            ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
↪ 25d815189854            ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
↪ 4c2be5e66537            ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                                all | Selects all devices
-U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                                ID: 0
                                ID: 1
                                ID: 2
                                ID: 3
                                all | Selects all devices
-a, --asic                All asic information
-b, --bus                 All bus information
-V, --vbios               All video bios information (if available)
-d, --driver              Displays driver version
-v, --vram                All vram information
-c, --cache               All cache information

```

(continues on next page)

(continued from previous page)

| | |
|-------------------------|---|
| -B, --board | All board information |
| -R, --process-isolation | The process isolation status |
| -r, --ras | Displays RAS features information |
| -p, --partition | Partition information |
| -l, --limit | All limit metric values (i.e. power and thermal limits) |
| -P, --policy | The available DPM policy |
| -x, --xgmi-plpd | The available XGMI per-link power down policy |
| -u, --numa | All numa node information |
| CPU Arguments: | |
| -s, --smu | All SMU FW information |
| -i, --interface-ver | Displays hsmp interface version |
| Command Modifiers: | |
| --json | Displays output in JSON format (human readable by default). |
| --csv | Displays output in CSV format (human readable by default). |
| --file FILE | Saves output into a file on the provided path (stdout by default). |
| --loglevel LEVEL | Set the logging level from the possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL |

5.3.3 amd-smi firmware

Gets firmware information about the specified GPU.

```
~$ amd-smi firmware --help
usage: amd-smi firmware [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                        [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]] [-f]

If no GPU is specified, return firmware information for all GPUs on the system.

Firmware Arguments:
  -h, --help                show this help message and exit
  -g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
                           ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
                           ↪7e20764e2714
                           ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
                           ↪7c8cefe1f084
                           ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
                           ↪25d815189854
                           ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                           ↪4c2be5e66537
                           all | Selects all devices
  -U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                           ID: 0
                           ID: 1
                           ID: 2
                           ID: 3
                           all | Selects all devices
  -O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                           ID: 0 - 95
                           all | Selects all devices
```

(continues on next page)

(continued from previous page)

```
-f, --ucode-list, --fw-list All FW list information
```

Command Modifiers:

```
--json Displays output in JSON format (human readable by default).
--csv Displays output in CSV format (human readable by default).
--file FILE Saves output into a file on the provided path (stdout by default).
--loglevel LEVEL Set the logging level from the possible choices:
                DEBUG, INFO, WARNING, ERROR, CRITICAL
```

5.3.4 amd-smi bad-pages

Gets bad page information about the specified GPU.

```
~$ amd-smi bad-pages --help
usage: amd-smi bad-pages [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                        [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]] [-p]
                        [-r] [-u]
```

If no GPU is specified, return bad page information for all GPUs on the system.

Bad Pages Arguments:

```
-h, --help show this help message and exit
-g, --gpu GPU [GPU ...] Select a GPU ID, BDF, or UUID from the possible choices:
                        ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
                        ↪7e20764e2714
                        ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
                        ↪7c8cefe1f084
                        ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
                        ↪25d815189854
                        ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                        ↪4c2be5e66537
                        all | Selects all devices
-U, --cpu CPU [CPU ...] Select a CPU ID from the possible choices:
                        ID: 0
                        ID: 1
                        ID: 2
                        ID: 3
                        all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                        ID: 0 - 95
                        all | Selects all devices
-p, --pending Displays all pending retired pages
-r, --retired Displays retired pages
-u, --un-res Displays unreservable pages
```

Command Modifiers:

```
--json Displays output in JSON format (human readable by default).
--csv Displays output in CSV format (human readable by default).
--file FILE Saves output into a file on the provided path (stdout by default).
```

(continues on next page)

(continued from previous page)

```
--loglevel LEVEL          Set the logging level from the possible choices:
                          DEBUG, INFO, WARNING, ERROR, CRITICAL
```

5.3.5 amd-smi metric

Gets metrics and performance information about the specified GPU.

```
~$ amd-smi metric --help
usage: amd-smi metric [-h] [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]]
                    [-w INTERVAL] [-W TIME] [-i ITERATIONS] [-m] [-u] [-p] [-c] [-t]
                    [-P] [-e] [-k] [-f] [-C] [-o] [-l] [-x] [-E] [--cpu-power-metrics]
                    [--cpu-prochot] [--cpu-freq-metrics] [--cpu-c0-res]
                    [--cpu-lclk-dpm-level NBIODID] [--cpu-pwr-svi-telemetry-rails]
                    [--cpu-io-bandwidth IO_BW LINKID_NAME]
                    [--cpu-xgmi-bandwidth XGMI_BW LINKID_NAME] [--cpu-metrics-ver]
                    [--cpu-metrics-table] [--cpu-socket-energy] [--cpu-ddr-bandwidth]
                    [--cpu-temp] [--cpu-dimm-temp-range-rate DIMM_ADDR]
                    [--cpu-dimm-pow-consumption DIMM_ADDR]
                    [--cpu-dimm-thermal-sensor DIMM_ADDR] [--core-boost-limit]
                    [--core-curr-active-freq-core-limit] [--core-energy]
                    [--json | --csv] [--file FILE] [--loglevel LEVEL]
```

If no GPU is specified, returns metric information for all GPUs on the system.

If no metric argument is provided, all metric information will be displayed.

Metric arguments:

| | |
|-------------------------------|---|
| -h, --help | show this help message and exit |
| -g, --gpu GPU [GPU ...] | Select a GPU ID, BDF, or UUID from the |
| ↳ possible choices: | |
| ↳ 0000-1000-808f-7e20764e2714 | ID: 0 BDF: 0000:01:00.0 UUID: 7eff74a0- |
| ↳ 0000-1000-80ae-7c8cefe1f084 | ID: 1 BDF: 0001:01:00.0 UUID: b6ff74a0- |
| ↳ 0000-1000-8071-25d815189854 | ID: 2 BDF: 0002:01:00.0 UUID: 36ff74a0- |
| ↳ 0000-1000-80c4-4c2be5e66537 | ID: 3 BDF: 0003:01:00.0 UUID: f4ff74a0- |
| | all Selects all devices |
| -U, --cpu CPU [CPU ...] | Select a CPU ID from the possible choices: |
| | ID: 0 |
| | ID: 1 |
| | ID: 2 |
| | ID: 3 |
| | all Selects all devices |
| -O, --core CORE [CORE ...] | Select a Core ID from the possible choices: |
| | ID: 0 - 95 |
| | all Selects all devices |
| -w, --watch INTERVAL | Reprint the command in a loop of INTERVAL |
| ↳ seconds | |
| -W, --watch_time TIME | The total TIME to watch the given command |
| -i, --iterations ITERATIONS | Total number of ITERATIONS to loop on the |
| ↳ given command | |

(continues on next page)

(continued from previous page)

| | |
|--|---|
| -m, --mem-usage | Memory usage per block |
| -u, --usage | Displays engine usage information |
| -p, --power | Current power usage |
| -c, --clock | Average, max, and current clock frequencies |
| -t, --temperature | Current temperatures |
| -P, --pcie | Current PCIe speed, width, and replay count |
| -e, --ecc | Total number of ECC errors |
| -k, --ecc-blocks | Number of ECC errors per block |
| -f, --fan | Current fan speed |
| -C, --voltage-curve | Display voltage curve |
| -o, --overdrive | Current GPU clock overdrive level |
| -l, --perf-level | Current DPM performance level |
| -x, --xgmi-err | XGMI error information since last read |
| -E, --energy | Amount of energy consumed |
| -T, --throttle | Displays throttle accumulators; Only |
| ↪available for MI300 or newer ASICs | |
| CPU Arguments: | |
| --cpu-power-metrics | CPU power metrics |
| --cpu-prochot | Displays prochot status |
| --cpu-freq-metrics | Displays currentFclkMemclk frequencies and |
| ↪cclk frequency limit | |
| --cpu-c0-res | Displays C0 residency |
| --cpu-lclk-dpm-level NBOID | Displays lclk dpm level range. Requires |
| ↪socket ID and NBOID as inputs | |
| --cpu-pwr-svi-telemetry-rails | Displays svi based telemetry for all rails |
| --cpu-io-bandwidth IO_BW LINKID_NAME | Displays current IO bandwidth for the |
| ↪selected CPU. | |
| | input parameters are bandwidth type(1) and |
| ↪link ID encodings | |
| | i.e. P2, P3, G0 - G7 |
| --cpu-xgmi-bandwidth XGMI_BW LINKID_NAME | Displays current XGMI bandwidth for the |
| ↪selected CPU | |
| | input parameters are bandwidth type(1,2,4) |
| ↪and link ID encodings | |
| | i.e. P2, P3, G0 - G7 |
| --cpu-metrics-ver | Displays metrics table version |
| --cpu-metrics-table | Displays metric table |
| --cpu-socket-energy | Displays socket energy for the selected CPU |
| ↪socket | |
| --cpu-ddr-bandwidth | Displays per socket max ddr bw, current |
| ↪utilized bw, | |
| | and current utilized ddr bw in percentage |
| --cpu-temp | Displays cpu socket temperature |
| --cpu-dimm-temp-range-rate DIMM_ADDR | Displays dimm temperature range and refresh |
| ↪rate | |
| --cpu-dimm-pow-consumption DIMM_ADDR | Displays dimm power consumption |
| --cpu-dimm-thermal-sensor DIMM_ADDR | Displays dimm thermal sensor |
| CPU Core Arguments: | |
| --core-boost-limit | Get boost limit for the selected cores |
| --core-curr-active-freq-core-limit | Get Current CCLK limit set per Core |

(continues on next page)

(continued from previous page)

```

--core-energy                Displays core energy for the selected core

Command Modifiers:
--json                      Displays output in JSON format (human-readable by default).
↪readable by default).
--csv                       Displays output in CSV format (human-readable by default).
↪readable by default).
--file FILE                 Saves output into a file on the provided path (stdout by default).
↪path (stdout by default).
--loglevel LEVEL           Set the logging level from the possible choices:
↪choices:
                                DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.6 amd-smi process

Lists general process information running on the specified GPU.

```

~$ amd-smi process --help
usage: amd-smi process [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                    [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]]
                    [-w INTERVAL] [-W TIME] [-i ITERATIONS] [-G] [-e] [-p PID]
                    [-n NAME]

```

If no GPU is specified, returns information for all GPUs on the system.
If no process argument is provided, all process information will be displayed.

```

Process arguments:
-h, --help                show this help message and exit
-g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
↪7e20764e2714             ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
↪7c8cefe1f084             ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
↪25d815189854             ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
↪4c2be5e66537             ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                                all | Selects all devices
-U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                                ID: 0
                                ID: 1
                                ID: 2
                                ID: 3
                                all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                                ID: 0 - 95
                                all | Selects all devices
-w, --watch INTERVAL     Reprint the command in a loop of INTERVAL seconds
-W, --watch_time TIME    The total TIME to watch the given command
-i, --iterations ITERATIONS Total number of ITERATIONS to loop on the given command
-G, --general             pid, process name, memory usage
-e, --engine              All engine usages

```

(continues on next page)

(continued from previous page)

| | |
|----------------------------|--|
| -p, --pid PID | Gets all process information about the specified process. |
| ↪based on Process ID | |
| -n, --name NAME | Gets all process information about the specified process. |
| ↪based on Process Name. | |
| | If multiple processes have the same name, information is |
| ↪returned for all of them. | |
| Command Modifiers: | |
| --json | Displays output in JSON format (human readable by |
| ↪default). | |
| --csv | Displays output in CSV format (human readable by default). |
| --file FILE | Saves output into a file on the provided path (stdout by |
| ↪default). | |
| --loglevel LEVEL | Set the logging level from the possible choices: |
| | DEBUG, INFO, WARNING, ERROR, CRITICAL |

5.3.7 amd-smi event

Displays event information for the given GPU.

```
~$ amd-smi event --help
usage: amd-smi event [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                  [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]]

If no GPU is specified, returns event information for all GPUs on the system.

Event Arguments:
  -h, --help                show this help message and exit
  -g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
  ↪7e20764e2714             ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
  ↪7c8cefe1f084             ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
  ↪25d815189854            ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
  ↪4c2be5e66537            ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                           all | Selects all devices
  -U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                           ID: 0
                           ID: 1
                           ID: 2
                           ID: 3
                           all | Selects all devices
  -O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                           ID: 0 - 95
                           all | Selects all devices

Command Modifiers:
  --json                    Displays output in JSON format (human readable by default).
  --csv                    Displays output in CSV format (human readable by default).
  --file FILE              Saves output into a file on the provided path (stdout by
```

(continues on next page)

(continued from previous page)

```
↪ default).
--loglevel LEVEL          Set the logging level from the possible choices:
                           DEBUG, INFO, WARNING, ERROR, CRITICAL
```

5.3.8 amd-smi topology

Displays topology information of the specified devices.

```
~$ amd-smi topology --help
usage: amd-smi topology [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                        [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]] [-a]
                        [-w] [-o] [-t] [-b]
```

If no GPU is specified, returns information for all GPUs on the system.
If no topology argument is provided, all topology information will be displayed.

Topology arguments:

```
-h, --help                show this help message and exit
-g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
↪ 7e20764e2714           ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
↪ 7c8cefe1f084           ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
↪ 25d815189854           ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
↪ 4c2be5e66537           ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                           all | Selects all devices
-U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                           ID: 0
                           ID: 1
                           ID: 2
                           ID: 3
                           all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                           ID: 0 - 95
                           all | Selects all devices
-a, --access              Displays link accessibility between GPUs
-w, --weight              Displays relative weight between GPUs
-o, --hops                Displays the number of hops between GPUs
-t, --link-type           Displays the link type between GPUs
-b, --numa-bw             Display max and min bandwidth between nodes
-c, --coherent            Display cache coherent (or non-coherent) link capability↪
↪ between nodes
-n, --atomics             Display 32 and 64-bit atomic io link capability between↪
↪ nodes
-d, --dma                 Display P2P direct memory access (DMA) link capability↪
↪ between nodes
-z, --bi-dir              Display P2P bi-directional link capability between nodes
```

Command Modifiers:

```
--json                    Displays output in JSON format (human readable by default).
```

(continues on next page)

(continued from previous page)

```

--csv           Displays output in CSV format (human readable by default).
--file FILE     Saves output into a file on the provided path (stdout by default).
--loglevel LEVEL Set the logging level from the possible choices:
                DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.9 amd-smi set

Set options for specified devices.

```

~$ amd-smi set --help
usage: amd-smi set [-h] (-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]) [-f %]
                  [-l LEVEL] [-P SETPROFILE] [-d SCLKMAX] [-C PARTITION] [-M PARTITION]
                  [-o WATTS] [-p POLICY_ID] [-x POLICY_ID] [-R STATUS]
                  [--cpu-pwr-limit PWR_LIMIT] [--cpu-xgmi-link-width MIN_WIDTH MAX_
                  WIDTH]
                  [--cpu-lclk-dpm-level NBIOD MIN_DPM MAX_DPM] [--cpu-pwr-eff-mode_
                  MODE]
                  [--cpu-gmi3-link-width MIN_LW MAX_LW] [--cpu-pcie-link-rate LINK_RATE]
                  [--cpu-df-pstate-range MAX_PSTATE MIN_PSTATE] [--cpu-enable-apb]
                  [--cpu-disable-apb DF_PSTATE] [--soc-boost-limit BOOST_LIMIT]
                  [--core-boost-limit BOOST_LIMIT] [--json | --csv] [--file FILE]
                  [--loglevel LEVEL]

```

If no GPU is specified, will select all GPUs on the system.

A set argument must be provided; Multiple set arguments are accepted

Set Arguments:

```

-h, --help           show this help message and exit
-g, --gpu GPU [GPU ...] Select a GPU ID, BDF, or UUID from the
possible choices:
ID: 0 | BDF: 0000:01:00.0 | UUID:
7eff74a0-0000-1000-808f-7e20764e2714
ID: 1 | BDF: 0001:01:00.0 | UUID:
b6ff74a0-0000-1000-80ae-7c8cefe1f084
ID: 2 | BDF: 0002:01:00.0 | UUID:
36ff74a0-0000-1000-8071-25d815189854
ID: 3 | BDF: 0003:01:00.0 | UUID:
f4ff74a0-0000-1000-80c4-4c2be5e66537
all | Selects all devices
-U, --cpu CPU [CPU ...] Select a CPU ID from the possible choices:
ID: 0
ID: 1
ID: 2
ID: 3
all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible
choices:
ID: 0 - 95
all | Selects all devices
-f, --fan %          Set GPU fan speed (0-255 or 0-100%)
-l, --perf-level LEVEL Set performance level

```

(continues on next page)

(continued from previous page)

| | |
|--|--|
| -P, --profile SETPROFILE | Set power profile level (#) or a quoted |
| ↳string of custom profile attributes | |
| -d, --perf-determinism SCLKMAX | Set GPU clock frequency limit and |
| ↳performance level to determinism to get minimal performance variation | |
| -C, --compute-partition PARTITION | Set one of the following the compute |
| ↳partition modes: | |
| | CPX, SPX, DPX, TPX, QPX |
| -M, --memory-partition PARTITION | Set one of the following the memory |
| ↳partition modes: | |
| | NPS1, NPS2, NPS4, NPS8 |
| -o, --power-cap WATTS | Set power capacity limit |
| -p, --soc-pstate POLICY_ID | Set the GPU soc pstate policy using |
| ↳policy id | |
| -x, --xgmi-plpd POLICY_ID | Set the GPU XGMI per-link power down |
| ↳policy using policy id | |
| -L, --clk-limit CLK_TYPE LIM_TYPE VALUE | Sets the sclk (aka gfxclk) or mclk |
| ↳minimum and maximum frequencies | |
| | amd-smi set -L (sclk mclk) (min max) |
| ↳value | |
| -R, --process-isolation STATUS | Enable or disable the GPU process |
| ↳isolation: | |
| | 0 for disable and 1 for enable. |
| CPU Arguments: | |
| --cpu-pwr-limit PWR_LIMIT | Set power limit for the given socket. |
| ↳Input parameter is power limit value. | |
| --cpu-xgmi-link-width MIN_WIDTH MAX_WIDTH | Set max and Min linkwidth. Input |
| ↳parameters are min and max link width values | |
| --cpu-lclk-dpm-level NBIOID MIN_DPM MAX_DPM | Sets the max and min dpm level on a given |
| ↳NBIO. | |
| | Input parameters are die_index, min dpm, |
| ↳max dpm. | |
| --cpu-pwr-eff-mode MODE | Sets the power efficiency mode policy. |
| ↳Input parameter is mode. | |
| --cpu-gmi3-link-width MIN_LW MAX_LW | Sets max and min gmi3 link width range |
| --cpu-pcie-link-rate LINK_RATE | Sets pcie link rate |
| --cpu-df-pstate-range MAX_PSTATE MIN_PSTATE | Sets max and min df-pstates |
| --cpu-enable-apb | Enables the DF p-state performance boost |
| ↳algorithm | |
| --cpu-disable-apb DF_PSTATE | Disables the DF p-state performance boost |
| ↳algorithm. Input parameter is DFPstate (0-3) | |
| --soc-boost-limit BOOST_LIMIT | Sets the boost limit for the given socket. |
| ↳ Input parameter is socket BOOST_LIMIT value | |
| CPU Core Arguments: | |
| --core-boost-limit BOOST_LIMIT | Sets the boost limit for the given core. |
| ↳Input parameter is core BOOST_LIMIT value | |
| Command Modifiers: | |
| --json | Displays output in JSON format (human |
| ↳readable by default). | |
| --csv | Displays output in CSV format (human |

(continues on next page)

(continued from previous page)

```

↪readable by default).
  --file FILE                               Saves output into a file on the provided.
↪path (stdout by default).
  --loglevel LEVEL                           Set the logging level from the possible.
↪choices:
                                             DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.10 amd-smi reset

Reset options for specified devices.

```

~$ amd-smi reset --help
usage: amd-smi reset [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                  (-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]) [-G] [-c]
                  [-f] [-p] [-x] [-d] [-C] [-M] [-o] [-l]

If no GPU is specified, will select all GPUs on the system.
A reset argument must be provided; Multiple reset arguments are accepted

Reset Arguments:
  -h, --help                show this help message and exit
  -g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
↪7e20764e2714              ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
↪7c8cefe1f084              ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
↪25d815189854              ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
↪4c2be5e66537              ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                           all | Selects all devices
  -U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                           ID: 0
                           ID: 1
                           ID: 2
                           ID: 3
                           all | Selects all devices
  -O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                           ID: 0 - 95
                           all | Selects all devices
  -G, --gpureset           Reset the specified GPU
  -c, --clocks             Reset clocks and overdrive to default
  -f, --fans               Reset fans to automatic (driver) control
  -p, --profile            Reset power profile back to default
  -x, --xgmierr           Reset XGMI error counts
  -d, --perf-determinism  Disable performance determinism
  -o, --power-cap         Reset power capacity limit to max capable
  -l, --clean-local-data  Clean up the local data in the LDS/GPRs on a per partition.
↪basis

Command Modifiers:
  --json                   Displays output in JSON format (human readable by default).

```

(continues on next page)

(continued from previous page)

```

--csv           Displays output in CSV format (human readable by default).
--file FILE     Saves output into a file on the provided path (stdout by default).
--loglevel LEVEL Set the logging level from the possible choices:
                DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.11 amd-smi monitor

Monitor metrics for target devices.

```

~$ amd-smi monitor --help
usage: amd-smi monitor [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                    [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]]
                    [-w INTERVAL] [-W TIME] [-i ITERATIONS] [-p] [-t] [-u] [-m] [-n]
                    [-d] [-e] [-v] [-r] [-q]

```

Monitor a target device for the specified arguments.
 If no arguments are provided, all arguments will be enabled.
 Use the watch arguments to run continuously.

Monitor Arguments:

```

-h, --help           show this help message and exit
-g, --gpu GPU [GPU ...] Select a GPU ID, BDF, or UUID from the possible choices:
                    ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
                    ↪7e20764e2714
                    ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
                    ↪7c8cefe1f084
                    ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
                    ↪25d815189854
                    ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                    ↪4c2be5e66537
                    all | Selects all devices
-U, --cpu CPU [CPU ...] Select a CPU ID from the possible choices:
                    ID: 0
                    ID: 1
                    ID: 2
                    ID: 3
                    all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                    ID: 0 - 95
                    all | Selects all devices
-w, --watch INTERVAL Reprint the command in a loop of INTERVAL seconds
-W, --watch_time TIME The total TIME to watch the given command
-i, --iterations ITERATIONS Total number of ITERATIONS to loop on the given command
-p, --power-usage      Monitor power usage in Watts
-t, --temperature      Monitor temperature in Celsius
-u, --gfx              Monitor graphics utilization (%) and clock (MHz)
-m, --mem              Monitor memory utilization (%) and clock (MHz)
-n, --encoder          Monitor encoder utilization (%) and clock (MHz)
-d, --decoder          Monitor decoder utilization (%) and clock (MHz)
-e, --ecc              Monitor ECC single bit, ECC double bit, and PCIe replay
↪error counts

```

(continues on next page)

(continued from previous page)

```

-v, --vram-usage      Monitor memory usage in MB
-r, --pcie            Monitor PCIe bandwidth in Mb/s
-q, --process         Enable Process information table below monitor output
-V, --violation       Monitor power and thermal violation status (%); Only
↳available for MI300 or newer ASICs

Command Modifiers:
--json                Displays output in JSON format (human readable by
↳default).
--csv                 Displays output in CSV format (human readable by default).
--file FILE           Saves output into a file on the provided path (stdout by
↳default).
--loglevel LEVEL      Set the logging level from the possible choices:
                      DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.12 amd-smi xgmi

Displays XGMI information of specified devices.

```

~$ amd-smi xgmi --help
usage: amd-smi xgmi [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                  [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]] [-m]

If no GPU is specified, returns information for all GPUs on the system.
If no xgmi argument is provided, all xgmi information will be displayed.

XGMI arguments:
-h, --help          show this help message and exit
-g, --gpu GPU [GPU ...] Select a GPU ID, BDF, or UUID from the possible choices:
↳7e20764e2714       ID: 0 | BDF: 0000:01:00.0 | UUID: 7eff74a0-0000-1000-808f-
↳7c8cefe1f084       ID: 1 | BDF: 0001:01:00.0 | UUID: b6ff74a0-0000-1000-80ae-
↳25d815189854       ID: 2 | BDF: 0002:01:00.0 | UUID: 36ff74a0-0000-1000-8071-
↳4c2be5e66537       ID: 3 | BDF: 0003:01:00.0 | UUID: f4ff74a0-0000-1000-80c4-
                    all | Selects all devices
-U, --cpu CPU [CPU ...] Select a CPU ID from the possible choices:
                    ID: 0
                    ID: 1
                    ID: 2
                    ID: 3
                    all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                    ID: 0 - 95
                    all | Selects all devices
-m, --metric         Metric XGMI information

Command Modifiers:
--json                Displays output in JSON format (human readable by default).
--csv                 Displays output in CSV format (human readable by default).

```

(continues on next page)

(continued from previous page)

```

--file FILE          Saves output into a file on the provided path (stdout by
↳ default).
--loglevel LEVEL     Set the logging level from the possible choices:
                     DEBUG, INFO, WARNING, ERROR, CRITICAL

```

5.3.13 Example output from amd-smi static

To gain a sense of the AMD SMI CLI's output, the following block is sample output from the CLI tool:

```

~$ amd-smi static
CPU: 0
SMU:
    FW_VERSION: 85.90.0
    INTERFACE_VERSION:
    PROTO VERSION: 6
...
GPU: 0
ASIC:
    MARKET_NAME: MI300A
    VENDOR_ID: 0x1002
    VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
    SUBVENDOR_ID: 0x1002
    DEVICE_ID: 0x74a0
    SUBSYSTEM_ID: 0x74a0
    REV_ID: 0x00
    ASIC_SERIAL: 0x7E8F7E20764E2714
    OAM_ID: 0
    NUM_COMPUTE_UNITS: 228
    TARGET_GRAPHICS_VERSION: gfx942
BUS:
    BDF: 0000:01:00.0
    MAX_PCIE_WIDTH: 16
    MAX_PCIE_SPEED: 32 GT/s
    PCIE_INTERFACE_VERSION: Gen 5
    SLOT_TYPE: PCIE
VBIOS:
    NAME: N/A
    BUILD_DATE: N/A
    PART_NUMBER: N/A
    VERSION: N/A
LIMIT:
    MAX_POWER: 550 W
    MIN_POWER: 0 W
    SOCKET_POWER: 550 W
    SLOWDOWN_EDGE_TEMPERATURE: N/A
    SLOWDOWN_HOTSPOT_TEMPERATURE: 100 °C
    SLOWDOWN_VRAM_TEMPERATURE: 105 °C
    SHUTDOWN_EDGE_TEMPERATURE: N/A
    SHUTDOWN_HOTSPOT_TEMPERATURE: 110 °C
    SHUTDOWN_VRAM_TEMPERATURE: 115 °C
DRIVER:

```

(continues on next page)

(continued from previous page)

```

NAME: amdgpu
VERSION: 6.9.0-rc5+
BOARD:
MODEL_NUMBER: N/A
PRODUCT_SERIAL: N/A
FRU_ID: N/A
PRODUCT_NAME: Aqua Vanjaram [Instinct MI300A]
MANUFACTURER_NAME: Advanced Micro Devices, Inc. [AMD/ATI]
RAS:
EEPROM_VERSION: 0x0
PARITY_SCHEMA: DISABLED
SINGLE_BIT_SCHEMA: DISABLED
DOUBLE_BIT_SCHEMA: DISABLED
POISON_SCHEMA: ENABLED
ECC_BLOCK_STATE:
  UMC: DISABLED
  SDMA: ENABLED
  GFX: ENABLED
  MMHUB: ENABLED
  ATHUB: DISABLED
  PCIE_BIF: DISABLED
  HDP: DISABLED
  XGMI_WAFL: DISABLED
  DF: DISABLED
  SMN: DISABLED
  SEM: DISABLED
  MP0: DISABLED
  MP1: DISABLED
  FUSE: DISABLED
  MCA: DISABLED
  VCN: DISABLED
  JPEG: DISABLED
  IH: DISABLED
  MPIO: DISABLED
PARTITION:
  COMPUTE_PARTITION: SPX
  MEMORY_PARTITION: NPS1
  PARTITION_ID: 0
SOC_PSTATE:
  NUM_SUPPORTED: 4
  CURRENT_ID: 1
  POLICIES:
    POLICY_ID: 0
    POLICY_DESCRIPTION: pstate_default
    POLICY_ID: 1
    POLICY_DESCRIPTION: soc_pstate_0
    POLICY_ID: 2
    POLICY_DESCRIPTION: soc_pstate_1
    POLICY_ID: 3
    POLICY_DESCRIPTION: soc_pstate_2
XGMI_PLPD:
  NUM_SUPPORTED: 3

```

(continues on next page)

(continued from previous page)

```
CURRENT_ID: 1
PLPDS:
  POLICY_ID: 0
  POLICY_DESCRIPTION: plpd_disallow
  POLICY_ID: 1
  POLICY_DESCRIPTION: plpd_default
  POLICY_ID: 2
  POLICY_DESCRIPTION: plpd_optimized
PROCESS_ISOLATION: N/A
NUMA:
  NODE: 0
  AFFINITY: 0
VRAM:
  TYPE: HBM
  VENDOR: N/A
  SIZE: 64289 MB
  BIT_WIDTH: 8192
CACHE_INFO:
  CACHE_0:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 32 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 348
  CACHE_1:
    CACHE_PROPERTIES: INST_CACHE, SIMD_CACHE
    CACHE_SIZE: 64 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 120
  CACHE_2:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 4096 KB
    CACHE_LEVEL: 2
    MAX_NUM_CU_SHARED: 228
    NUM_CACHE_INSTANCE: 1
  CACHE_3:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 262144 KB
    CACHE_LEVEL: 3
    MAX_NUM_CU_SHARED: 228
    NUM_CACHE_INSTANCE: 1
```

...

AMD SMI C++ API REFERENCE

This section provides comprehensive documentation for the AMD SMI C++ API. Explore these sections to understand the full scope of available functionalities and how to implement them in your applications.

- *Modules*
- *Files*
- *Globals*
- *Data structures*

6.1 Modules

6.2 File List

6.3 Globals

6.4 Data Structures

6.5 Data Fields

6.5.1 Data Fields

6.5.2 Data Fields - Variables

AMD SMI PYTHON API REFERENCE

The AMD SMI Python interface provides a convenient way to interact with AMD hardware through a simple and accessible API. Compatible with Python 3.6 and higher, this library requires the AMD driver to be loaded for initialization – review the *prerequisites*.

This section provides comprehensive documentation for the AMD SMI Python API. Explore these sections to understand the full scope of available functionalities and how to implement them in your applications.

7.1 API

7.1.1 `amdsmi_init`

Description: Initialize `amdsmi` with `AmdSmiInitFlags`

Input parameters: `AmdSmiInitFlags`

Output: None

Exceptions that can be thrown by `amdsmi_init` function:

- `AmdSmiLibraryException`

Initialize GPUs only example:

```
try:
    # by default we initialize with AmdSmiInitFlags.INIT_AMD_GPUS
    ret = amdsmi_init()
    # continue with amdsmi
except AmdSmiException as e:
    print("Init GPUs failed")
    print(e)
```

Initialize CPUs only example:

```
try:
    ret = amdsmi_init(AmdSmiInitFlags.INIT_AMD_CPUS)
    # continue with amdsmi
except AmdSmiException as e:
    print("Init CPUs failed")
    print(e)
```

Initialize both GPUs and CPUs example:

```
try:
    ret = amdsmi_init(AmdSmiInitFlags.INIT_AMD_APUS)
    # continue with amdsmi
except AmdSmiException as e:
    print("Init both GPUs & CPUs failed")
    print(e)
```

7.1.2 amdsmi_shut_down

Description: Finalize and close connection to driver

Input parameters: None

Output: None

Exceptions that can be thrown by amdsmi_shut_down function:

- AmdSmiLibraryException

Example:

```
try:
    amdsmi_init()
    amdsmi_shut_down()
except AmdSmiException as e:
    print("Shut down failed")
    print(e)
```

7.1.3 amdsmi_get_processor_type

Description: Checks the type of device with provided handle.

Input parameters: device handle as an instance of amdsmi_processor_handle

Output: Integer, type of gpu

Exceptions that can be thrown by amdsmi_get_processor_type function:

- AmdSmiLibraryException

Example:

```
try:
    type_of_GPU = amdsmi_get_processor_type(processor_handle)
    if type_of_GPU == 1:
        print("This is an AMD GPU")
except AmdSmiException as e:
    print(e)
```

7.1.4 amdsmi_get_processor_handles

Description: Returns list of GPU device handle objects on current machine

Input parameters: None

Output: List of GPU device handle objects

Exceptions that can be thrown by amdsmi_get_processor_handles function:

- AmdSmiLibraryException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            print(amdsmi_get_gpu_device_uuid(device))
except AmdSmiException as e:
    print(e)
```

7.1.5 amdsmi_get_socket_handles

Note: CURRENTLY HARDCODED TO RETURN DUMMY DATA

Description: Returns list of socket device handle objects on current machine

Input parameters: None

Output: List of socket device handle objects

Exceptions that can be thrown by amdsmi_get_socket_handles function:

- AmdSmiLibraryException

Example:

```
try:
    sockets = amdsmi_get_socket_handles()
    print('Socket numbers: {}'.format(len(sockets)))
except AmdSmiException as e:
    print(e)
```

7.1.6 amdsmi_get_socket_info

Note: CURRENTLY HARDCODED TO RETURN EMPTY VALUES

Description: Return socket name

Input parameters: socket_handle socket handle

Output: Socket name

Exceptions that can be thrown by amdsmi_get_socket_info function:

- AmdSmiLibraryException

Example:

```
try:
    socket_handles = amdsmi_get_socket_handles()
    if len(socket_handles) == 0:
        print("No sockets on machine")
    else:
        for socket in socket_handles:
            print(amdsmi_get_socket_info(socket))
except AmdSmiException as e:
    print(e)
```

7.1.7 amdsmi_get_processor_handle_from_bdf

Description: Returns device handle from the given BDF

Input parameters: bdf string in form of either <domain>:<bus>:<device>.<function> or <bus>:<device>.<function> in hexcode format. Where:

- <domain> is 4 hex digits long from 0000-FFFF interval
- <bus> is 2 hex digits long from 00-FF interval
- <device> is 2 hex digits long from 00-1F interval
- <function> is 1 hex digit long from 0-7 interval

Output: device handle object

Exceptions that can be thrown by amdsmi_get_processor_handle_from_bdf function:

- AmdSmiLibraryException
- AmdSmiBdfFormatException

Example:

```
try:
    device = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
    print(amdsmi_get_gpu_device_uuid(device))
except AmdSmiException as e:
    print(e)
```

7.1.8 amdsmi_get_gpu_device_bdf

Description: Returns BDF of the given device

Input parameters:

- processor_handle dev for which to query

Output: BDF string in form of <domain>:<bus>:<device>.<function> in hexcode format. Where:

- <domain> is 4 hex digits long from 0000-FFFF interval
- <bus> is 2 hex digits long from 00-FF interval
- <device> is 2 hex digits long from 00-1F interval
- <function> is 1 hex digit long from 0-7 interval

Exceptions that can be thrown by amdsmi_get_gpu_device_bdf function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```
try:
    device = amdsmi_get_processor_handles()[0]
    print("Device's bdf:", amdsmi_get_gpu_device_bdf(device))
except AmdSmiException as e:
    print(e)
```

7.1.9 amdsmi_get_gpu_device_uuid

Description: Returns the UUID of the device

Input parameters:

- `processor_handle` dev for which to query

Output: UUID string unique to the device

Exceptions that can be thrown by `amdsmi_get_gpu_device_uuid` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
    device = amdsmi_get_processor_handles()[0]
    print("Device UUID: ", amdsmi_get_gpu_device_uuid(device))
except AmdSmiException as e:
    print(e)
```

7.1.10 amdsmi_get_gpu_driver_info

Description: Returns the info of the driver

Input parameters:

- `processor_handle` dev for which to query

Output: Dictionary with fields

| Field | Content |
|-----------------------------|----------------|
| <code>driver_name</code> | driver name |
| <code>driver_version</code> | driver_version |
| <code>driver_date</code> | driver_date |

Exceptions that can be thrown by `amdsmi_get_gpu_driver_info` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
    device = amdsmi_get_processor_handles()[0]
    print("Driver info: ", amdsmi_get_gpu_driver_info(device))
except AmdSmiException as e:
    print(e)
```

7.1.11 amdsmi_get_gpu_asic_info

Description: Returns asic information for the given GPU

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Content |
|-------------------------|---------------------------------|
| market_name | market name |
| vendor_id | vendor id |
| vendor_name | vendor name |
| device_id | device id |
| rev_id | revision id |
| asic_serial | asic serial |
| oam_id | oam id |
| num_of_compute_units | number of compute units on asic |
| target_graphics_version | hardware graphics version |

Exceptions that can be thrown by `amdsmi_get_gpu_asic_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            asic_info = amdsmi_get_gpu_asic_info(device)
            print(asic_info)
except AmdSmiException as e:
    print(e)
```

7.1.12 `amdsmi_get_gpu_kfd_info`

Description: Returns KFD(kernel fusion driver) information for the given GPU This correlates to GUID in rocm-smi

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Content |
|---------|-----------------------------|
| kfd_id | KFD's unique GPU identifier |
| node_id | KFD's internal GPU index |

Exceptions that can be thrown by `amdsmi_get_gpu_kfd_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            kfd_info = amdsmi_get_gpu_kfd_info(device)
            print(kfd_info)
except AmdSmiException as e:
    print(e)

```

7.1.13 amdsmi_get_power_cap_info

Description: Returns dictionary of power capabilities as currently configured on the given GPU. It is not supported on virtual machine guest

Input parameters:

- processor_handle device which to query

Output: Dictionary with fields

| Field | Description | Units |
|-------------------|-------------------------------------|-------|
| power_cap | power capability | uW |
| dpm_cap | dynamic power management capability | MHz |
| default_power_cap | default power capability | uW |
| min_power_cap | min power capability | uW |
| max_power_cap | max power capability | uW |

Exceptions that can be thrown by amdsmi_get_power_cap_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            power_info = amdsmi_get_power_cap_info(device)
            print(power_info['power_cap'])
            print(power_info['dpm_cap'])
            print(power_info['default_power_cap'])
            print(power_info['min_power_cap'])
            print(power_info['max_power_cap'])
except AmdSmiException as e:
    print(e)

```

7.1.14 amdsmi_get_gpu_vram_info

Description: Returns dictionary of vram information for the given GPU.

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Description |
|-----------------------------|-----------------|
| <code>vram_type</code> | vram type |
| <code>vram_vendor</code> | vram vendor |
| <code>vram_size</code> | vram size in mb |
| <code>vram_bit_width</code> | vram bit width |

Exceptions that can be thrown by `amdsmi_get_gpu_vram_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vram_info = amdsmi_get_gpu_vram_info(device)
            print(vram_info['vram_type'])
            print(vram_info['vram_vendor'])
            print(vram_info['vram_size'])
            print(vram_info['vram_bit_width'])
except AmdSmiException as e:
    print(e)
```

7.1.15 amdsmi_get_gpu_cache_info

Description: Returns a list of dictionaries containing cache information for the given GPU.

Input parameters:

- `processor_handle` device which to query

Output: List of Dictionaries containing cache information following the schema below: Schema:

```
{
  "cache_properties":
  {
    "type" : "array",
    "items" : {"type" : "string"}
  },
  "cache_size": {"type" : "number"},
```

(continues on next page)

(continued from previous page)

```

"cache_level": {"type" : "number"},
"max_num_cu_shared": {"type" : "number"},
"num_cache_instance": {"type" : "number"}
}

```

| Field | Description |
|--------------|--|
| cache_proper | list of up to 4 cache property type strings. Ex. data (“DATA_CACHE”), instruction (“INST_CACHE”), CPU (“CPU_CACHE”), or SIMD (“SIMD_CACHE”). |
| cache_size | size of cache in KB |
| cache_level | level of cache |
| max_num_cu_s | max number of compute units shared |
| num_cache_in | number of cache instances |

Exceptions that can be thrown by `amdsmi_get_gpu_cache_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            cache_info = amdsmi_get_gpu_cache_info(device)
            for cache_index, cache_values in cache_info.items():
                print(cache_values['cache_properties'])
                print(cache_values['cache_size'])
                print(cache_values['cache_level'])
                print(cache_values['max_num_cu_shared'])
                print(cache_values['num_cache_instance'])
except AmdSmiException as e:
    print(e)

```

7.1.16 `amdsmi_get_gpu_vbios_info`

Description: Returns the static information for the VBIOS on the device.

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Description |
|-------------|----------------------|
| name | vbios name |
| build_date | vbios build date |
| part_number | vbios part number |
| version | vbios version string |

Exceptions that can be thrown by `amdsmi_get_gpu_vbios_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vbios_info = amdsmi_get_gpu_vbios_info(device)
            print(vbios_info['name'])
            print(vbios_info['build_date'])
            print(vbios_info['part_number'])
            print(vbios_info['version'])
except AmdSmiException as e:
    print(e)
```

7.1.17 amdsmi_get_fw_info

Description: Returns GPU firmware related information.

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Description |
|----------------------|--|
| <code>fw_list</code> | List of dictionaries that contain information about a certain firmware block |

Exceptions that can be thrown by `amdsmi_get_fw_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
```

(continues on next page)

(continued from previous page)

```

if len(devices) == 0:
    print("No GPUs on machine")
else:
    for device in devices:
        firmware_list = amdsmi_get_fw_info(device)['fw_list']
        for firmware_block in firmware_list:
            print(firmware_block['fw_name'])
            # String formatted hex or decimal value ie: 21.00.00.AC or 130
            print(firmware_block['fw_version'])
except AmdSmiException as e:
    print(e)

```

7.1.18 amdsmi_get_gpu_activity

Description: Returns the engine usage for the given GPU. It is not supported on virtual machine guest

Input parameters:

- processor_handle device which to query

Output: Dictionary of activities to their respective usage percentage or 'N/A' if not supported

| Field | Description |
|--------------|--|
| gfx_activity | graphics engine usage percentage (0 - 100) |
| umc_activity | memory engine usage percentage (0 - 100) |
| mm_activity | average multimedia engine usages in percentage (0 - 100) |

Exceptions that can be thrown by amdsmi_get_gpu_activity function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            engine_usage = amdsmi_get_gpu_activity(device)
            print(engine_usage['gfx_activity'])
            print(engine_usage['umc_activity'])
            print(engine_usage['mm_activity'])
except AmdSmiException as e:
    print(e)

```

7.1.19 amdsmi_get_power_info

Description: Returns the current power and voltage for the given GPU. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Description |
|-----------------------------------|----------------------|
| <code>current_socket_power</code> | current socket power |
| <code>average_socket_power</code> | average socket power |
| <code>gfx_voltage</code> | voltage gfx |
| <code>soc_voltage</code> | voltage soc |
| <code>mem_voltage</code> | voltage mem |
| <code>power_limit</code> | power limit |

Exceptions that can be thrown by `amdsmi_get_power_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            power_measure = amdsmi_get_power_info(device)
            print(power_measure['current_socket_power'])
            print(power_measure['average_socket_power'])
            print(power_measure['gfx_voltage'])
            print(power_measure['soc_voltage'])
            print(power_measure['mem_voltage'])
            print(power_measure['power_limit'])
except AmdSmiException as e:
    print(e)
```

7.1.20 amdsmi_get_gpu_vram_usage

Description: Returns total VRAM and VRAM in use

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Description |
|-------------------------|-----------------------|
| <code>vram_total</code> | VRAM total |
| <code>vram_used</code> | VRAM currently in use |

Exceptions that can be thrown by `amdsmi_get_gpu_vram_usage` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vram_usage = amdsmi_get_gpu_vram_usage(device)
            print(vram_usage['vram_used'])
            print(vram_usage['vram_total'])
except AmdSmiException as e:
    print(e)
```

7.1.21 amdsmi_get_clock_info

Description: Returns the clock measure for the given GPU. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query
- `clock_type` one of `AmdSmiClkType` enum values:

| Field | Description |
|-------|------------------|
| SYS | SYS clock type |
| GFX | GFX clock type |
| DF | DF clock type |
| DCEF | DCEF clock type |
| SOC | SOC clock type |
| MEM | MEM clock type |
| PCIE | PCIE clock type |
| VCLK0 | VCLK0 clock type |
| VCLK1 | VCLK1 clock type |
| DCLK0 | DCLK0 clock type |
| DCLK1 | DCLK1 clock type |

Output: Dictionary with fields

| Field | Description |
|-----------------------------|---|
| <code>clk</code> | Current clock for given clock type |
| <code>min_clk</code> | Minimum clock for given clock type |
| <code>max_clk</code> | Maximum clock for given clock type |
| <code>clk_locked</code> | flag only supported on GFX clock domain |
| <code>clk_deep_sleep</code> | clock deep sleep mode flag |

Exceptions that can be thrown by `amdsmi_get_clock_info` function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            clock_measure = amdsmi_get_clock_info(device, AmdSmiClkType.GFX)
            print(clock_measure['clk'])
            print(clock_measure['min_clk'])
            print(clock_measure['max_clk'])
            print(clock_measure['clk_locked'])
            print(clock_measure['clk_deep_sleep'])
except AmdSmiException as e:
    print(e)
```

7.1.22 amdsmi_get_pcie_info

Description: Returns the pcie metric and static information for the given GPU. For accurate PCIe Bandwidth measurements it is recommended to use this function once per 1000ms It is not supported on virtual machine guest

Input parameters:

- processor_handle device which to query

Output: Dictionary with 2 fields pcie_static and pcie_metric

| Field | Description |
|-------|---|
| pcie | Subfield Description max_pcie_widthMaximum number of pcie lanes availablemax_pcie_speedMaximum capable pcie speed in GT/spcie_interface_versionPCIe generation ie. 3,4,5... slot_typeThe type of form factor of the slot: OAM, PCIE, CEM, or Unknown |
| pcie | Subfield Description pcie_widthCurrent number of pcie lanes availablepcie_speedCurrent pcie speed capable in GT/spcie_bandwidthCurrent instantaneous bandwidth usage in Mb/spcie_replay_countTotal number of PCIe replays (NAKs)pcie_l0_to_recovery_countPCIe L0 to recovery state transition accumulated countpcie_replay_roll_over_countPCIe Replay accumulated countpcie_nak_sent_countPCIe NAK sent accumulated countpcie_nak_received_countPCIe NAK received accumulated count |

Exceptions that can be thrown by amdsmi_get_pcie_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
```

(continues on next page)

(continued from previous page)

```

if len(devices) == 0:
    print("No GPUs on machine")
else:
    for device in devices:
        pcie_info = amdsmi_get_pcie_info(device)
        print(pcie_info["pcie_static"])
        print(pcie_info["pcie_metric"])
except AmdSmiException as e:
    print(e)

```

7.1.23 amdsmi_get_gpu_bad_page_info

Description: Returns bad page info for the given GPU. It is not supported on virtual machine guest

Input parameters:

- processor_handle device which to query

Output: List consisting of dictionaries with fields for each bad page found; can be an empty list

| Field | Description |
|--------------|---------------------|
| value | Value of page |
| page_address | Address of bad page |
| page_size | Size of bad page |
| status | Status of bad page |

Exceptions that can be thrown by amdsmi_get_gpu_bad_page_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            bad_page_info = amdsmi_get_gpu_bad_page_info(device)
            if not bad_page_info: # Can be empty list
                print("No bad pages found")
                continue
            for bad_page in bad_page_info:
                print(bad_page["value"])
                print(bad_page["page_address"])
                print(bad_page["page_size"])
                print(bad_page["status"])
except AmdSmiException as e:
    print(e)

```

7.1.24 amdsmi_get_gpu_memory_reserved_pages

Description: Returns reserved memory page info for the given GPU. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: List consisting of dictionaries with fields for each reserved memory page found; can be an empty list

| Field | Description |
|---------------------------|---------------------------------|
| <code>value</code> | Value of memory reserved page |
| <code>page_address</code> | Address of memory reserved page |
| <code>page_size</code> | Size of memory reserved page |
| <code>status</code> | Status of memory reserved page |

Exceptions that can be thrown by `amdsmi_get_gpu_memory_reserved_pages` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            reserved_memory_page_info = amdsmi_get_gpu_memory_reserved_pages(device)
            if not reserved_memory_page_info: # Can be empty list
                print("No memory reserved pages found")
                continue
            for reserved_memory_page in reserved_memory_page_info:
                print(reserved_memory_page["value"])
                print(reserved_memory_page["page_address"])
                print(reserved_memory_page["page_size"])
                print(reserved_memory_page["status"])
except AmdSmiException as e:
    print(e)
```

7.1.25 amdsmi_get_gpu_process_list

Description: Returns the list of processes running on the target GPU; Requires root level access to display root process names; otherwise will return "N/A"

Input parameters:

- `processor_handle` device which to query

Output: List of Dictionaries with the corresponding fields; empty list if no running process are detected

| Field | Description |
|--------------|---|
| name | Name of process. If user does not have permission this will be "N/A" |
| pid | Process ID |
| mem | Process memory usage |
| engine_usage | Subfield Description gfxGFX engine usage in nsencEncode engine usage in ns |
| memory_usage | Subfield Description gtt_memGTT memory usagecpu_memCPU memory usageevram_memVRAM memory usage |

Exceptions that can be thrown by `amdsmi_get_gpu_process_list` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            processes = amdsmi_get_gpu_process_list(device)
            if len(processes) == 0:
                print("No processes running on this GPU")
            else:
                for process in processes:
                    print(process)
except AmdSmiException as e:
    print(e)
```

7.1.26 amdsmi_get_gpu_total_ecc_count

Description: Returns the ECC error count for the given GPU. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Description |
|----------------------------------|-------------------------------|
| <code>correctable_count</code> | Correctable ECC error count |
| <code>uncorrectable_count</code> | Uncorrectable ECC error count |
| <code>deferred_count</code> | Deferred ECC error count |

Exceptions that can be thrown by `amdsmi_get_gpu_total_ecc_count` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            ecc_error_count = amdsmi_get_gpu_total_ecc_count(device)
            print(ecc_error_count["correctable_count"])
            print(ecc_error_count["uncorrectable_count"])
except AmdSmiException as e:
    print(e)
```

7.1.27 amdsmi_get_gpu_board_info

Description: Returns board info for the given GPU

Input parameters:

- processor_handle device which to query

Output: Dictionary with fields correctable and uncorrectable

| Field | Description |
|-------------------|---------------------|
| model_number | Board serial number |
| product_serial | Product serial |
| fru_id | FRU ID |
| product_name | Product name |
| manufacturer_name | Manufacturer name |

Exceptions that can be thrown by amdsmi_get_gpu_board_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    device = amdsmi_get_processor_handle_from_bdf("0000:23.00.0")
    board_info = amdsmi_get_gpu_board_info(device)
    print(board_info["model_number"])
    print(board_info["product_serial"])
    print(board_info["fru_id"])
    print(board_info["product_name"])
    print(board_info["manufacturer_name"])
except AmdSmiException as e:
    print(e)
```

7.1.28 amdsmi_get_gpu_ras_feature_info

Description: Returns RAS version and schema information It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: List containing dictionaries with fields

| Field | Description |
|--------------------------------|-------------------|
| <code>eeeprom_version</code> | eeeprom version |
| <code>parity_schema</code> | parity schema |
| <code>single_bit_schema</code> | single bit schema |
| <code>double_bit_schema</code> | double bit schema |
| <code>poison_schema</code> | poison schema |

Exceptions that can be thrown by `amdsmi_get_gpu_ras_feature_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            ras_info = amdsmi_get_gpu_ras_feature_info(device)
            print(ras_info)
except AmdSmiException as e:
    print(e)
```

7.1.29 amdsmi_get_gpu_ras_block_features_enabled

Description: Returns status of each RAS block for the given GPU. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: List containing dictionaries with fields for each RAS block

| Field | Description |
|---------------------|------------------|
| <code>block</code> | RAS block |
| <code>status</code> | RAS block status |

Exceptions that can be thrown by `amdsmi_get_gpu_ras_block_features_enabled` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`

- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            ras_block_features = amdsmi_get_gpu_ras_block_features_enabled(device)
            print(ras_block_features)
except AmdSmiException as e:
    print(e)
```

7.1.30 AmdSmiEventReader class

Description: Providing methods for event monitoring. This is context manager class. Can be used with `with` statement for automatic cleanup.

Methods:

Constructor

Description: Allocates a new event reader notifier to monitor different types of events for the given GPU

Input parameters:

- `processor_handle` device handle corresponding to the device on which to listen for events
- `event_types` list of event types from `AmdSmiEvtNotificationType` enum. Specifying which events to collect for the given device.

| Event Type | Description |
|------------------|------------------|
| VMFAULT | VM page fault |
| THERMAL_THROTTLE | thermal throttle |
| GPU_PRE_RESET | gpu pre reset |
| GPU_POST_RESET | gpu post reset |
| RING_HANG | ring hang event |

read

Description: Reads events on the given device. When event is caught, device handle, message and event type are returned. Reading events stops when timestamp passes without event reading.

Input parameters:

- `timestamp` number of milliseconds to wait for an event to occur. If event does not happen monitoring is finished
- `num_elem` number of events. This is optional parameter. Default value is 10.

stop

Description: Any resources used by event notification for the the given device will be freed with this function. This can be used explicitly or automatically using `with` statement, like in the examples below. This should be called either manually or automatically for every created `AmdSmiEventReader` object.

Input parameters: None

Example with manual cleanup of AmdSmiEventReader:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        event = AmdSmiEventReader(device[0], AmdSmiEvtNotificationType.GPU_PRE_RESET,
        ↪AmdSmiEvtNotificationType.GPU_POST_RESET)
        event.read(10000)
except AmdSmiException as e:
    print(e)
finally:
    event.stop()
```

Example with automatic cleanup using with statement:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        with AmdSmiEventReader(device[0], AmdSmiEvtNotificationType.GPU_PRE_RESET,
        ↪AmdSmiEvtNotificationType.GPU_POST_RESET) as event:
            event.read(10000)
except AmdSmiException as e:
    print(e)
```

7.1.31 amdsmi_set_gpu_pci_bandwidth

Description: Control the set of allowed PCIe bandwidths that can be used It is not supported on virtual machine guest

Input parameters:

- processor_handle handle for the given device
- bw_bitmask A bitmask indicating the indices of the bandwidths that are to be enabled (1) and disabled (0)

Output: None

Exceptions that can be thrown by amdsmi_set_gpu_pci_bandwidth function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_pci_bandwidth(device, 0)
```

(continues on next page)

```
except AmdSmiException as e:  
    print(e)
```

7.1.32 amdsmi_set_power_cap

Description: Set the power cap value. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_ind` a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0
- `cap` int that indicates the desired power cap, in microwatts

Output: None

Exceptions that can be thrown by `amdsmi_set_power_cap` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:  
    devices = amdsmi_get_processor_handles()  
    if len(devices) == 0:  
        print("No GPUs on machine")  
    else:  
        for device in devices:  
            power_cap = 250 * 1000000  
            amdsmi_set_power_cap(device, 0, power_cap)  
except AmdSmiException as e:  
    print(e)
```

7.1.33 amdsmi_set_gpu_power_profile

Description: Set the power profile. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `reserved` Not currently used, set to 0
- `profile` a `amdsmi_power_profile_preset_masks_t` that hold the mask of the desired new power profile

Output: None

Exceptions that can be thrown by `amdsmi_set_gpu_power_profile` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            profile = ...
            amdsmi_set_gpu_power_profile(device, 0, profile)
except AmdSmiException as e:
    print(e)

```

7.1.34 amdsmi_set_gpu_clk_range

Description: This function sets the clock range information. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `min_clk_value` minimum clock value for desired clock range
- `max_clk_value` maximum clock value for desired clock range
- `clk_type` `AMDSMI_CLK_TYPE_SYS` | `AMDSMI_CLK_TYPE_MEM` range type

Output: None

Exceptions that can be thrown by `amdsmi_set_gpu_clk_range` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_clk_range(device, 0, 1000, AmdSmiClkType.AMDSMI_CLK_TYPE_SYS)
except AmdSmiException as e:
    print(e)

```

7.1.35 amdsmi_get_gpu_bdf_id

Description: Get the unique PCI device identifier associated for a device

Input parameters:

- `processor_handle` device which to query

Output: device bdf The format of bdfid will be as follows:

$BDFID = ((DOMAIN \& 0xffffffff) \ll 32) | ((BUS \& 0xff) \ll 8) | ((DEVICE \& 0x1f) \ll 3) | (FUNCTION \& 0x7)$

| Name | Field |
|----------|---------|
| Domain | [64:32] |
| Reserved | [31:16] |
| Bus | [15: 8] |
| Device | [7: 3] |
| Function | [2: 0] |

Exceptions that can be thrown by `amdsmi_get_gpu_bdf_id` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            bdfid = amdsmi_get_gpu_bdf_id(device)
            print(bdfid)
except AmdSmiException as e:
    print(e)
```

7.1.36 `amdsmi_get_gpu_pci_bandwidth`

Description: Get the list of possible PCIe bandwidths that are available. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with the possible T/s values and associated number of lanes

| Field | Content |
|----------------------------|---------------------------------------|
| <code>transfer_rate</code> | <code>transfer_rate</code> dictionary |
| <code>lanes</code> | <code>lanes</code> |

`transfer_rate` dictionary

| Field | Content |
|----------------------------|----------------------------|
| <code>num_supported</code> | <code>num_supported</code> |
| <code>current</code> | <code>current</code> |
| <code>frequency</code> | list of frequency |

Exceptions that can be thrown by `amdsmi_get_gpu_pci_bandwidth` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`

- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            bandwidth = amdsmi_get_gpu_pci_bandwidth(device)
            print(bandwidth)
except AmdSmiException as e:
    print(e)
```

7.1.37 amdsmi_get_gpu_pci_throughput

Description: Get PCIe traffic information. It is not supported on virtual machine guest

Input parameters:

- processor_handle device which to query

Output: Dictionary with the fields

| Field | Content |
|------------|----------------------------------|
| sent | number of bytes sent in 1 second |
| received | the number of bytes received |
| max_pkt_sz | maximum packet size |

Exceptions that can be thrown by amdsmi_get_gpu_pci_throughput function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            pci = amdsmi_get_gpu_pci_throughput(device)
            print(pci)
except AmdSmiException as e:
    print(e)
```

7.1.38 amdsmi_get_gpu_pci_replay_counter

Description: Get PCIe replay counter

Input parameters:

- `processor_handle` device which to query

Output: counter value The sum of the NAK's received and generated by the GPU

Exceptions that can be thrown by `amdsmi_get_gpu_pci_replay_counter` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            counter = amdsmi_get_gpu_pci_replay_counter(device)
            print(counter)
except AmdSmiException as e:
    print(e)
```

7.1.39 `amdsmi_get_gpu_topo_numa_affinity`

Description: Get the NUMA node associated with a device

Input parameters:

- `processor_handle` device which to query

Output: NUMA node value

Exceptions that can be thrown by `amdsmi_get_gpu_topo_numa_affinity` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            numa_node = amdsmi_get_gpu_topo_numa_affinity(device)
            print(numa_node)
except AmdSmiException as e:
    print(e)
```

7.1.40 amdsmi_get_energy_count

Description: Get the energy accumulator counter information of the device. $\text{energy_accumulator} * \text{counter_resolution} = \text{total_energy_consumption}$ in micro-Joules It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query

Output: Dictionary with fields

| Field | Content |
|---------------------------------|--|
| <code>power</code> | counter for energy accumulation since last restart/gpu rest (Deprecating in 6.4) |
| <code>energy_accumulator</code> | counter for energy accumulation since last restart/gpu rest |
| <code>counter_resolution</code> | counter resolution |
| <code>timestamp</code> | timestamp |

Exceptions that can be thrown by `amdsmi_get_energy_count` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            energy_dict = amdsmi_get_energy_count(device)
            print(energy_dict)
except AmdSmiException as e:
    print(e)
```

7.1.41 amdsmi_get_gpu_memory_total

Description: Get the total amount of memory that exists

Input parameters:

- `processor_handle` device which to query
- `mem_type` enum `AmdSmiMemoryType`

Output: total amount of memory

Exceptions that can be thrown by `amdsmi_get_gpu_memory_total` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vram_memory_total = amdsmi_get_gpu_memory_total(device, amdsmi_interface.
↪AmdSmiMemoryType.VRAM)
            print(vram_memory_total)
            vis_vram_memory_total = amdsmi_get_gpu_memory_total(device, amdsmi_interface.
↪AmdSmiMemoryType.VIS_VRAM)
            print(vis_vram_memory_total)
            gtt_memory_total = amdsmi_get_gpu_memory_total(device, amdsmi_interface.
↪AmdSmiMemoryType.GTT)
            print(gtt_memory_total)
except AmdSmiException as e:
    print(e)

```

7.1.42 amdsmi_set_gpu_od_clk_info

Description: This function sets the clock frequency information. It is not supported on virtual machine guest

Input parameters:

- processor_handle handle for the given device
- level AMDSMI_FREQ_IND_MIN|AMDSMI_FREQ_IND_MAX to set the minimum (0) or maximum (1) speed
- clk_value value to apply to the clock range
- clk_type AMDSMI_CLK_TYPE_SYS | AMDSMI_CLK_TYPE_MEM range type

Output: None

Exceptions that can be thrown by amdsmi_set_gpu_od_clk_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_od_clk_info(
                device,
                AmdSmiFreqInd.AMDSMI_FREQ_IND_MAX,
                1000,
                AmdSmiClkType.AMDSMI_CLK_TYPE_SYS
            )

```

(continues on next page)

(continued from previous page)

```
except AmdSmiException as e:
    print(e)
```

7.1.43 amdsmi_get_gpu_memory_usage

Description: Get the current memory usage

Input parameters:

- `processor_handle` device which to query
- `mem_type` enum `AmdSmiMemoryType`

Output: the amount of memory currently being used

Exceptions that can be thrown by `amdsmi_get_gpu_memory_usage` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vram_memory_usage = amdsmi_get_gpu_memory_usage(device, amdsmi_interface.
↪AmdSmiMemoryType.VRAM)
            print(vram_memory_usage)
            vis_vram_memory_usage = amdsmi_get_gpu_memory_usage(device, amdsmi_interface.
↪AmdSmiMemoryType.VIS_VRAM)
            print(vis_vram_memory_usage)
            gtt_memory_usage = amdsmi_get_gpu_memory_usage(device, amdsmi_interface.
↪AmdSmiMemoryType.GTT)
            print(gtt_memory_usage)
except AmdSmiException as e:
    print(e)
```

7.1.44 amdsmi_set_gpu_od_volt_info

Description: This function sets 1 of the 3 voltage curve points. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `vpoint` voltage point [0|1|2] on the voltage curve
- `clk_value` clock value component of voltage curve point
- `volt_value` voltage value component of voltage curve point

Output: None

Exceptions that can be thrown by `amdsmi_set_gpu_od_volt_info` function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_od_volt_info(device, 1, 1000, 980)
except AmdSmiException as e:
    print(e)
```

7.1.45 amdsmi_get_gpu_fan_rpms

Description: Get the fan speed in RPMs of the device with the specified device handle and 0-based sensor index. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_idx` a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0.

Output: Fan speed in rpms as integer

Exceptions that can be thrown by `amdsmi_get_gpu_fan_rpms` function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            fan_rpm = amdsmi_get_gpu_fan_rpms(device, 0)
            print(fan_rpm)
except AmdSmiException as e:
    print(e)
```

7.1.46 amdsmi_get_gpu_fan_speed

Description: Get the fan speed for the specified device as a value relative to `AMDSMI_MAX_FAN_SPEED`. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device

- `sensor_idx` a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0.

Output: Fan speed in relative to MAX

Exceptions that can be thrown by `amdsmi_get_gpu_fan_speed` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            fan_speed = amdsmi_get_gpu_fan_speed(device, 0)
            print(fan_speed)
except AmdSmiException as e:
    print(e)
```

7.1.47 `amdsmi_get_gpu_fan_speed_max`

Description: Get the max fan speed of the device with provided device handle. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_idx` a 0-based sensor index. Normally, this will be 0. If a device has more than one sensor, it could be greater than 0.

Output: Max fan speed as integer

Exceptions that can be thrown by `amdsmi_get_gpu_fan_speed_max` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            max_fan_speed = amdsmi_get_gpu_fan_speed_max(device, 0)
            print(max_fan_speed)
except AmdSmiException as e:
    print(e)
```

7.1.48 amdsmi_is_gpu_power_management_enabled

Description: Returns is power management enabled

Input parameters:

- `processor_handle` GPU device which to query

Output: Bool true if power management enabled else false

Exceptions that can be thrown by `amdsmi_is_gpu_power_management_enabled` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for processor in devices:
            is_power_management_enabled = amdsmi_is_gpu_power_management_
↳ enabled(processor)
            print(is_power_management_enabled)
except AmdSmiException as e:
    print(e)
```

7.1.49 amdsmi_get_temp_metric

Description: Get the temperature metric value for the specified metric, from the specified temperature sensor on the specified device. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_type` part of device from which temperature should be obtained
- `metric` enum indicated which temperature value should be retrieved

Output: Temperature as integer in millidegrees Celcius

Exceptions that can be thrown by `amdsmi_get_temp_metric` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
```

(continues on next page)

(continued from previous page)

```

        temp_metric = amdsmi_get_temp_metric(device, AmdSmiTemperatureType.EDGE,
                                             AmdSmiTemperatureMetric.CURRENT)
        print(temp_metric)
except AmdSmiException as e:
    print(e)

```

7.1.50 amdsmi_get_gpu_volt_metric

Description: Get the voltage metric value for the specified metric, from the specified voltage sensor on the specified device. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_type` part of device from which voltage should be obtained
- `metric` enum indicated which voltage value should be retrieved

Output: Voltage as integer in millivolts

Exceptions that can be thrown by `amdsmi_get_gpu_volt_metric` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            voltage = amdsmi_get_gpu_volt_metric(device, AmdSmiVoltageType.VDDGFX,
                                                AmdSmiVoltageMetric.AVERAGE)
            print(voltage)
except AmdSmiException as e:
    print(e)

```

7.1.51 amdsmi_get_utilization_count

Description: Get coarse/fine grain utilization counter of the specified device

Input parameters:

- `processor_handle` handle for the given device
- `counter_types` List of `AmdSmiUtilizationCounterType` counters requested

Output: List containing dictionaries with fields

| Field | Description |
|-----------------------------|---|
| timestamp | The timestamp when the counter is retrieved - Resolution: 1 ns |
| Dictionary for each counter | Subfield DescriptiontypeCounter that was requestedvalueValue gotten for utilization counter |

Exceptions that can be thrown by `amdsmi_get_utilization_count` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            utilization = amdsmi_get_utilization_count(
                device,
                AmdSmiUtilizationCounterType.COARSE_GRAIN_GFX_ACTIVITY
            )
            print(utilization)
            utilization = amdsmi_get_utilization_count(
                device,
                [AmdSmiUtilizationCounterType.COARSE_GRAIN_GFX_ACTIVITY,
                AmdSmiUtilizationCounterType.COARSE_GRAIN_MEM_ACTIVITY,
                AmdSmiUtilizationCounterType.COARSE_DECODER_ACTIVITY,
                AmdSmiUtilizationCounterType.FINE_GRAIN_GFX_ACTIVITY,
                AmdSmiUtilizationCounterType.FINE_GRAIN_MEM_ACTIVITY,
                AmdSmiUtilizationCounterType.FINE_DECODER_ACTIVITY]
            )
            print(utilization)
except AmdSmiException as e:
    print(e)
```

7.1.52 `amdsmi_get_gpu_perf_level`

Description: Get the performance level of the device with provided device handle. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device

Output: Performance level as enum value of `dev_perf_level_t`

Exceptions that can be thrown by `amdsmi_get_gpu_perf_level` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            perf_level = amdsmi_get_gpu_perf_level(dev)
            print(perf_level)
except AmdSmiException as e:
    print(e)
```

7.1.53 amdsmi_set_gpu_perf_determinism_mode

Description: Enter performance determinism mode with provided device handle. It is not supported on virtual machine guest

Input parameters:

- processor_handle handle for the given device
- clkvalue softmax value for GFXCLK in MHz

Output: None

Exceptions that can be thrown by amdsmi_set_gpu_perf_determinism_mode function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_perf_determinism_mode(device, 1333)
except AmdSmiException as e:
    print(e)
```

7.1.54 amdsmi_get_gpu_process_isolation

Description: Get the status of the Process Isolation

Input parameters:

- processor_handle handle for the given device

Output: integer corresponding to isolation_status; 0 - disabled, 1 - enabled

Exceptions that can be thrown by amdsmi_get_gpu_process_isolation function:

- AmdSmiLibraryException
- AmdSmiRetryException

- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            isolate = amdsmi_get_gpu_process_isolation(device)
            print("Process Isolation Status: ", isolate)
except AmdSmiException as e:
    print(e)
```

7.1.55 amdsmi_set_gpu_process_isolation

Description: Enable/disable the system Process Isolation for the given device handle.

Input parameters:

- processor_handle handle for the given device
- pisolate the process isolation status to set. 0 is the process isolation disabled, and 1 is the process isolation enabled.

Output: None

Exceptions that can be thrown by amdsmi_set_gpu_process_isolation function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_process_isolation(device, 1)
except AmdSmiException as e:
    print(e)
```

7.1.56 amdsmi_clean_gpu_local_data

Description: Clear the SRAM data of the given device. This can be called between user logins to prevent information leak.

Input parameters:

- processor_handle handle for the given device

Output: None

Exceptions that can be thrown by amdsmi_clean_gpu_local_data function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_clean_gpu_local_data(device)
except AmdSmiException as e:
    print(e)
```

7.1.57 amdsmi_get_gpu_overdrive_level

Description: Get the overdrive percent associated with the device with provided device handle. It is not supported on virtual machine guest

Input parameters:

- processor_handle handle for the given device

Output: Overdrive percentage as integer

Exceptions that can be thrown by amdsmi_get_gpu_overdrive_level function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            od_level = amdsmi_get_gpu_overdrive_level(dev)
            print(od_level)
except AmdSmiException as e:
    print(e)
```

7.1.58 amdsmi_get_gpu_mem_overdrive_level

Description: Get the GPU memory clock overdrive percent associated with the device with provided device handle. It is not supported on virtual machine guest

Input parameters:

- processor_handle handle for the given device

Output: Overdrive percentage as integer

Exceptions that can be thrown by `amdsmi_get_gpu_mem_overdrive_level` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            od_level = amdsmi_get_gpu_mem_overdrive_level(dev)
            print(od_level)
except AmdSmiException as e:
    print(e)
```

7.1.59 `amdsmi_get_clk_freq`

Description: Get the list of possible system clock speeds of device for a specified clock type. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `clk_type` the type of clock for which the frequency is desired

Output: Dictionary with fields

| Field | Description |
|----------------------------|--|
| <code>num_supported</code> | The number of supported frequencies |
| <code>current</code> | The current frequency index |
| <code>frequency</code> | List of frequencies, only the first <code>num_supported</code> frequencies are valid |

Exceptions that can be thrown by `amdsmi_get_clk_freq` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_get_clk_freq(device, AmdSmiClkType.SYS)
```

(continues on next page)

(continued from previous page)

```
except AmdSmiException as e:
    print(e)
```

7.1.60 amdsmi_get_gpu_od_volt_info

Description: This function retrieves the voltage/frequency curve information. If the num_regions is 0 then the voltage curve is not supported. It is not supported on virtual machine guest.

Input parameters:

- processor_handle handle for the given device

Output: Dictionary with fields

| Field | Description |
|---------------------|--|
| curr_sclk_range | Subfield Descriptionlower_boundlower bound sclk rangeupper_boundupper bound sclk range |
| curr_mclk_range | Subfield Descriptionlower_boundlower bound mclk rangeupper_boundupper bound mclk range |
| sclk_freq_limit | Subfield Descriptionlower_boundlower bound sclk range limitupper_boundupper bound sclk range limit |
| mclk_freq_limit | Subfield Descriptionlower_boundlower bound mclk range limitupper_boundupper bound mclk range limit |
| curve. vc_points | List of voltage curve points |
| num_regions | The number of voltage curve regions |

Exceptions that can be thrown by amdsmi_get_gpu_od_volt_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_get_gpu_od_volt_info(dev)
except AmdSmiException as e:
    print(e)
```

7.1.61 amdsmi_get_gpu_metrics_info

Description: This function retrieves the gpu metrics information. It is not supported on virtual machine guest

Input parameters:

- processor_handle handle for the given device

Output: Dictionary with fields

| Field | Description |
|-----------------------------|--|
| temperature_edge | Edge temperature value |
| temperature_hotspot | Hotspot (aka junction) temperature value |
| temperature_mem | Memory temperature value |
| temperature_vrgfx | vrgfx temperature value |
| temperature_vrsoc | vrsoc temperature value |
| temperature_vrmem | vrmem temperature value |
| average_gfx_activity | Average gfx activity |
| average_umc_activity | Average umc (Universal Memory Controller) activity |
| average_mm_activity | Average mm (multimedia) engine activity |
| average_socket_power | Average socket power |
| energy_accumulator | Energy accumulated with a 15.3 uJ resolution over 1ns |
| system_clock_counter | System clock counter |
| average_gfxclk_frequency | Average gfx clock frequency |
| average_socclk_frequency | Average soc clock frequency |
| average_uclk_frequency | Average uclk frequency |
| average_vclk0_frequency | Average vclk0 frequency |
| average_dclk0_frequency | Average dclk0 frequency |
| average_vclk1_frequency | Average vclk1 frequency |
| average_dclk1_frequency | Average dclk1 frequency |
| current_gfxclk | Current gfx clock |
| current_socclk | Current soc clock |
| current_uclk | Current uclk |
| current_vclk0 | Current vclk0 |
| current_dclk0 | Current dclk0 |
| current_vclk1 | Current vclk1 |
| current_dclk1 | Current dclk1 |
| throttle_status | Current throttle status |
| current_fan_speed | Current fan speed |
| pcie_link_width | PCIe link width (number of lanes) |
| pcie_link_speed | PCIe link speed in 0.1 GT/s (Giga Transfers per second) |
| padding | padding |
| gfx_activity_acc | gfx activity accumulated |
| mem_activity_acc | Memory activity accumulated |
| temperature_hbm | list of hbm temperatures |
| firmware_timestamp | timestamp from PMFW (10ns resolution) |
| voltage_soc | soc voltage |
| voltage_gfx | gfx voltage |
| voltage_mem | mem voltage |
| indep_throttle_status | ASIC independent throttle status (see drivers/gpu/drm/amd/pm/swsmu/inc/amdgpu_smu.h for b |
| current_socket_power | Current socket power (also known as instant socket power) |
| vcn_activity | List of VCN encode/decode engine utilization per AID |
| gfxclk_lock_status | Clock lock status. Bits 0:7 correspond to each gfx clock engine instance. Bits 0:5 for APU/AID |
| xgmi_link_width | XGMI bus width |
| xgmi_link_speed | XGMI bitrate |
| pcie_bandwidth_acc | PCIe accumulated bandwidth |
| pcie_bandwidth_inst | PCIe instantaneous bandwidth |
| pcie_l0_to_recov_count_acc | PCIe L0 to recovery state transition accumulated count |
| pcie_replay_count_acc | PCIe replay accumulated count |
| pcie_replay_rover_count_acc | PCIe replay rollover accumulated count |

cont

Table 7.1 – continued from previous page

| Field | Description |
|-------------------------|---|
| xgmi_read_data_acc | XGMI accumulated read data transfer size (KiloBytes) |
| xgmi_write_data_acc | XGMI accumulated write data transfer size (KiloBytes) |
| current_gfxclks | List of current gfx clock frequencies |
| current_socclks | List of current soc clock frequencies |
| current_vclk0s | List of current v0 clock frequencies |
| current_dclk0s | List of current d0 clock frequencies |
| pcie_nak_sent_count_acc | PCIe NAC sent count accumulated |
| pcie_nak_rcvd_count_acc | PCIe NAC received count accumulated |
| jpeg_activity | List of JPEG engine activity |

Exceptions that can be thrown by `amdsmi_get_gpu_metrics_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_get_gpu_metrics_info(dev)
except AmdSmiException as e:
    print(e)
```

7.1.62 `amdsmi_get_gpu_pm_metrics_info`

Description: This function will retrieve the name and value for each item in the pm metrics table with the given processor handle.

Input parameters:

- `processor_handle` handle for the given device

Output: List containing dictionaries of pm metrics and their values

| Field | Description |
|-------|--------------------|
| name | name of PM metric |
| value | value of pm metric |

Exceptions that can be thrown by `amdsmi_get_gpu_pm_metrics_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            print(amdsmi_get_gpu_pm_metrics_info(device))
except AmdSmiException as e:
    print(e)

```

7.1.63 amdsmi_get_gpu_reg_table_info

Description: This function will retrieve register metrics table with provided device index and register type.

Input parameters:

- `processor_handle` handle for the given device
- `reg_type` register type

Output: List containing dictionaries of register metrics and their values

| Field | Description |
|--------------------|--------------------------|
| <code>name</code> | name of register metric |
| <code>value</code> | value of register metric |

Exceptions that can be thrown by `amdsmi_get_gpu_reg_table_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            print(amdsmi_get_gpu_reg_table_info(device, AmdSmiRegType.PCIE))
except AmdSmiException as e:
    print(e)

```

7.1.64 amdsmi_get_gpu_od_volt_curve_regions

Description: This function will retrieve the current valid regions in the frequency/voltage space. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `num_regions` number of freq volt regions

Output: List containing a dictionary with fields for each freq volt region

| Field | Description |
|------------|--|
| freq_range | Subfield Descriptionlower_boundlower bound freq rangeupper_boundupper bound freq range |
| volt_range | Subfield Descriptionlower_boundlower bound volt rangeupper_boundupper bound volt range |

Exceptions that can be thrown by `amdsmi_get_gpu_od_volt_curve_regions` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_get_gpu_od_volt_curve_regions(device, 3)
except AmdSmiException as e:
    print(e)
```

7.1.65 amdsmi_get_gpu_power_profile_presets

Description: Get the list of available preset power profiles and an indication of which profile is currently active. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_idx` number of freq volt regions

Output: Dictionary with fields

| Field | Description |
|---------------------------------|---|
| <code>available_profiles</code> | Which profiles are supported by this system |
| <code>current</code> | Which power profile is currently active |
| <code>num_profiles</code> | How many power profiles are available |

Exceptions that can be thrown by `amdsmi_get_gpu_power_profile_presets` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
```

(continues on next page)

(continued from previous page)

```
if len(devices) == 0:
    print("No GPUs on machine")
else:
    for device in devices:
        amdsmi_get_gpu_power_profile_presets(device, 0)
except AmdSmiException as e:
    print(e)
```

7.1.66 amdsmi_gpu_counter_group_supported

Description: Tell if an event group is supported by a given device. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` device which to query
- `event_group` event group being checked for support

Output: None

Exceptions that can be thrown by `amdsmi_gpu_counter_group_supported` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_gpu_counter_group_supported(device, AmdSmiEventGroup.XGMI)
except AmdSmiException as e:
    print(e)
```

7.1.67 amdsmi_gpu_create_counter

Description: Creates a performance counter object

Input parameters:

- `processor_handle` device which to query
- `event_type` event group being checked for support

Output: An event handle of the newly created performance counter object

Exceptions that can be thrown by `amdsmi_gpu_create_counter` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            event_handle = amdsmi_gpu_create_counter(device, AmdSmiEventGroup.XGMI)
except AmdSmiException as e:
    print(e)

```

7.1.68 amdsmi_gpu_destroy_counter

Description: Destroys a performance counter object

Input parameters:

- `event_handle` event handle of the performance counter object

Output: None

Exceptions that can be thrown by `amdsmi_gpu_destroy_counter` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            event_handle = amdsmi_gpu_create_counter(device, AmdSmiEventGroup.XGMI)
            amdsmi_gpu_destroy_counter(event_handle)
except AmdSmiException as e:
    print(e)

```

7.1.69 amdsmi_gpu_control_counter

Description: Issue performance counter control commands. It is not supported on virtual machine guest

Input parameters:

- `event_handle` event handle of the performance counter object
- `counter_command` command being passed to counter as `AmdSmiCounterCommand`

Output: None

Exceptions that can be thrown by `amdsmi_gpu_control_counter` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            event_handle = amdsmi_gpu_create_counter(device, AmdSmiEventType.XGMI_1_
↳REQUEST_TX)
            amdsmi_gpu_control_counter(event_handle, AmdSmiCounterCommand.CMD_START)
except AmdSmiException as e:
    print(e)
```

7.1.70 amdsmi_gpu_read_counter

Description: Read the current value of a performance counter

Input parameters:

- event_handle event handle of the performance counter object

Output: Dictionary with fields

| Field | Description |
|--------------|--|
| value | Counter value |
| time_enabled | Time that the counter was enabled in nanoseconds |
| time_running | Time that the counter was running in nanoseconds |

Exceptions that can be thrown by amdsmi_gpu_read_counter function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            event_handle = amdsmi_gpu_create_counter(device, AmdSmiEventType.XGMI_1_
↳REQUEST_TX)
            amdsmi_gpu_read_counter(event_handle)
except AmdSmiException as e:
    print(e)
```

7.1.71 amdsmi_get_gpu_available_counters

Description: Get the number of currently available counters. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `event_group` event group being checked as `AmdSmiEventGroup`

Output: Number of available counters for the given device of the inputted event group

Exceptions that can be thrown by `amdsmi_get_gpu_available_counters` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            available_counters = amdsmi_get_gpu_available_counters(device, ↵
↵AmdSmiEventGroup.XGMI)
            print(available_counters)
except AmdSmiException as e:
    print(e)
```

7.1.72 amdsmi_set_gpu_perf_level

Description: Set a desired performance level for given device. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `perf_level` performance level being set as `AmdSmiDevPerfLevel`

Output: None

Exceptions that can be thrown by `amdsmi_set_gpu_perf_level` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_perf_level(device, AmdSmiDevPerfLevel.STABLE_PEAK)
```

(continues on next page)

```
except AmdSmiException as e:  
    print(e)
```

7.1.73 amdsmi_reset_gpu

Description: Reset the gpu associated with the device with provided device handle It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device

Output: None

Exceptions that can be thrown by `amdsmi_reset_gpu` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:  
    devices = amdsmi_get_processor_handles()  
    if len(devices) == 0:  
        print("No GPUs on machine")  
    else:  
        for device in devices:  
            amdsmi_reset_gpu(device)  
except AmdSmiException as e:  
    print(e)
```

7.1.74 amdsmi_set_gpu_fan_speed

Description: Set the fan speed for the specified device with the provided speed, in RPMs. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_idx` sensor index as integer
- `fan_speed` the speed to which the function will attempt to set the fan

Output: None

Exceptions that can be thrown by `amdsmi_set_gpu_fan_speed` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_fan_speed(device, 0, 1333)
except AmdSmiException as e:
    print(e)

```

7.1.75 amdsmi_reset_gpu_fan

Description: Reset the fan to automatic driver control. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `sensor_idx` sensor index as integer

Output: None

Exceptions that can be thrown by `amdsmi_reset_gpu_fan` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_reset_gpu_fan(device, 0)
except AmdSmiException as e:
    print(e)

```

7.1.76 amdsmi_set_clk_freq

Description: Control the set of allowed frequencies that can be used for the specified clock. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `clk_type` the type of clock for which the set of frequencies will be modified as `AmdSmiClkType`
- `freq_bitmask` bitmask indicating the indices of the frequencies that are to be enabled (1) and disabled (0). Only the lowest `::amdsmi_frequencies_t.num_supported` bits of this mask are relevant.

Output: None

Exceptions that can be thrown by `amdsmi_set_clk_freq` function:

- `AmdSmiLibraryException`

- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            freq_bitmask = 0
            amdsmi_set_clk_freq(device, AmdSmiClkType.GFX, freq_bitmask)
except AmdSmiException as e:
    print(e)
```

7.1.77 amdsmi_get_soc_pstate

Description: Get dpm policy information.

Input parameters:

- processor_handle handle for the given device
- policy_id the policy id to set.

Output: Dictionary with fields

| Field | Description |
|---------------|---|
| num_supported | total number of supported policies |
| current_id | current policy id |
| policies | list of dictionaries containing possible policies |

Exceptions that can be thrown by amdsmi_get_soc_pstate function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            dpm_policies = amdsmi_get_soc_pstate(device)
            print(dpm_policies)
except AmdSmiException as e:
    print(e)
```

7.1.78 amdsmi_set_soc_pstate

Description: Set the dpm policy to corresponding policy_id. Typically following: 0(default),1,2,3

Input parameters:

- processor_handle handle for the given device
- policy_id the policy id to set.

Output: None

Exceptions that can be thrown by amdsmi_set_soc_pstate function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_soc_pstate(device, 0)
except AmdSmiException as e:
    print(e)
```

7.1.79 amdsmi_set_xgmi_plpd

Description: Set the xgmi per-link power down policy parameter for the processor

Input parameters:

- processor_handle handle for the given device
- policy_id the xgmi plpd id to set.

Output: None

Exceptions that can be thrown by amdsmi_set_xgmi_plpd function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_xgmi_plpd(device, 0)
except AmdSmiException as e:
    print(e)
```

7.1.80 amdsmi_get_xgmi_plpd

Description: Get the xgmi per-link power down policy parameter for the processor

Input parameters:

- `processor_handle` handle for the given device

Output: Dict containing information about xgmi per-link power down policy

| Field | Description |
|----------------------------|----------------------------------|
| <code>num_supported</code> | The number of supported policies |
| <code>current_id</code> | The current policy index |
| <code>plpds</code> | List of policies. |

Exceptions that can be thrown by `amdsmi_get_xgmi_plpd` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            xgmi_plpd = amdsmi_get_xgmi_plpd(device)
            print(xgmi_plpd)
except AmdSmiException as e:
    print(e)
```

7.1.81 amdsmi_set_gpu_overdrive_level

Description: **deprecated** Set the overdrive percent associated with the device with provided device handle with the provided value. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `overdrive_value` value to which the overdrive level should be set

Output: None

Exceptions that can be thrown by `amdsmi_set_gpu_overdrive_level` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_overdrive_level(device, 0)
except AmdSmiException as e:
    print(e)

```

7.1.82 amdsmi_get_gpu_ecc_count

Description: Retrieve the error counts for a GPU block. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `block` The block for which error counts should be retrieved

Output: Dict containing information about error counts

| Field | Description |
|----------------------------------|-------------------------------|
| <code>correctable_count</code> | Count of correctable errors |
| <code>uncorrectable_count</code> | Count of uncorrectable errors |
| <code>deferred_count</code> | Count of deferred errors |

Exceptions that can be thrown by `amdsmi_get_gpu_ecc_count` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            ecc_count = amdsmi_get_gpu_ecc_count(device, AmdSmiGpuBlock.UMC)
            print(ecc_count)
except AmdSmiException as e:
    print(e)

```

7.1.83 amdsmi_get_gpu_ecc_enabled

Description: Retrieve the enabled ECC bit-mask. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device

Output: Enabled ECC bit-mask

Exceptions that can be thrown by `amdsmi_get_gpu_ecc_enabled` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            enabled = amdsmi_get_gpu_ecc_enabled(device)
            print(enabled)
except AmdSmiException as e:
    print(e)
```

7.1.84 `amdsmi_get_gpu_ecc_status`

Description: Retrieve the ECC status for a GPU block. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device
- `block` The block for which ECC status should be retrieved

Output: ECC status for a requested GPU block

Exceptions that can be thrown by `amdsmi_get_gpu_ecc_status` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            status = amdsmi_get_gpu_ecc_status(device, AmdSmiGpuBlock.UMC)
            print(status)
except AmdSmiException as e:
    print(e)
```

7.1.85 amdsmi_status_code_to_string

Description: Get a description of a provided AMDSMI error status

Input parameters:

- `status` The error status for which a description is desired

Output: String description of the provided error code

Exceptions that can be thrown by `amdsmi_status_code_to_string` function:

- `AmdSmiParameterException`

Example:

```
try:
    status_str = amdsmi_status_code_to_string(ctypes.c_uint32(0))
    print(status_str)
except AmdSmiException as e:
    print(e)
```

7.1.86 amdsmi_get_gpu_compute_process_info

Description: Get process information about processes currently using GPU

Input parameters: None

Output: List of python dicts each containing a process information

| Field | Description |
|---------------------------|--------------------------------|
| <code>process_id</code> | Process ID |
| <code>pasid</code> | PASID |
| <code>vram_usage</code> | VRAM usage |
| <code>sdma_usage</code> | SDMA usage in microseconds |
| <code>cu_occupancy</code> | Compute Unit usage in percents |

Exceptions that can be thrown by `amdsmi_get_gpu_compute_process_info` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`

Example:

```
try:
    procs = amdsmi_get_gpu_compute_process_info()
    for proc in procs:
        print(proc)
except AmdSmiException as e:
    print(e)
```

7.1.87 amdsmi_get_gpu_compute_process_info_by_pid

Description: Get process information about processes currently using GPU

Input parameters:

- `pid` The process ID for which process information is being requested

Output: Dict containing a process information

| Field | Description |
|--------------|--------------------------------|
| process_id | Process ID |
| pasid | PASID |
| vram_usage | VRAM usage |
| sdma_usage | SDMA usage in microseconds |
| cu_occupancy | Compute Unit usage in percents |

Exceptions that can be thrown by `amdsmi_get_gpu_compute_process_info_by_pid` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    pid = 0 # << valid pid here
    proc = amdsmi_get_gpu_compute_process_info_by_pid(pid)
    print(proc)
except AmdSmiException as e:
    print(e)
```

7.1.88 `amdsmi_get_gpu_compute_process_gpus`

Description: Get the device indices currently being used by a process

Input parameters:

- `pid` The process id of the process for which the number of gpus currently being used is requested

Output: List of indices of devices currently being used by the process

Exceptions that can be thrown by `amdsmi_get_gpu_compute_process_gpus` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    pid = 0 # << valid pid here
    indices = amdsmi_get_gpu_compute_process_gpus(pid)
    print(indices)
except AmdSmiException as e:
    print(e)
```

7.1.89 `amdsmi_gpu_xgmi_error_status`

Description: Retrieve the XGMI error status for a device. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device

Output: XGMI error status for a requested device

Exceptions that can be thrown by `amdsmi_gpu_xgmi_error_status` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            status = amdsmi_gpu_xgmi_error_status(device)
            print(status)
except AmdSmiException as e:
    print(e)
```

7.1.90 `amdsmi_reset_gpu_xgmi_error`

Description: Reset the XGMI error status for a device. It is not supported on virtual machine guest

Input parameters:

- `processor_handle` handle for the given device

Output: None

Exceptions that can be thrown by `amdsmi_reset_gpu_xgmi_error` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_reset_gpu_xgmi_error(device)
except AmdSmiException as e:
    print(e)
```

7.1.91 `amdsmi_get_gpu_vendor_name`

Description: Returns the device vendor name

Input parameters:

- `processor_handle` device which to query

Output: device vendor name

Exceptions that can be thrown by `amdsmi_get_gpu_vendor_name` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vendor_name = amdsmi_get_gpu_vendor_name(device)
            print(vendor_name)
except AmdSmiException as e:
    print(e)
```

7.1.92 `amdsmi_get_gpu_id`

Description: Get the device id associated with the device with provided device handler

Input parameters:

- `processor_handle` device which to query

Output: device id

Exceptions that can be thrown by `amdsmi_get_gpu_id` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            dev_id = amdsmi_get_gpu_id(device)
            print(dev_id)
except AmdSmiException as e:
    print(e)
```

7.1.93 `amdsmi_get_gpu_vram_vendor`

Description: Get the vram vendor string of a gpu device.

Input parameters:

- `processor_handle` device which to query

Output: vram vendor

Exceptions that can be thrown by `amdsmi_get_gpu_vram_vendor` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            vram_vendor = amdsmi_get_gpu_vram_vendor(device)
            print(vram_vendor)
except AmdSmiException as e:
    print(e)
```

7.1.94 `amdsmi_get_gpu_subsystem_id`

Description: Get the subsystem device id associated with the device with provided device handle.

Input parameters:

- `processor_handle` device which to query

Output: subsystem device id

Exceptions that can be thrown by `amdsmi_get_gpu_subsystem_id` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            id = amdsmi_get_gpu_subsystem_id(device)
            print(id)
except AmdSmiException as e:
    print(e)
```

7.1.95 `amdsmi_get_gpu_subsystem_name`

Description: Get the name string for the device subsystem

Input parameters:

- `processor_handle` device which to query

Output: device subsystem

Exceptions that can be thrown by `amdsmi_get_gpu_subsystem_name` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            subsystem_nam = amdsmi_get_gpu_subsystem_name(device)
            print(subsystem_nam)
except AmdSmiException as e:
    print(e)
```

7.1.96 `amdsmi_get_lib_version`

Description: Get the build version information for the currently running build of AMDSMI.

Output: `amdsmi` build version

Exceptions that can be thrown by `amdsmi_get_lib_version` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    version = amdsmi_get_lib_version()
    print(version)
except AmdSmiException as e:
    print(e)
```

7.1.97 `amdsmi_topo_get_numa_node_number`

Description: Retrieve the NUMA CPU node number for a device

Input parameters:

- `processor_handle` device which to query

Output: node number of NUMA CPU for the device

Exceptions that can be thrown by `amdsmi_topo_get_numa_node_number` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            node_number = amdsmi_topo_get_numa_node_number()
            print(node_number)
except AmdSmiException as e:
    print(e)
```

7.1.98 amdsmi_topo_get_link_weight

Description: Retrieve the weight for a connection between 2 GPUs.

Input parameters:

- `processor_handle_src` the source device handle
- `processor_handle_dest` the destination device handle

Output: the weight for a connection between 2 GPUs

Exceptions that can be thrown by `amdsmi_topo_get_link_weight` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        processor_handle_src = devices[0]
        processor_handle_dest = devices[1]
        weight = amdsmi_topo_get_link_weight(processor_handle_src, processor_handle_dest)
        print(weight)
except AmdSmiException as e:
    print(e)
```

7.1.99 amdsmi_get_minmax_bandwidth_between_processors

Description: Retrieve minimal and maximal io link bandwidth between 2 GPUs.

Input parameters:

- `processor_handle_src` the source device handle
- `processor_handle_dest` the destination device handle

Output: Dictionary with fields:

| Field | Description |
|---------------|--------------------------------------|
| min_bandwidth | minimal bandwidth for the connection |
| max_bandwidth | maximal bandwidth for the connection |

Exceptions that can be thrown by `amdsmi_get_minmax_bandwidth_between_processors` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        processor_handle_src = devices[0]
        processor_handle_dest = devices[1]
        bandwidth = amdsmi_get_minmax_bandwidth_between_processors(processor_handle_src,
↪ processor_handle_dest)
        print(bandwidth['min_bandwidth'])
        print(bandwidth['max_bandwidth'])
except AmdSmiException as e:
    print(e)
```

7.1.100 amdsmi_topo_get_link_type

Description: Retrieve the hops and the connection type between 2 GPUs

Input parameters:

- `processor_handle_src` the source device handle
- `processor_handle_dest` the destination device handle

Output: Dictionary with fields:

| Field | Description |
|-------|---------------------|
| hops | number of hops |
| type | the connection type |

Exceptions that can be thrown by `amdsmi_topo_get_link_type` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        processor_handle_src = devices[0]
        processor_handle_dest = devices[1]
        link_type = amdsmi_topo_get_link_type(processor_handle_src, processor_handle_
↪dest)
        print(link_type['hops'])
        print(link_type['type'])
except AmdSmiException as e:
    print(e)

```

7.1.101 amdsmi_get_P2P_status

Description: Retrieve the connection type and P2P capabilities between 2 GPUs

Input parameters:

- processor_handle_src the source device handle
- processor_handle_dest the destination device handle

Output: Dictionary with fields:

| Field | Description |
|-------|--|
| type | AmdSmiIoLinkType |
| cap | Subfield Description is_iolink_coherent1 == True; 0 == False; Uint_max = Undefined is_iolink_atomics_32bitSupports 32bit atomics is_iolink_atomics_64bitSupports 64bit atomics is_iolink_dmaSupports DMA is_iolink_bi_directionalIs the IOLink Bidirectional |

Exceptions that can be thrown by amdsmi_get_P2P_status function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        processor_handle_src = devices[0]
        processor_handle_dest = devices[1]
        link_type = amdsmi_get_P2P_status(processor_handle_src, processor_handle_dest)
        print(link_type['type'])
        print(link_type['caps'])
except AmdSmiException as e:
    print(e)

```

7.1.102 amdsmi_is_P2P_accessible

Description: Return P2P availability status between 2 GPUs

Input parameters:

- `processor_handle_src` the source device handle
- `processor_handle_dest` the destination device handle

Output: P2P availability status between 2 GPUs

Exceptions that can be thrown by `amdsmi_is_P2P_accessible` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        processor_handle_src = devices[0]
        processor_handle_dest = devices[1]
        accessible = amdsmi_is_P2P_accessible(processor_handle_src, processor_handle_
↪dest)
        print(accessible)
except AmdSmiException as e:
    print(e)
```

7.1.103 amdsmi_get_gpu_compute_partition

Description: Get the compute partition from the given GPU

Input parameters:

- `processor_handle` the device handle

Output: String of the partition type

Exceptions that can be thrown by `amdsmi_get_gpu_compute_partition` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            compute_partition_type = amdsmi_get_gpu_compute_partition(device)
```

(continues on next page)

(continued from previous page)

```

        print(compute_partition_type)
except AmdSmiException as e:
    print(e)

```

7.1.104 amdsmi_set_gpu_compute_partition

Description: Set the compute partition to the given GPU

Input parameters:

- `processor_handle` the device handle
- `compute_partition` the type of compute_partition to set

Output: String of the partition type

Exceptions that can be thrown by `amdsmi_set_gpu_compute_partition` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    compute_partition = AmdSmiComputePartitionType.SPX
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_compute_partition(device, compute_partition)
except AmdSmiException as e:
    print(e)

```

7.1.105 amdsmi_get_gpu_memory_partition

Description: Get the memory partition from the given GPU

Input parameters:

- `processor_handle` the device handle

Output: String of the partition type

Exceptions that can be thrown by `amdsmi_get_gpu_memory_partition` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:

```

(continues on next page)

(continued from previous page)

```

        print("No GPUs on machine")
    else:
        for device in devices:
            memory_partition_type = amdsmi_get_gpu_memory_partition(device)
            print(memory_partition_type)
except AmdSmiException as e:
    print(e)

```

7.1.106 amdsmi_set_gpu_memory_partition

Description: Set the memory partition to the given GPU

Input parameters:

- `processor_handle` the device handle
- `memory_partition` the type of memory_partition to set

Output: String of the partition type

Exceptions that can be thrown by `amdsmi_set_gpu_memory_partition` function:

- `AmdSmiLibraryException`
- `AmdSmiRetryException`
- `AmdSmiParameterException`

Example:

```

try:
    memory_partition = AmdSmiMemoryPartitionType.NPS1
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            amdsmi_set_gpu_memory_partition(device, memory_partition)
except AmdSmiException as e:
    print(e)

```

7.1.107 amdsmi_get_gpu_accelerator_partition_profile

Note: CURRENTLY HARDCODED TO RETURN EMPTY VALUES

Description: Get partition information for target device

Input parameters:

- `processor_handle` the device handle

Output: Dictionary with fields:

| Field | Description |
|--------------------------------|---|
| <code>partition_id</code> | ID of the partition on the GPU provided |
| <code>partition_profile</code> | Dict containing partition data (TBD) |

Exceptions that can be thrown by `amdsmi_get_gpu_accelerator_partition_profile` function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            partition_id = amdsmi_get_gpu_accelerator_partition_profile(device)[
↪ "partition_id"]
            print(partition_id)
except AmdSmiException as e:
    print(e)
```

7.1.108 amdsmi_get_xgmi_info

Description: Returns XGMI information for the GPU.

Input parameters:

- processor_handle device handle

Output: Dictionary with fields:

| Field | Description |
|--------------|--------------|
| xgmi_lanes | xgmi lanes |
| xgmi_hive_id | xgmi hive id |
| xgmi_node_id | xgmi node id |
| index | index |

Exceptions that can be thrown by amdsmi_get_xgmi_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs on machine")
    else:
        for device in devices:
            xgmi_info = amdsmi_get_xgmi_info(device)
            print(xgmi_info['xgmi_lanes'])
            print(xgmi_info['xgmi_hive_id'])
            print(xgmi_info['xgmi_node_id'])
            print(xgmi_info['index'])
```

(continues on next page)

```
except AmdSmiException as e:
    print(e)
```

7.1.109 amdsmi_get_link_topology_nearest

Description: Retrieve the set of GPUs that are nearest to a given device at a specific interconnectivity level.

Input parameters:

- `processor_handle` The identifier of the given device.
- `link_type` The `AmdSmiLinkType` level to search for nearest devices

Output: Dictionary holding the following fields.

- `count` number of nearest devices found based on given topology level
- `processor_list` list of all nearest device handlers found

Exceptions that can be thrown by `amdsmi_get_link_topology_nearest` function:

- `AmdSmiLibraryException`

Example:

```
try:
    amdsmi_init()

    devices = amdsmi_get_processor_handles()
    if len(devices) == 0:
        print("No GPUs found on machine")
        exit()
    else:
        print(amdsmi_get_gpu_device_uuid(devices[0]))

    nearest_gpus = amdsmi_get_link_topology_nearest(devices[0], AmdSmiLinkType.AMDSMI_
↪LINK_TYPE_PCIE)
    if (nearest_gpus['count']) == 0:
        print("No nearest GPUs found on machine")
    else:
        print("Nearest GPUs")
        for gpu in nearest_gpus['processor_list']:
            print(amdsmi_get_gpu_device_uuid(gpu))

except AmdSmiException as e:
    print(e)
finally:
    try:
        amdsmi_shut_down()
    except AmdSmiException as e:
        print(e)
```

7.2 CPU APIs

7.2.1 amdsmi_get_processor_info

Note: CURRENTLY HARDCODED TO RETURN EMPTY VALUES

Description: Return processor name

Input parameters: processor_handle processor handle

Output: Processor name

Exceptions that can be thrown by amdsmi_get_processor_info function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_processor_handles()
    if len(processor_handles) == 0:
        print("No processors on machine")
    else:
        for processor in processor_handles:
            print(amdsmi_get_processor_info(processor))
except AmdSmiException as e:
    print(e)
```

7.2.2 amdsmi_get_cpu_hsmv_proto_ver

Description: Get the hsmv protocol version.

Output: amdsmi hsmv protocol version

Exceptions that can be thrown by amdsmi_get_cpu_hsmv_proto_ver function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpsocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            version = amdsmi_get_cpu_hsmv_proto_ver(processor)
            print(version)
except AmdSmiException as e:
    print(e)
```

7.2.3 amdsmi_get_cpu_smu_fw_version

Description: Get the SMU Firmware version.

Output: amdsmi SMU Firmware version

Exceptions that can be thrown by amdsmi_get_cpu_smu_fw_version function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpsocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            version = amdsmi_get_cpu_smu_fw_version(processor)
            print(version['debug'])
            print(version['minor'])
            print(version['major'])
except AmdSmiException as e:
    print(e)
```

7.2.4 amdsmi_get_cpu_prochot_status

Description: Get the CPU's prochot status.

Output: amdsmi cpu prochot status

Exceptions that can be thrown by amdsmi_get_cpu_prochot_status function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpsocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            prochot = amdsmi_get_cpu_prochot_status(processor)
            print(prochot)
except AmdSmiException as e:
    print(e)
```

7.2.5 amdsmi_get_cpu_fclk_mclk

Description: Get the Data fabric clock and Memory clock in MHz.

Output: amdsmi data fabric clock and memory clock

Exceptions that can be thrown by amdsmi_get_cpu_fclk_mclk function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpsocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            clk = amdsmi_get_cpu_fclk_mclk(processor)
```

(continues on next page)

(continued from previous page)

```

        for fclk, mclk in clk.items():
            print(fclk)
            print(mclk)
except AmdSmiException as e:
    print(e)

```

7.2.6 amdsmi_get_cpu_cclk_limit

Description: Get the core clock in MHz.

Output: amdsmi core clock

Exceptions that can be thrown by amdsmi_get_cpu_cclk_limit function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            cclk_limit = amdsmi_get_cpu_cclk_limit(processor)
            print(cclk_limit)
except AmdSmiException as e:
    print(e)

```

7.2.7 amdsmi_get_cpu_socket_current_active_freq_limit

Description: Get current active frequency limit of the socket.

Output: amdsmi frequency value in MHz and frequency source name

Exceptions that can be thrown by amdsmi_get_cpu_socket_current_active_freq_limit function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            freq_limit = amdsmi_get_cpu_socket_current_active_freq_limit(processor)
            for freq, src in freq_limit.items():
                print(freq)
                print(src)
except AmdSmiException as e:
    print(e)

```

7.2.8 amdsmi_get_cpu_socket_freq_range

Description: Get socket frequency range

Output: amdsmi maximum frequency and minimum frequency

Exceptions that can be thrown by `amdsmi_get_cpu_socket_freq_range` function:

- `AmdSmiLibraryException`

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            freq_range = amdsmi_get_cpu_socket_freq_range(processor)
            for fmax, fmin in freq_range.items():
                print(fmax)
                print(fmin)
except AmdSmiException as e:
    print(e)
```

7.2.9 amdsmi_get_cpu_core_current_freq_limit

Description: Get socket frequency limit of the core

Output: amdsmi frequency

Exceptions that can be thrown by `amdsmi_get_cpu_core_current_freq_limit` function:

- `AmdSmiLibraryException`

Example:

```
try:
    processor_handles = amdsmi_get_cpucore_handles()
    if len(processor_handles) == 0:
        print("No CPU cores on machine")
    else:
        for processor in processor_handles:
            freq_limit = amdsmi_get_cpu_core_current_freq_limit(processor)
            print(freq_limit)
except AmdSmiException as e:
    print(e)
```

7.2.10 amdsmi_get_cpu_socket_power

Description: Get the socket power.

Output: amdsmi socket power

Exceptions that can be thrown by `amdsmi_get_cpu_socket_power` function:

- `AmdSmiLibraryException`

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            sock_power = amdsmi_get_cpu_socket_power(processor)
            print(sock_power)
except AmdSmiException as e:
    print(e)

```

7.2.11 amdsmi_get_cpu_socket_power_cap

Description: Get the socket power cap.

Output: amdsmi socket power cap

Exceptions that can be thrown by amdsmi_get_cpu_socket_power_cap function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            sock_power = amdsmi_get_cpu_socket_power_cap(processor)
            print(sock_power)
except AmdSmiException as e:
    print(e)

```

7.2.12 amdsmi_get_cpu_socket_power_cap_max

Description: Get the socket power cap max.

Output: amdsmi socket power cap max

Exceptions that can be thrown by amdsmi_get_cpu_socket_power_cap_max function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            sock_power = amdsmi_get_cpu_socket_power_cap_max(processor)
            print(sock_power)
except AmdSmiException as e:
    print(e)

```

7.2.13 amdsmi_get_cpu_pwr_svi_telemetry_all_rails

Description: Get the SVI based power telemetry for all rails.

Output: amdsmi svi based power value

Exceptions that can be thrown by `amdsmi_get_cpu_pwr_svi_telemetry_all_rails` function:

- `AmdSmiLibraryException`

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            power = amdsmi_get_cpu_pwr_svi_telemetry_all_rails(processor)
            print(power)
except AmdSmiException as e:
    print(e)
```

7.2.14 amdsmi_set_cpu_socket_power_cap

Description: Set the power cap value for a given socket.

Input: amdsmi socket power cap value

Exceptions that can be thrown by `amdsmi_set_cpu_socket_power_cap` function:

- `AmdSmiLibraryException`

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            power = amdsmi_set_cpu_socket_power_cap(processor, 1000)
except AmdSmiException as e:
    print(e)
```

7.2.15 amdsmi_set_cpu_pwr_efficiency_mode

Description: Set the power efficiency profile policy.

Input: mode(0, 1, or 2)

Exceptions that can be thrown by `amdsmi_set_cpu_pwr_efficiency_mode` function:

- `AmdSmiLibraryException`

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
```

(continues on next page)

(continued from previous page)

```

if len(processor_handles) == 0:
    print("No CPU sockets on machine")
else:
    for processor in processor_handles:
        policy = amdsmi_set_cpu_pwr_efficiency_mode(processor, 0)
except AmdSmiException as e:
    print(e)

```

7.2.16 amdsmi_get_cpu_core_boostlimit

Description: Get boost limit of the cpu core

Output: amdsmi frequency

Exceptions that can be thrown by amdsmi_get_cpu_core_boostlimit function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpucore_handles()
    if len(processor_handles) == 0:
        print("No CPU cores on machine")
    else:
        for processor in processor_handles:
            boost_limit = amdsmi_get_cpu_core_boostlimit(processor)
            print(boost_limit)
except AmdSmiException as e:
    print(e)

```

7.2.17 amdsmi_get_cpu_socket_c0_residency

Description: Get the cpu socket C0 residency.

Output: amdsmi C0 residency value

Exceptions that can be thrown by amdsmi_get_cpu_socket_c0_residency function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            c0_residency = amdsmi_get_cpu_socket_c0_residency(processor)
            print(c0_residency)
except AmdSmiException as e:
    print(e)

```

7.2.18 amdsmi_set_cpu_core_boostlimit

Description: Set the cpu core boost limit.

Output: amdsmi boostlimit value

Exceptions that can be thrown by amdsmi_set_cpu_core_boostlimit function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpucore_handles()
    if len(processor_handles) == 0:
        print("No CPU cores on machine")
    else:
        for processor in processor_handles:
            boost_limit = amdsmi_set_cpu_core_boostlimit(processor, 1000)
except AmdSmiException as e:
    print(e)
```

7.2.19 amdsmi_set_cpu_socket_boostlimit

Description: Set the cpu socket boost limit.

Input: amdsmi boostlimit value

Exceptions that can be thrown by amdsmi_set_cpu_socket_boostlimit function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            boost_limit = amdsmi_set_cpu_socket_boostlimit(processor, 1000)
except AmdSmiException as e:
    print(e)
```

7.2.20 amdsmi_get_cpu_ddr_bw

Description: Get the CPU DDR Bandwidth.

Output: amdsmi ddr bandwidth data

Exceptions that can be thrown by amdsmi_get_cpu_ddr_bw function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
```

(continues on next page)

(continued from previous page)

```

        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            ddr_bw = amdsmi_get_cpu_ddr_bw(processor)
            print(dds_bw['max_bw'])
            print(dds_bw['utilized_bw'])
            print(dds_bw['utilized_pct'])
except AmdSmiException as e:
    print(e)

```

7.2.21 amdsmi_get_cpu_socket_temperature

Description: Get the socket temperature.

Output: amdsmi temperature value

Exceptions that can be thrown by amdsmi_get_cpu_socket_temperature function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            ptmon = amdsmi_get_cpu_socket_temperature(processor)
            print(ptmon)
except AmdSmiException as e:
    print(e)

```

7.2.22 amdsmi_get_cpu_dimm_temp_range_and_refresh_rate

Description: Get DIMM temperature range and refresh rate.

Output: amdsmi dimm metric data

Exceptions that can be thrown by amdsmi_get_cpu_dimm_temp_range_and_refresh_rate function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            dimm = amdsmi_get_cpu_dimm_temp_range_and_refresh_rate(processor)
            print(dimm['range'])
            print(dimm['ref_rate'])
except AmdSmiException as e:
    print(e)

```

7.2.23 amdsmi_get_cpu_dimm_power_consumption

Description: amdsmi_get_cpu_dimm_power_consumption.

Output: amdsmi dimm power consumption value

Exceptions that can be thrown by amdsmi_get_cpu_dimm_power_consumption function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            dimm = amdsmi_get_cpu_dimm_power_consumption(processor)
            print(dimm['power'])
            print(dimm['update_rate'])
            print(dimm['dimm_addr'])
except AmdSmiException as e:
    print(e)
```

7.2.24 amdsmi_get_cpu_dimm_thermal_sensor

Description: Get DIMM thermal sensor value.

Output: amdsmi dimm temperature data

Exceptions that can be thrown by amdsmi_get_cpu_dimm_thermal_sensor function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            dimm = amdsmi_get_cpu_dimm_thermal_sensor(processor)
            print(dimm['sensor'])
            print(dimm['update_rate'])
            print(dimm['dimm_addr'])
            print(dimm['temp'])
except AmdSmiException as e:
    print(e)
```

7.2.25 amdsmi_set_cpu_xgmi_width

Description: Set xgmi width.

Input: amdsmi xgmi width

Exceptions that can be thrown by amdsmi_set_cpu_xgmi_width function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            xgmi_width = amdsmi_set_cpu_xgmi_width(processor, 0, 100)
except AmdSmiException as e:
    print(e)
```

7.2.26 amdsmi_set_cpu_gmi3_link_width_range

Description: Set gmi3 link width range.

Input: minimum & maximum link width to be set.

Exceptions that can be thrown by amdsmi_set_cpu_gmi3_link_width_range function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            gmi_link_width = amdsmi_set_cpu_gmi3_link_width_range(processor, 0, 100)
except AmdSmiException as e:
    print(e)
```

7.2.27 amdsmi_cpu_apb_enable

Description: Enable APB.

Input: amdsmi processor handle

Exceptions that can be thrown by amdsmi_cpu_apb_enable function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            apb_enable = amdsmi_cpu_apb_enable(processor)
except AmdSmiException as e:
    print(e)
```

7.2.28 amdsmi_cpu_apb_disable

Description: Disable APB.

Input: pstate value

Exceptions that can be thrown by amdsmi_cpu_apb_disable function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            apb_disable = amdsmi_cpu_apb_disable(processor, 0)
except AmdSmiException as e:
    print(e)
```

7.2.29 amdsmi_set_cpu_socket_lclk_dpm_level

Description: Set NBIO lclk dpm level value.

Input: nbio id, min value, max value

Exceptions that can be thrown by amdsmi_set_cpu_socket_lclk_dpm_level function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for socket in socket_handles:
            nbio = amdsmi_set_cpu_socket_lclk_dpm_level(socket, 0, 0, 2)
except AmdSmiException as e:
    print(e)
```

7.2.30 amdsmi_get_cpu_socket_lclk_dpm_level

Description: Get NBIO LCLK dpm level.

Output: nbio id

Exceptions that can be thrown by amdsmi_get_cpu_socket_lclk_dpm_level function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
```

(continues on next page)

(continued from previous page)

```

        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            nbio = amdsmi_get_cpu_socket_lclk_dpm_level(processor)
            print(nbio['max_dpm_level'])
            print(nbio['max_dpm_level'])
except AmdSmiException as e:
    print(e)

```

7.2.31 amdsmi_set_cpu_pcie_link_rate

Description: Set pcie link rate.

Input: rate control value

Exceptions that can be thrown by amdsmi_set_cpu_pcie_link_rate function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            link_rate = amdsmi_set_cpu_pcie_link_rate(processor, 0, 0)
except AmdSmiException as e:
    print(e)

```

7.2.32 amdsmi_set_cpu_df_pstate_range

Description: Set df pstate range.

Input: max pstate, min pstate

Exceptions that can be thrown by amdsmi_set_cpu_df_pstate_range function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            pstate_range = amdsmi_set_cpu_df_pstate_range(processor, 0, 2)
except AmdSmiException as e:
    print(e)

```

7.2.33 amdsmi_get_cpu_current_io_bandwidth

Description: Get current input output bandwidth.

Output: link id and bw type to which io bandwidth to be obtained

Exceptions that can be thrown by amdsmi_get_cpu_current_io_bandwidth function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            io_bw = amdsmi_get_cpu_current_io_bandwidth(processor)
            print(io_bw)
except AmdSmiException as e:
    print(e)
```

7.2.34 amdsmi_get_cpu_current_xgmi_bw

Description: Get current xgmi bandwidth.

Output: amdsmi link id and bw type to which xgmi bandwidth to be obtained

Exceptions that can be thrown by amdsmi_get_cpu_current_xgmi_bw function:

- AmdSmiLibraryException

Example:

```
try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            xgmi_bw = amdsmi_get_cpu_current_xgmi_bw(processor)
            print(xgmi_bw)
except AmdSmiException as e:
    print(e)
```

7.2.35 amdsmi_get_hsmc_metrics_table_version

Description: Get HSMP metrics table version.

Output: amdsmi HSMP metrics table version

Exceptions that can be thrown by amdsmi_get_hsmc_metrics_table_version function:

- AmdSmiLibraryException

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            met_ver = amdsmi_get_hsmp_metrics_table_version(processor)
            print(met_ver)
except AmdSmiException as e:
    print(e)

```

7.2.36 amdsmi_get_hsmp_metrics_table

Description: Get HSMP metrics table

Output: HSMP metric table data

Exceptions that can be thrown by `amdsmi_get_hsmp_metrics_table` function:

- `AmdSmiLibraryException`

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")
    else:
        for processor in processor_handles:
            mtbl = amdsmi_get_hsmp_metrics_table(processor)
            print(mtbl['accumulation_counter'])
            print(mtbl['max_socket_temperature'])
            print(mtbl['max_vr_temperature'])
            print(mtbl['max_hbm_temperature'])
            print(mtbl['socket_power_limit'])
            print(mtbl['max_socket_power_limit'])
            print(mtbl['socket_power'])
except AmdSmiException as e:
    print(e)

```

7.2.37 amdsmi_first_online_core_on_cpu_socket

Description: Get first online core on cpu socket.

Output: first online core on cpu socket

Exceptions that can be thrown by `amdsmi_first_online_core_on_cpu_socket` function:

- `AmdSmiLibraryException`

Example:

```

try:
    processor_handles = amdsmi_get_cpusocket_handles()
    if len(processor_handles) == 0:
        print("No CPU sockets on machine")

```

(continues on next page)

(continued from previous page)

```
else:
    for processor in processor_handles:
        pcore_ind = amdsmi_first_online_core_on_cpu_socket(processor)
        print(pcore_ind)
except AmdSmiException as e:
    print(e)
```

7.2.38 amdsmi_get_cpu_family

Description: Get cpu family.

Output: cpu family

Exceptions that can be thrown by amdsmi_get_cpu_family function:

- AmdSmiLibraryException

Example:

```
try:
    cpu_family = amdsmi_get_cpu_family()
    print(cpu_family)
except AmdSmiException as e:
    print(e)
```

7.2.39 amdsmi_get_cpu_model

Description: Get cpu model.

Output: cpu model

Exceptions that can be thrown by amdsmi_get_cpu_model function:

- AmdSmiLibraryException

Example:

```
try:
    cpu_model = amdsmi_get_cpu_model()
    print(cpu_model)
except AmdSmiException as e:
    print(e)
```

CHANGELOG FOR AMD SMI LIBRARY

Full documentation for amd_smi_lib is available at <https://rocm.docs.amd.com/projects/amdsmi>.

All information listed below is for reference and subject to change.

8.1 amd_smi_lib for ROCm 6.3.1

8.1.1 Added

8.1.2 Changed

- **Changed amd-smi monitor: No longer display ENC_CLOCK/DEC_CLOCK but VCLOCK and DCLOCK.**
Due to fix mentioned in Resolved Issues, this change was needed.
Reason: Navi products use vclk and dclk for both encode and decode. On MI products, only decode is supported.

Before:

```
$ amd-smi monitor -n -d
GPU  ENC_UTIL  ENC_CLOCK  DEC_UTIL  DEC_CLOCK
  0    0.0 %    29 MHz    N/A      22 MHz
  1    0.0 %    29 MHz    N/A      22 MHz
  2    0.0 %    29 MHz    N/A      22 MHz
  3    0.0 %    29 MHz    N/A      22 MHz
  4    0.0 %    29 MHz    N/A      22 MHz
  5    0.0 %    29 MHz    N/A      22 MHz
  6    0.0 %    29 MHz    N/A      22 MHz
  7    0.0 %    29 MHz    N/A      22 MHz
```

After:

```
$ amd-smi monitor -n -d
GPU  ENC_UTIL  DEC_UTIL  VCLOCK  DCLOCK
  0    N/A      0.0 %    29 MHz  22 MHz
  1    N/A      0.0 %    29 MHz  22 MHz
  2    N/A      0.0 %    29 MHz  22 MHz
  3    N/A      0.0 %    29 MHz  22 MHz
  4    N/A      0.0 %    29 MHz  22 MHz
  5    N/A      0.0 %    29 MHz  22 MHz
  6    N/A      0.0 %    29 MHz  22 MHz
  7    N/A      0.0 %    29 MHz  22 MHz
```

8.1.3 Removed

8.1.4 Optimized

8.1.5 Resolved issues

- Fixed `amd-smi monitor`'s encode/decode: `ENC_UTIL`, `DEC_UTIL`, and now associate `VCLOCK/DCLOCK` with both.

Navi products use `vclock` and `dclk` for both encode and decode. On MI products, only decode is supported.

Navi products cannot support displaying `ENC_UTIL %` at this time.

Before:

```
$ amd-smi monitor -n -d
GPU  ENC_UTIL  ENC_CLOCK  DEC_UTIL  DEC_CLOCK
  0    0.0 %    29 MHz    N/A      22 MHz
  1    0.0 %    29 MHz    N/A      22 MHz
  2    0.0 %    29 MHz    N/A      22 MHz
  3    0.0 %    29 MHz    N/A      22 MHz
  4    0.0 %    29 MHz    N/A      22 MHz
  5    0.0 %    29 MHz    N/A      22 MHz
  6    0.0 %    29 MHz    N/A      22 MHz
  7    0.0 %    29 MHz    N/A      22 MHz
```

After:

```
$ amd-smi monitor -n -d
GPU  ENC_UTIL  DEC_UTIL  VCLOCK  DCLOCK
  0    N/A      0.0 %    29 MHz  22 MHz
  1    N/A      0.0 %    29 MHz  22 MHz
  2    N/A      0.0 %    29 MHz  22 MHz
  3    N/A      0.0 %    29 MHz  22 MHz
  4    N/A      0.0 %    29 MHz  22 MHz
  5    N/A      0.0 %    29 MHz  22 MHz
  6    N/A      0.0 %    29 MHz  22 MHz
  7    N/A      0.0 %    29 MHz  22 MHz
```

8.1.6 Upcoming changes

8.1.7 Known issues

8.2 amd_smi_lib for ROCm 6.3.0

8.2.1 Added

- Added support for `amd-smi metric --ecc` & `amd-smi metric --ecc-blocks` on Guest VMs. Guest VMs now support getting current ECC counts and ras information from the Host cards.
- Added support for GPU metrics 1.6 to `amdsmi_get_gpu_metrics_info()`. Updated `amdsmi_get_gpu_metrics_info()` and structure `amdsmi_gpu_metrics_t` to include new fields for PVIOL / TVIOL, XCP (Graphics Compute Partitions) stats, and `pcie_lc_perf_other_end_recovery`:
 - `uint64_t accumulation_counter` - used for all throttled calculations
 - `uint64_t prochot_residency_acc` - Processor hot accumulator

- uint64_t ppt_residency_acc - Package Power Tracking (PPT) accumulator (used in PVIOL calculations)
 - uint64_t socket_thm_residency_acc - Socket thermal accumulator - (used in TVIOL calculations)
 - uint64_t vr_thm_residency_acc - Voltage Rail (VR) thermal accumulator
 - uint64_t hbm_thm_residency_acc - High Bandwidth Memory (HBM) thermal accumulator
 - uint16_t num_partition - corresponds to the current total number of partitions
 - struct amdgpu_xcp_metrics_t xcp_stats[MAX_NUM_XCP] - for each partition associated with current GPU, provides gfx busy & accumulators, jpeg, and decoder (VCN) engine utilizations
 - * uint32_t gfx_busy_inst[MAX_NUM_XCC] - graphic engine utilization (%)
 - * uint16_t jpeg_busy[MAX_NUM_JPEG_ENGS] - jpeg engine utilization (%)
 - * uint16_t vcn_busy[MAX_NUM_VCN] - decoder (VCN) engine utilization (%)
 - * uint64_t gfx_busy_acc[MAX_NUM_XCC] - graphic engine utilization accumulated (%)
 - uint32_t pcie_lc_perf_other_end_recovery - corresponds to the pcie other end recovery counter
- **Added new violation status outputs and APIs: `amdsmi_status_t amdsmi_get_violation_status()`, `amd-smi metric --throttle`, and `amd-smi monitor --violation`.**
Only available for MI300+ ASICs.

Users can now retrieve violation status' through either our Python or C++ APIs. Additionally, we have added capability to view these outputs conveniently through `amd-smi metric --throttle` and `amd-smi monitor --violation`.

Example outputs are listed below (below is for reference, output is subject to change):

```
$ amd-smi metric --throttle
GPU: 0
  THROTTLE:
    ACCUMULATION_COUNTER: 3808991
    PROCHOT_ACCUMULATED: 0
    PPT_ACCUMULATED: 585613
    SOCKET_THERMAL_ACCUMULATED: 2190
    VR_THERMAL_ACCUMULATED: 0
    HBM_THERMAL_ACCUMULATED: 0
    PROCHOT_VIOLATION_STATUS: NOT ACTIVE
    PPT_VIOLATION_STATUS: NOT ACTIVE
    SOCKET_THERMAL_VIOLATION_STATUS: NOT ACTIVE
    VR_THERMAL_VIOLATION_STATUS: NOT ACTIVE
    HBM_THERMAL_VIOLATION_STATUS: NOT ACTIVE
    PROCHOT_VIOLATION_ACTIVITY: 0 %
    PPT_VIOLATION_ACTIVITY: 0 %
    SOCKET_THERMAL_VIOLATION_ACTIVITY: 0 %
    VR_THERMAL_VIOLATION_ACTIVITY: 0 %
    HBM_THERMAL_VIOLATION_ACTIVITY: 0 %

GPU: 1
  THROTTLE:
    ACCUMULATION_COUNTER: 3806335
    PROCHOT_ACCUMULATED: 0
    PPT_ACCUMULATED: 586332
```

(continues on next page)

(continued from previous page)

```

↔A,
    N/A, N/A, N/A]
GFX_BUSY_INST:
XCP_0: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_1: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_2: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_3: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_4: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_5: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_6: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_7: [0 %, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
JPEG_BUSY:
XCP_0: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_1: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_2: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_3: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_4: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_5: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_6: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
XCP_7: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
        N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↔A,
    N/A, N/A, N/A]
VCN_BUSY:
XCP_0: [0 %, N/A, N/A, N/A]
XCP_1: [0 %, N/A, N/A, N/A]
XCP_2: [0 %, N/A, N/A, N/A]
XCP_3: [0 %, N/A, N/A, N/A]
XCP_4: [0 %, N/A, N/A, N/A]
XCP_5: [0 %, N/A, N/A, N/A]
XCP_6: [0 %, N/A, N/A, N/A]

```

(continues on next page)

(continued from previous page)

```

N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↪A,
    N/A, N/A, N/A]
XCP_6: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↪A,
    N/A, N/A, N/A]
XCP_7: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, N/A, N/A, N/A, N/A, N/A, N/A,
N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/
↪A,
    N/A, N/A, N/A]
VCN_BUSY:
XCP_0: [0 %, N/A, N/A, N/A]
XCP_1: [0 %, N/A, N/A, N/A]
XCP_2: [0 %, N/A, N/A, N/A]
XCP_3: [0 %, N/A, N/A, N/A]
XCP_4: [0 %, N/A, N/A, N/A]
XCP_5: [0 %, N/A, N/A, N/A]
XCP_6: [0 %, N/A, N/A, N/A]
XCP_7: [0 %, N/A, N/A, N/A]
GFX_BUSY_ACC:
XCP_0: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_1: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_2: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_3: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_4: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_5: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_6: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
XCP_7: [N/A, N/A, N/A, N/A, N/A, N/A, N/A, N/A]
...

```

- Added `LC_PERF_OTHER_END_RECOVERY` CLI output to `amd-smi metric --pcie` and updated `amdsmi_get_pcie_info()` to include this value.

Feature is only available on MI300+ ASICs

Users can now retrieve both through `amdsmi_get_pcie_info()` which has an updated structure:

```

typedef struct {
    ...
    struct pcie_metric {
        uint16_t pcie_width;           //!< current PCIe width
        uint32_t pcie_speed;           //!< current PCIe speed in MT/s
        uint32_t pcie_bandwidth;       //!< current instantaneous PCIe bandwidth in
↪Mb/s
        uint64_t pcie_replay_count;    //!< total number of the replays issued on the
↪PCIe link
        uint64_t pcie_l0_to_recovery_count; //!< total number of times the PCIe link
↪transitioned from L0 to the recovery state
        uint64_t pcie_replay_roll_over_count; //!< total number of replay rollovers issued
↪on the PCIe link
        uint64_t pcie_nak_sent_count;  //!< total number of NAKs issued on the PCIe
↪link by the device
    };
};

```

(continues on next page)

(continued from previous page)

```

uint64_t pcie_nak_received_count;    //!< total number of NAKs issued on the PCIe
↪link by the receiver
uint32_t pcie_lc_perf_other_end_recovery_count;    //!< PCIe other end recovery counter
uint64_t reserved[12];
} pcie_metric;
uint64_t reserved[32];
} amdsmi_pcie_info_t;

```

- Example outputs are listed below (below is for reference, output is subject to change):

```

$ amd-smi metric --pcie
GPU: 0
  PCIE:
    WIDTH: 16
    SPEED: 32 GT/s
    BANDWIDTH: 18 Mb/s
    REPLAY_COUNT: 0
    L0_TO_RECOVERY_COUNT: 0
    REPLAY_ROLL_OVER_COUNT: 0
    NAK_SENT_COUNT: 0
    NAK_RECEIVED_COUNT: 0
    CURRENT_BANDWIDTH_SENT: N/A
    CURRENT_BANDWIDTH_RECEIVED: N/A
    MAX_PACKET_SIZE: N/A
    LC_PERF_OTHER_END_RECOVERY: 0

GPU: 1
  PCIE:
    WIDTH: 16
    SPEED: 32 GT/s
    BANDWIDTH: 18 Mb/s
    REPLAY_COUNT: 0
    L0_TO_RECOVERY_COUNT: 0
    REPLAY_ROLL_OVER_COUNT: 0
    NAK_SENT_COUNT: 0
    NAK_RECEIVED_COUNT: 0
    CURRENT_BANDWIDTH_SENT: N/A
    CURRENT_BANDWIDTH_RECEIVED: N/A
    MAX_PACKET_SIZE: N/A
    LC_PERF_OTHER_END_RECOVERY: 0
...

```

- **Added retrieving a set of GPUs that are nearest to a given device at a specific link type level.**
 - Added `amdsmi_get_link_topology_nearest()` function to amd-smi C and Python Libraries.
- **Added more supported utilization count types to `amdsmi_get_utilization_count()`.**
- **Added `amd-smi set -L/--clk-limit ...` command.**
 - Equivalent to rocm-smi's `--extremum` command which sets `sclk`'s or `mclk`'s soft minimum or soft maximum clock frequency.
- **Added unittest functionality to test amdsmi API calls in Python.**
- **Changed the power parameter in `amdsmi_get_energy_count()` to `energy_accumulator`.**

- Changes propagate forwards into the python interface as well, however we are maintaining backwards compatibility and keeping the power field in the python API until ROCm 6.4.
- **Added GPU memory overdrive percentage to `amd-smi metric -o`.**
 - Added `amdsmi_get_gpu_mem_overdrive_level()` function to amd-smi C and Python Libraries.
- **Added retrieving connection type and P2P capabilities between two GPUs.**
 - Added `amdsmi_topo_get_p2p_status()` function to amd-smi C and Python Libraries.
 - Added retrieving P2P link capabilities to CLI `amd-smi topology`.

```
$ amd-smi topology -h
usage: amd-smi topology [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                        [-g GPU [GPU ...]] [-a] [-w] [-o] [-t] [-b]
```

If no GPU is specified, returns information **for** all GPUs on the system.
If no topology argument is provided all topology information will be displayed.

Topology arguments:

```
-h, --help                show this help message and exit
-g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
                          ID: 0 | BDF: 0000:0c:00.0 | UUID: <redacted>
                          ID: 1 | BDF: 0000:22:00.0 | UUID: <redacted>
                          ID: 2 | BDF: 0000:38:00.0 | UUID: <redacted>
                          ID: 3 | BDF: 0000:5c:00.0 | UUID: <redacted>
                          ID: 4 | BDF: 0000:9f:00.0 | UUID: <redacted>
                          ID: 5 | BDF: 0000:af:00.0 | UUID: <redacted>
                          ID: 6 | BDF: 0000:bf:00.0 | UUID: <redacted>
                          ID: 7 | BDF: 0000:df:00.0 | UUID: <redacted>
                          all | Selects all devices

-a, --access              Displays link accessibility between GPUs
-w, --weight              Displays relative weight between GPUs
-o, --hops                Displays the number of hops between GPUs
-t, --link-type           Displays the link type between GPUs
-b, --numa-bw             Display max and min bandwidth between nodes
-c, --coherent            Display cache coherent (or non-coherent) link capability
↪ between nodes
-n, --atomics             Display 32 and 64-bit atomic io link capability between nodes
-d, --dma                 Display P2P direct memory access (DMA) link capability
↪ between nodes
-z, --bi-dir              Display P2P bi-directional link capability between nodes
```

Command Modifiers:

```
--json                    Displays output in JSON format (human readable by default).
--csv                     Displays output in CSV format (human readable by default).
--file FILE               Saves output into a file on the provided path (stdout by
↪ default).
--loglevel LEVEL          Set the logging level from the possible choices:
                          DEBUG, INFO, WARNING, ERROR, CRITICAL
```

```
$ amd-smi topology -cndz
CACHE COHERANCY TABLE:
    0000:0c:00.0 0000:22:00.0 0000:38:00.0 0000:5c:00.0 0000:9f:00.0 0000:af:00.0
                                                                (continues on next page)
```

(continued from previous page)

| | | | | | | | | |
|-----|--------------|--------------|------|------|------|------|------|---|
| → 0 | 0000:bf:00.0 | 0000:df:00.0 | | | | | | |
| | 0000:0c:00.0 | SELF | C | NC | NC | C | C | ⌵ |
| → | C | NC | | | | | | |
| | 0000:22:00.0 | C | SELF | NC | C | C | C | ⌵ |
| → | NC | C | | | | | | |
| | 0000:38:00.0 | NC | NC | SELF | C | C | NC | ⌵ |
| → | C | NC | | | | | | |
| | 0000:5c:00.0 | NC | C | C | SELF | NC | C | ⌵ |
| → | NC | NC | | | | | | |
| | 0000:9f:00.0 | C | C | C | NC | SELF | NC | ⌵ |
| → | NC | C | | | | | | |
| | 0000:af:00.0 | C | C | NC | C | NC | SELF | ⌵ |
| → | C | C | | | | | | |
| | 0000:bf:00.0 | C | NC | C | NC | NC | C | ⌵ |
| → | SELF | NC | | | | | | |
| | 0000:df:00.0 | NC | C | NC | NC | C | C | ⌵ |
| → | NC | SELF | | | | | | |

ATOMICS TABLE:

| | | | | | | | | |
|-----|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---|
| | | 0000:0c:00.0 | 0000:22:00.0 | 0000:38:00.0 | 0000:5c:00.0 | 0000:9f:00.0 | 0000:af:00.0 | |
| → 0 | 0000:bf:00.0 | 0000:df:00.0 | | | | | | |
| | 0000:0c:00.0 | SELF | 64,32 | 64,32 | 64 | 32 | 32 | ⌵ |
| → | N/A | 64,32 | | | | | | |
| | 0000:22:00.0 | 64,32 | SELF | 64 | 32 | 32 | N/A | ⌵ |
| → | 64,32 | 64,32 | | | | | | |
| | 0000:38:00.0 | 64,32 | 64 | SELF | 32 | N/A | 64,32 | ⌵ |
| → | 64,32 | 64,32 | | | | | | |
| | 0000:5c:00.0 | 64 | 32 | 32 | SELF | 64,32 | 64,32 | ⌵ |
| → | 64,32 | 32 | | | | | | |
| | 0000:9f:00.0 | 32 | 32 | N/A | 64,32 | SELF | 64,32 | ⌵ |
| → | 32 | 32 | | | | | | |
| | 0000:af:00.0 | 32 | N/A | 64,32 | 64,32 | 64,32 | SELF | ⌵ |
| → | 32 | N/A | | | | | | |
| | 0000:bf:00.0 | N/A | 64,32 | 64,32 | 64,32 | 32 | 32 | ⌵ |
| → | SELF | 64,32 | | | | | | |
| | 0000:df:00.0 | 64,32 | 64,32 | 64,32 | 32 | 32 | N/A | ⌵ |
| → | 64,32 | SELF | | | | | | |

DMA TABLE:

| | | | | | | | | |
|-----|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---|
| | | 0000:0c:00.0 | 0000:22:00.0 | 0000:38:00.0 | 0000:5c:00.0 | 0000:9f:00.0 | 0000:af:00.0 | |
| → 0 | 0000:bf:00.0 | 0000:df:00.0 | | | | | | |
| | 0000:0c:00.0 | SELF | T | T | F | F | T | ⌵ |
| → | F | T | | | | | | |
| | 0000:22:00.0 | T | SELF | F | F | T | F | ⌵ |
| → | T | T | | | | | | |
| | 0000:38:00.0 | T | F | SELF | T | F | T | ⌵ |
| → | T | T | | | | | | |
| | 0000:5c:00.0 | F | F | T | SELF | T | T | ⌵ |
| → | T | F | | | | | | |
| | 0000:9f:00.0 | F | T | F | T | SELF | T | ⌵ |
| → | F | F | | | | | | |
| | 0000:af:00.0 | T | F | T | T | T | SELF | ⌵ |

(continues on next page)

(continued from previous page)

```

→ F          T
0000:bf:00.0 F          T          T          T          F          F          ↵
→ SELF      F
0000:df:00.0 T          T          T          F          F          T          ↵
→ F          SELF

BI-DIRECTIONAL TABLE:
          0000:0c:00.0 0000:22:00.0 0000:38:00.0 0000:5c:00.0 0000:9f:00.0 0000:af:00.0
→ 0 0000:bf:00.0 0000:df:00.0
0000:0c:00.0 SELF      T          T          F          F          T          ↵
→ F          T
0000:22:00.0 T          SELF      F          F          T          F          ↵
→ T          T
0000:38:00.0 T          F          SELF      T          F          T          ↵
→ T          T
0000:5c:00.0 F          F          T          SELF      T          T          ↵
→ T          F
0000:9f:00.0 F          T          F          T          SELF      T          ↵
→ F          F
0000:af:00.0 T          F          T          T          T          SELF      ↵
→ F          T
0000:bf:00.0 F          T          T          T          F          F          ↵
→ SELF      F
0000:df:00.0 T          T          T          F          F          T          ↵
→ F          SELF

Legend:
SELF = Current GPU
ENABLED / DISABLED = Link is enabled or disabled
N/A = Not supported
T/F = True / False
C/NC = Coherent / Non-Coherent io links
64,32 = 64 bit and 32 bit atomic support
<BW from>-<BW to>

```

- Created new `amdsmi_kfd_info_t` and added information under `amd-smi list`.

- Due to fixes needed to properly enumerate all logical GPUs in CPX, new device identifiers were added in to a new `amdsmi_kfd_info_t` which gets populated via the API `amdsmi_get_gpu_kfd_info()`.
- This info has been added to the `amd-smi list`.
- These new fields are only available for BM/Guest Linux devices at this time.

```

typedef struct {
    uint64_t kfd_id;    ///< 0xFFFFFFFFFFFFFFFF if not supported
    uint32_t node_id;   ///< 0xFFFFFFFF if not supported
    uint32_t current_partition_id; ///< 0xFFFFFFFF if not supported
    uint32_t reserved[12];
} amdsmi_kfd_info_t;

```

```

$ amd-smi list
GPU: 0

```

(continues on next page)

(continued from previous page)

```

BDF: 0000:23:00.0
UUID: <redacted>
KFD_ID: 45412
NODE_ID: 1
PARTITION_ID: 0

GPU: 1
BDF: 0000:26:00.0
UUID: <redacted>
KFD_ID: 59881
NODE_ID: 2
PARTITION_ID: 0

```

- Added Subsystem Device ID to `amd-smi static --asic`.
 - No underlying changes to `amdsmi_get_gpu_asic_info`

```

$ amd-smi static --asic
GPU: 0
ASIC:
  MARKET_NAME: MI308X
  VENDOR_ID: 0x1002
  VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
  SUBVENDOR_ID: 0x1002
  DEVICE_ID: 0x74a2
  SUBSYSTEM_ID: 0x74a2
  REV_ID: 0x00
  ASIC_SERIAL: <redacted>
  OAM_ID: 5
  NUM_COMPUTE_UNITS: 20
  TARGET_GRAPHICS_VERSION: gfx942

```

- Added `Target_Graphics_Version` to `amd-smi static --asic` and `amdsmi_get_gpu_asic_info()`.

```

typedef struct {
  char market_name[AMDSMI_256_LENGTH];
  uint32_t vendor_id;    ///< Use 32 bit to be compatible with other platform.
  char vendor_name[AMDSMI_MAX_STRING_LENGTH];
  uint32_t subvendor_id; ///< The subsystem vendor id
  uint64_t device_id;   ///< The device id of a GPU
  uint32_t rev_id;
  char asic_serial[AMDSMI_NORMAL_STRING_LENGTH];
  uint32_t oam_id;      ///< 0xFFFF if not supported
  uint32_t num_of_compute_units; ///< 0xFFFFFFFF if not supported
  uint64_t target_graphics_version; ///< 0xFFFFFFFFFFFFFFFF if not supported
  uint32_t reserved[15];
} amdsmi_asic_info_t;

```

```

$ amd-smi static --asic
GPU: 0
ASIC:
  MARKET_NAME: MI308X
  VENDOR_ID: 0x1002

```

(continues on next page)

(continued from previous page)

```

VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
SUBVENDOR_ID: 0x1002
DEVICE_ID: 0x74a2
SUBSYSTEM_ID: 0x74a2
REV_ID: 0x00
ASIC_SERIAL: <redacted>
OAM_ID: 5
NUM_COMPUTE_UNITS: 20
TARGET_GRAPHICS_VERSION: gfx942

```

8.2.2 Changed

- Improvement: Users now have the ability to set and reset without providing `-g all` using AMD SMI CLI.**
 Users can now provide set and reset without `-g all`. Previously, users were required to provide:
`sudo amd-smi set -g all <set arguments>` or `sudo amd-smi reset -g all <set arguments>`
 This update allows users to set or reset without providing `-g all` arguments. Allowing commands:
`sudo amd-smi set <set arguments>` or `sudo amd-smi reset <set arguments>`
 This action will default to try to set/reset for all AMD GPUs on the user's system.
- Improvement: `amd-smi set --memory-partition` now includes a warning banner and progress bar.**
 For devices which support dynamically changing memory partitions, we now provide a warning for users. We provide this warning to provide users knowledge that this action requires users to quit any gpu workloads. Also to let them know this process will trigger an AMD GPU driver reload. Since this process takes time to complete, a progress bar has been provided until actions can verified as a successful change. Otherwise, AMD SMI will report any errors to users and what actions can be taken. See example below:

```

$ sudo amd-smi set -M NPS1

***** WARNING *****

Setting Dynamic Memory (NPS) partition modes require users to quit all GPU
workloads.
AMD SMI will then attempt to change memory (NPS) partition mode.
Upon a successful set, AMD SMI will then initiate an action to restart amdgpu
driver.
This action will change all GPU's in the hive to the requested memory (NPS)
partition mode.

Please use this utility with caution.

Do you accept these terms? [Y/N] y

Updating memory partition for gpu 0: [] 40/40 secs remain

GPU: 0
MEMORYPARTITION: Successfully set memory partition to NPS1

GPU: 1
MEMORYPARTITION: Successfully set memory partition to NPS1

GPU: 2
MEMORYPARTITION: Successfully set memory partition to NPS1
...

```

- **Updated `amdsmi_get_gpu_accelerator_partition_profile` to provide driver memory partition capabilities.**

Driver now has the ability to report what the user can set memory partition modes to. User can now see available memory partition modes upon an invalid argument return from memory partition mode set (`amdsmi_set_gpu_memory_partition`). This change also updates `amd-smi partition`, `amd-smi partition --memory`, and `amd-smi partition --accelerator` (*see note below)

***Note: Subject to change for ROCm 6.4**

- **Updated `amdsmi_set_gpu_memory_partition` to not return until a successful restart of AMD GPU Driver.**

This change keeps checking for ~ up to 40 seconds for a successful restart of the AMD GPU driver. Additionally, the API call continues to check if memory partition (NPS) SYSFS files are successfully updated to reflect the user's requested memory partition (NPS) mode change. Otherwise, reports an error back to the user. Due to these changes, we have updated AMD SMI's CLI to reflect the maximum wait of 40 seconds, while a memory partition change is in progress.

- **All APIs now have the ability to catch driver reporting invalid arguments.**

Now AMD SMI APIs can show `AMDSMI_STATUS_INVALID` when driver returns `EINVAL`.

For example, if user tries to set to `NPS8`, but the memory partition mode is not an available mode to set to. Commonly referred to as CAPS (see `amd-smi partition --memory`), provided by `amdsmi_get_gpu_accelerator_partition_profile`(*see note below).

***Note: Subject to change for ROCm 6.4**

- **Updated BDF commands to look use KFD SYSFS for BDF: `amdsmi_get_gpu_device_bdf()`.**

This aligns BDF output with ROCm SMI. See below for overview as seen from `rsmi_dev_pci_id_get()` now provides partition ID. See API for better detail. Previously these bits were reserved bits (right before domain) and partition id was within function.

- bits [63:32] = domain
- bits [31:28] = partition id
- bits [27:16] = reserved
- bits [15: 0] = pci bus/device/function

- **Moved python tests directory path install location.**

- `/opt/<rocm-path>/share/amd_smi/pytest/..` to `/opt/<rocm-path>/share/amd_smi/tests/pythonunittest/..`
- On `amd-smi-lib-tests` uninstall, the `amd_smi` tests folder is removed
- Removed `pytest` dependency, our python testing now only depends on the `unittest` framework.

- **Updated Partition APIs and struct information and added `partition_id` to `amd-smi static --partition`.**

- As part of an overhaul to partition information, some partition information will be made available in the `amdsmi_accelerator_partition_profile_t`.
- This struct will be filled out by a new API, `amdsmi_get_gpu_accelerator_partition_profile()`.
- Future data from these APIs will eventually get added to `amd-smi partition`.

```
#define AMDSMI_MAX_ACCELERATOR_PROFILE    32
#define AMDSMI_MAX_CP_PROFILE_RESOURCES  32
#define AMDSMI_MAX_ACCELERATOR_PARTITIONS 8

/**
 * @brief Accelerator Partition. This enum is used to identify
```

(continues on next page)

(continued from previous page)

```

* various accelerator partitioning settings.
*/
typedef enum {
    AMDSMI_ACCELERATOR_PARTITION_INVALID = 0,
    AMDSMI_ACCELERATOR_PARTITION_SPX,          //!< Single GPU mode (SPX)- All XCCs work
                                              //!< together with shared memory
    AMDSMI_ACCELERATOR_PARTITION_DPX,        //!< Dual GPU mode (DPX)- Half XCCs work
                                              //!< together with shared memory
    AMDSMI_ACCELERATOR_PARTITION_TPX,        //!< Triple GPU mode (TPX)- One-third XCCs
                                              //!< work together with shared memory
    AMDSMI_ACCELERATOR_PARTITION_QPX,        //!< Quad GPU mode (QPX)- Quarter XCCs
                                              //!< work together with shared memory
    AMDSMI_ACCELERATOR_PARTITION_CPX,        //!< Core mode (CPX)- Per-chip XCC with
                                              //!< shared memory
} amdsmi_accelerator_partition_type_t;

/**
 * @brief Possible Memory Partition Modes.
 * This union is used to identify various memory partitioning settings.
 */
typedef union {
    struct {
        uint32_t nps1_cap :1; // bool 1 = true; 0 = false; Max uint32 means unsupported
        uint32_t nps2_cap :1; // bool 1 = true; 0 = false; Max uint32 means unsupported
        uint32_t nps4_cap :1; // bool 1 = true; 0 = false; Max uint32 means unsupported
        uint32_t nps8_cap :1; // bool 1 = true; 0 = false; Max uint32 means unsupported
        uint32_t reserved :28;
    } amdsmi_nps_flags_t;

    uint32_t nps_cap_mask;
} amdsmi_nps_caps_t;

typedef struct {
    amdsmi_accelerator_partition_type_t profile_type; // SPX, DPX, QPX, CPX and so on
    uint32_t num_partitions; // On MI300X, SPX: 1, DPX: 2, QPX: 4, CPX: 8, length of
    ↪resources array
    uint32_t profile_index;
    amdsmi_nps_caps_t memory_caps; // Possible memory partition capabilities
    uint32_t num_resources; // length of index_of_resources_profile
    uint32_t resources[AMDSMI_MAX_ACCELERATOR_PARTITIONS][AMDSMI_MAX_CP_PROFILE_RESOURCES];
    uint64_t reserved[6];
} amdsmi_accelerator_partition_profile_t;

```

```

$ amd-smi static --partition
GPU: 0
PARTITION:
    COMPUTE_PARTITION: CPX
    MEMORY_PARTITION: NPS4
    PARTITION_ID: 0

```

8.2.3 Removed

- **Removed `amd-smi reset --compute-partition` and `... --memory-partition` and associated APIs.**
 - This change is part of the partition redesign. Reset functionality will be reintroduced in a later update.
 - associated APIs include `amdsmi_reset_gpu_compute_partition()` and `amdsmi_reset_gpu_memory_partition()`
- **Removed usage of `_validate_positive` in Parser and replaced with `_positive_int` and `_not_negative_int` as appropriate.**
 - This will allow 0 to be a valid input for several options in setting CPUs where appropriate (for example, as a mode or NBIOID)

8.2.4 Optimized

- **Adjusted ordering of `gpu_metrics` calls to ensure that `pcie_bw` values remain stable in `amd-smi metric` & `amd-smi monitor`.**
 - With this change additional padding was added to PCIE_BW `amd-smi monitor --pcie`

8.2.5 Resolved issues

- **Fixed `amdsmi_get_gpu_asic_info`'s `target_graphics_version` and `amd-smi --asic` not displaying properly for MI2x or Navi 3x ASICs.**
- **Fixed `amd-smi reset` commands showing an `AttributeError`.**
- **Improved Offline install process & lowered dependency for PyYAML.**
- **Fixed CPX not showing total number of logical GPUs.**
 - Updates were made to `amdsmi_init()` and `amdsmi_get_gpu_bdf_id(...)`. In order to display all logical devices, we needed a way to provide order to GPU's enumerated. This was done by adding a `partition_id` within the BDF optional `pci_id` bits.
 - * bits [63:32] = domain
 - * bits [31:28] or bits [2:0] = partition id
 - * bits [27:16] = reserved
 - * bits [15:8] = Bus
 - * bits [7:3] = Device
 - * bits [2:0] = Function (partition id maybe in bits [2:0]) <– Fallback for non SPX modes
 - Previously in non-SPX modes (ex. CPX/TPX/DPX/etc) some MI3x ASICs would not report all logical GPU devices within AMD SMI.

```
$ amd-smi monitor -p -t -v
GPU POWER GPU_TEMP MEM_TEMP VRAM_USED VRAM_TOTAL
0 248 W 55 °C 48 °C 283 MB 196300 MB
1 247 W 55 °C 48 °C 283 MB 196300 MB
2 247 W 55 °C 48 °C 283 MB 196300 MB
3 247 W 55 °C 48 °C 283 MB 196300 MB
```

(continues on next page)

(continued from previous page)

| | | | | | | |
|----|-----|---|-------|-------|--------|-----------|
| 4 | 221 | W | 50 °C | 42 °C | 283 MB | 196300 MB |
| 5 | 221 | W | 50 °C | 42 °C | 283 MB | 196300 MB |
| 6 | 222 | W | 50 °C | 42 °C | 283 MB | 196300 MB |
| 7 | 221 | W | 50 °C | 42 °C | 283 MB | 196300 MB |
| 8 | 239 | W | 53 °C | 46 °C | 283 MB | 196300 MB |
| 9 | 239 | W | 53 °C | 46 °C | 283 MB | 196300 MB |
| 10 | 239 | W | 53 °C | 46 °C | 283 MB | 196300 MB |
| 11 | 239 | W | 53 °C | 46 °C | 283 MB | 196300 MB |
| 12 | 219 | W | 51 °C | 48 °C | 283 MB | 196300 MB |
| 13 | 219 | W | 51 °C | 48 °C | 283 MB | 196300 MB |
| 14 | 219 | W | 51 °C | 48 °C | 283 MB | 196300 MB |
| 15 | 219 | W | 51 °C | 48 °C | 283 MB | 196300 MB |
| 16 | 222 | W | 51 °C | 47 °C | 283 MB | 196300 MB |
| 17 | 222 | W | 51 °C | 47 °C | 283 MB | 196300 MB |
| 18 | 222 | W | 51 °C | 47 °C | 283 MB | 196300 MB |
| 19 | 222 | W | 51 °C | 48 °C | 283 MB | 196300 MB |
| 20 | 241 | W | 55 °C | 48 °C | 283 MB | 196300 MB |
| 21 | 241 | W | 55 °C | 48 °C | 283 MB | 196300 MB |
| 22 | 241 | W | 55 °C | 48 °C | 283 MB | 196300 MB |
| 23 | 240 | W | 55 °C | 48 °C | 283 MB | 196300 MB |
| 24 | 211 | W | 51 °C | 45 °C | 283 MB | 196300 MB |
| 25 | 211 | W | 51 °C | 45 °C | 283 MB | 196300 MB |
| 26 | 211 | W | 51 °C | 45 °C | 283 MB | 196300 MB |
| 27 | 211 | W | 51 °C | 45 °C | 283 MB | 196300 MB |
| 28 | 227 | W | 51 °C | 49 °C | 283 MB | 196300 MB |
| 29 | 227 | W | 51 °C | 49 °C | 283 MB | 196300 MB |
| 30 | 227 | W | 51 °C | 49 °C | 283 MB | 196300 MB |
| 31 | 227 | W | 51 °C | 49 °C | 283 MB | 196300 MB |

- Fixed incorrect implementation of the Python API `amdsmi_get_gpu_metrics_header_info()`.
- `amdsmitst TestGpuMetricsRead` now prints metric in correct units.

8.2.6 Upcoming changes

- Python API for `amdsmi_get_energy_count()` will deprecate the `power` field in ROCm 6.4 and use `energy_accumulator` field instead.
- New memory and compute partition APIs incoming for ROCm 6.4.
 - These APIs will be updated to fully populate the CLI and allowing compute (accelerator) partitions to be set by profile ID.
 - One API will be provided, to reset both memory and compute (accelerator).
 - * There are dependencies regarding available compute partitions when in other memory modes.
 - * Driver will be providing these default modes
 - * Memory partition resets (for BM) require driver reloads - this will allow us to notify users before taking this action, then change to the default compute partition modes.
 - The following APIs will remain:

```
amdsmi_status_t
amdsmi_set_gpu_compute_partition(amdsmi_processor_handle processor_handle,
```

(continues on next page)

(continued from previous page)

```

amdsmi_compute_partition_type_t compute_partition);
amdsmi_status_t
amdsmi_get_gpu_compute_partition(amdsmi_processor_handle processor_handle,
                                char *compute_partition, uint32_t len);
amdsmi_status_t
amdsmi_get_gpu_memory_partition(amdsmi_processor_handle processor_handle,
                                char *memory_partition, uint32_t len);
amdsmi_status_t
amdsmi_set_gpu_memory_partition(amdsmi_processor_handle processor_handle,
                                amdsmi_memory_partition_type_t memory_partition);

```

- **amd-smi set --compute-partition “SPX/DPX/CPX...” will modified to accept profile IDs in ROCm 6.4.**
 - This is due to aligning with Host setups and providing more robust partition information through the APIs outlined above. Furthermore, new APIs which will be available on both BM/Host can set by profile ID. (functionality coming soon!)
- **Added preliminary amd-smi partition command.**
 - The new partition command can be used to display GPU information, including memory and accelerator partition information.
 - The command will be at full functionality once additional partition information from amdsmi_get_gpu_accelerator_partition_profile() has been implemented.

8.3 amd_smi_lib for ROCm 6.2.1

8.3.1 Added

- **Removed amd-smi metric --ecc & amd-smi metric --ecc-blocks on Guest VMs.** Guest VMs do not support getting current ECC counts from the Host cards.
- **Added amd-smi static --rason Guest VMs.** Guest VMs can view enabled/disabled ras features that are on Host cards.

8.3.2 Resolved issues

- **Fixed TypeError in amd-smi process -G.**
- **Updated CLI error strings to handle empty and invalid GPU/CPU inputs.**
- **Fixed Guest VM showing passthrough options.**
- **Fixed firmware formatting where leading 0s were missing.**

8.4 amd_smi_lib for ROCm 6.2.0

8.4.1 Added

- **amd-smi dmon is now available as an alias to amd-smi monitor.**
- **Added optional process table under amd-smi monitor -q.**
The monitor subcommand within the CLI Tool now has the -q option to enable an optional process table underneath the original monitored output.

```

$ amd-smi monitor -q
GPU POWER GPU_TEMP MEM_TEMP GFX_UTIL GFX_CLOCK MEM_UTIL MEM_CLOCK ENC_UTIL ENC_
↪CLOCK DEC_UTIL DEC_CLOCK SINGLE_ECC DOUBLE_ECC PCIE_REPLAY VRAM_USED VRAM_TOTAL↪
↪ PCIE_BW
  0 199 W 103 °C 84 °C 99 % 1920 MHz 31 % 1000 MHz N/A ↪
↪ MHz N/A 0 MHz 0 0 0 1235 MB 16335 MB↪
↪ N/A Mb/s

PROCESS INFO:
GPU NAME PID GTT_MEM CPU_MEM VRAM_MEM MEM_USAGE GFX ENC
  0 rvs 1564865 0.0 B 0.0 B 1.1 GB 0.0 B 0 ns 0 ns

```

- **Added Handling to detect VMs with passthrough configurations in CLI Tool.**

CLI Tool had only allowed a restricted set of options for Virtual Machines with passthrough GPUs. Now we offer an expanded set of functions available to passthrough configured GPUs.

- **Added Process Isolation and Clear SRAM functionality to the CLI Tool for VMs.**

VMs now have the ability to set the process isolation and clear the sram from the CLI tool. Using the following commands

```

amd-smi set --process-isolation <0 or 1>
amd-smi reset --clean_local_data

```

- **Added macros that were in `amdsmi.h` to the `amdsmi` Python library `amdsmi_interface.py`.**

Added macros to reference max size limitations for certain `amdsmi` functions such as max dpm policies and max fanspeed.

- **Added Ring Hang event.**

Added `AMDSMI_EVT_NOTIF_RING_HANG` to the possible events in the `amdsmi_evt_notification_type_t` enum.

8.4.2 Optimized

- **Updated CLI error strings to specify invalid device type queried**

```

$ amd-smi static --asic --gpu 123123
Can not find a device: GPU '123123' Error code: -3

```

- **Removed elevated permission requirements for `amdsmi_get_gpu_process_list()`.**

Previously if a processes with elevated permissions was running `amd-smi` would required `sudo` to display all output. Now `amd-smi` will populate all process data and return `N/A` for elevated process names instead. However if ran with `sudo` you will be able to see the name like so:

```

$ amd-smi process
GPU: 0
PROCESS_INFO:
  NAME: N/A
  PID: 1693982
MEMORY_USAGE:
  GTT_MEM: 0.0 B
  CPU_MEM: 0.0 B
  VRAM_MEM: 10.1 GB
  MEM_USAGE: 0.0 B
  USAGE:

```

(continues on next page)

(continued from previous page)

```
GFX: 0 ns
ENC: 0 ns
```

```
$ sudo amd-smi process
GPU: 0
PROCESS_INFO:
  NAME: TransferBench
  PID: 1693982
  MEMORY_USAGE:
    GTT_MEM: 0.0 B
    CPU_MEM: 0.0 B
    VRAM_MEM: 10.1 GB
  MEM_USAGE: 0.0 B
  USAGE:
    GFX: 0 ns
    ENC: 0 ns
```

- **Updated naming for `amdsmi_set_gpu_clear_sram_data()` to `amdsmi_clean_gpu_local_data()`.**
Changed the naming to be more accurate to what the function was doing. This change also extends to the CLI where we changed the `clear-sram-data` command to `clean_local_data`.
- **Updated `amdsmi_clk_info_t` struct in `amdsmi.h` and `amdsmi_interface.py` to align with host/guest.**
Changed `cur_clk` to `clk`, changed `sleep_clk` to `clk_deep_sleep`, and added `clk_locked` value. New struct will be in the following format:

```
typedef struct {
+ uint32_t clk;
  uint32_t min_clk;
  uint32_t max_clk;
+ uint8_t clk_locked;
+ uint8_t clk_deep_sleep;
  uint32_t reserved[4];
} amdsmi_clk_info_t;
```

- **Multiple structure updates in `amdsmi.h` and `amdsmi_interface.py` to align with host/guest.**
Multiple structures used by APIs were changed for alignment unification:
 - Changed `amdsmi_vram_info_t` `vram_size_mb` field changed to `vram_size`
 - Updated `amdsmi_vram_type_t` struct updated to include new enums and added AMDSMI prefix
 - Updated `amdsmi_status_t` some enums were missing the AMDSMI_STATUS prefix
 - Added AMDSMI_PROCESSOR_TYPE prefix to `processor_type_t` enums
 - Removed the fields structure definition in favor for an anonymous definition in `amdsmi_bdf_t`
- **Added AMDSMI prefix in `amdsmi.h` and `amdsmi_interface.py` to align with host/guest.**
Multiple structures used by APIs were changed for alignment unification. AMDSMI prefix was added to the following structures:
 - Added AMDSMI prefix to `amdsmi_container_types_t` enums
 - Added AMDSMI prefix to `amdsmi_clk_type_t` enums
 - Added AMDSMI prefix to `amdsmi_compute_partition_type_t` enums
 - Added AMDSMI prefix to `amdsmi_memory_partition_type_t` enums

- Added AMDSMI prefix to `amdsmi_clk_type_t` enums
- Added AMDSMI prefix to `amdsmi_temperature_type_t` enums
- Added AMDSMI prefix to `amdsmi_fw_block_t` enums

- **Changed `dpm_policy` references to `soc_pstate`.**

The file structure referenced to `dpm_policy` changed to `soc_pstate` and we have changed the APIs and CLI tool to be inline with the current structure. `amdsmi_get_dpm_policy()` and `amdsmi_set_dpm_policy()` is no longer valid with the new API being `amdsmi_get_soc_pstate()` and `amdsmi_set_soc_pstate()`. The CLI tool has been changed from `--policy` to `--soc-pstate`

- **Updated `amdsmi_get_gpu_board_info()` `product_name` to fallback to `pciids`.**

Previously on devices without a FRU we would not populate the product name in the `amdsmi_board_info_t` structure, now we will fallback to using the name listed according to the `pciids` file if available.

- **Updated CLI voltage curve command output.**

The output for `amd-smi metric --voltage-curve` now splits the frequency and voltage output by curve point or outputs N/A for each curve point if not applicable

```
GPU: 0
VOLTAGE_CURVE:
  POINT_0_FREQUENCY: 872 Mhz
  POINT_0_VOLTAGE: 736 mV
  POINT_1_FREQUENCY: 1354 Mhz
  POINT_1_VOLTAGE: 860 mV
  POINT_2_FREQUENCY: 1837 Mhz
  POINT_2_VOLTAGE: 1186 mV
```

- **Updated `amdsmi_get_gpu_board_info()` now has larger structure sizes for `amdsmi_board_info_t`.**

Updated sizes that work for retrieving relevant board information across AMD's ASIC products. This requires users to update any ABIs using this structure.

8.4.3 Resolved issues

- **Fixed Leftover Mutex deadlock when running multiple instances of the CLI tool.**

When running `amd-smi reset --gpureset --gpu all` and then running an instance of `amd-smi static` (or any other subcommand that access the GPUs) a mutex would lock and not return requiring either a clear of the mutex in `/dev/shm` or rebooting the machine.

- **Fixed multiple processes not being registered in `amd-smi process` with json and csv format.**

Multiple process outputs in the CLI tool were not being registered correctly. The json output did not handle multiple processes and is now in a new valid json format:

```
[
  {
    "gpu": 0,
    "process_list": [
      {
        "process_info": {
          "name": "TransferBench",
          "pid": 420157,
          "mem_usage": {
            "value": 0,
            "unit": "B"
          }
        }
      }
    ]
  }
]
```

(continues on next page)

(continued from previous page)

```

    },
    {
      "process_info": {
        "name": "rvs",
        "pid": 420315,
        "mem_usage": {
          "value": 0,
          "unit": "B"
        }
      }
    }
  ]
}
]

```

- **Removed `throttle-status` from `amd-smi monitor` as it is no longer reliably supported.**
Throttle status may work for older ASICs, but will be replaced with PVIOL and TVIOL metrics for future ASIC support. It remains a field in the `gpu_metrics` API and in `amd-smi metric --power`.
- **`amdsmi_get_gpu_board_info()` no longer returns junk char strings.**
Previously if there was a partial failure to retrieve character strings, we would return garbage output to users using the API. This fix intends to populate as many values as possible. Then any failure(s) found along the way, `\0` is provided to `amdsmi_board_info_t` structures data members which cannot be populated. Ensuring empty char string values.
- **Fixed parsing of `pp_od_clk_voltage` within `amdsmi_get_gpu_od_volt_info`.**
The parsing of `pp_od_clk_voltage` was not dynamic enough to work with the dropping of voltage curve support on MI series cards. This propagates down to correcting the CLI's output `amd-smi metric --voltage-curve` to N/A if voltage curve is not enabled.

8.4.4 Known issues

- **`amdsmi_get_gpu_process_isolation` and `amdsmi_clean_gpu_local_data` commands do not currently work and will be supported in a future release.**

8.5 `amd_smi_lib` for ROCm 6.1.2

8.5.1 Added

- **Added process isolation and clean shader APIs and CLI commands.**
Added APIs CLI and APIs to address LeftoverLocals security issues. Allowing clearing the sram data and setting process isolation on a per GPU basis. New APIs:
 - `amdsmi_get_gpu_process_isolation()`
 - `amdsmi_set_gpu_process_isolation()`
 - `amdsmi_set_gpu_clear_sram_data()`
- **Added `MIN_POWER` to output of `amd-smi static --limit`.**
This change helps users identify the range to which they can change the power cap of the GPU. The change is added to simplify why a device supports (or does not support) power capping (also known as overdrive). See `amd-smi set -g all --power-cap <value in W>` or `amd-smi reset -g all --power-cap`.

```

$ amd-smi static --limit
GPU: 0
  LIMIT:
    MAX_POWER: 203 W
    MIN_POWER: 0 W
    SOCKET_POWER: 203 W
    SLOWDOWN_EDGE_TEMPERATURE: 100 °C
    SLOWDOWN_HOTSPOT_TEMPERATURE: 110 °C
    SLOWDOWN_VRAM_TEMPERATURE: 100 °C
    SHUTDOWN_EDGE_TEMPERATURE: 105 °C
    SHUTDOWN_HOTSPOT_TEMPERATURE: 115 °C
    SHUTDOWN_VRAM_TEMPERATURE: 105 °C

GPU: 1
  LIMIT:
    MAX_POWER: 213 W
    MIN_POWER: 213 W
    SOCKET_POWER: 213 W
    SLOWDOWN_EDGE_TEMPERATURE: 109 °C
    SLOWDOWN_HOTSPOT_TEMPERATURE: 110 °C
    SLOWDOWN_VRAM_TEMPERATURE: 100 °C
    SHUTDOWN_EDGE_TEMPERATURE: 114 °C
    SHUTDOWN_HOTSPOT_TEMPERATURE: 115 °C
    SHUTDOWN_VRAM_TEMPERATURE: 105 °C

```

8.5.2 Optimized

- **Updated amd-smi monitor --pcie output.**

The source for pcie bandwidth monitor output was a legacy file we no longer support and was causing delays within the monitor command. The output is no longer using TX/RX but instantaneous bandwidth from `gpu_metrics` instead; updated output:

```

$ amd-smi monitor --pcie
GPU  PCIE_BW
0    26 Mb/s

```

- **amdsmi_get_power_cap_info now returns values in uW instead of W.**
`amdsmi_get_power_cap_info` will return in uW as originally reflected by driver. Previously `amdsmi_get_power_cap_info` returned W values, this conflicts with our sets and modifies values retrieved from driver. We decided to keep the values returned from driver untouched (in original units, uW). Then in CLI we will convert to watts (as previously done - no changes here). Additionally, driver made updates to min power cap displayed for devices when overdrive is disabled which prompted for this change (in this case `min_power_cap` and `max_power_cap` are the same).
- **Updated Python Library return types for amdsmi_get_gpu_memory_reserved_pages & amdsmi_get_gpu_bad_page_info.**
 Previously calls were returning “No bad pages found.” if no pages were found, now it only returns the list type and can be empty.
- **Updated amd-smi metric --ecc-blocks output.**
 The ecc blocks argument was outputting blocks without counters available, updated the filtering show blocks that counters are available for:

```

$ amd-smi metric --ecc-block
GPU: 0
  ECC_BLOCKS:
    UMC:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0
    SDMA:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0
    GFX:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0
    MMHUB:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0
    PCIE_BIF:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0
    HDP:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0
    XGMI_WAFL:
      CORRECTABLE_COUNT: 0
      UNCORRECTABLE_COUNT: 0
      DEFERRED_COUNT: 0

```

- **Removed `amdsmi_get_gpu_process_info` from Python library.**

`amdsmi_get_gpu_process_info` was removed from the C library in an earlier build, but the API was still in the Python interface.

8.5.3 Resolved issues

- **Fixed `amd-smi metric --power` now provides power output for Navi2x/Navi3x/MI1x.**

These systems use an older version of `gpu_metrics` in `amdgpu`. This fix only updates what CLI outputs. No change in any of our APIs.

```

$ amd-smi metric --power
GPU: 0
  POWER:
    SOCKET_POWER: 11 W
    GFX_VOLTAGE: 768 mV
    SOC_VOLTAGE: 925 mV
    MEM_VOLTAGE: 1250 mV
    POWER_MANAGEMENT: ENABLED
    THROTTLE_STATUS: UNTHROTTLED

GPU: 1

```

(continues on next page)

(continued from previous page)

```

POWER:
  SOCKET_POWER: 17 W
  GFX_VOLTAGE: 781 mV
  SOC_VOLTAGE: 806 mV
  MEM_VOLTAGE: 1250 mV
  POWER_MANAGEMENT: ENABLED
  THROTTLE_STATUS: UNTHROTTLED

```

- **Fixed `amdsmitstReadWrite.TestPowerCapReadWrite` test for Navi3X, Navi2X, MI100.**

Updates required `amdsmi_get_power_cap_info` to return in uW as originally reflected by driver. Previously `amdsmi_get_power_cap_info` returned W values, this conflicts with our sets and modifies values retrieved from driver. We decided to keep the values returned from driver untouched (in original units, uW). Then in CLI we will convert to watts (as previously done - no changes here). Additionally, driver made updates to min power cap displayed for devices when overdrive is disabled which prompted for this change (in this case `min_power_cap` and `max_power_cap` are the same).

- **Fixed Python interface call `amdsmi_get_gpu_memory_reserved_pages` & `amdsmi_get_gpu_bad_page_info`.**

Previously Python interface calls to populated bad pages resulted in a `ValueError: NULL pointer access`. This fixes the bad-pages subcommand CLI subcommand as well.

8.6 `amd_smi_lib` for ROCm 6.1.1

8.6.1 Changed

- **Updated metrics `--clocks`.**

Output for `amd-smi metric --clock` is updated to reflect each engine and bug fixes for the clock lock status and deep sleep status.

```

$ amd-smi metric --clock
GPU: 0
CLOCK:
  GFX_0:
    CLK: 113 MHz
    MIN_CLK: 500 MHz
    MAX_CLK: 1800 MHz
    CLK_LOCKED: DISABLED
    DEEP_SLEEP: ENABLED
  GFX_1:
    CLK: 113 MHz
    MIN_CLK: 500 MHz
    MAX_CLK: 1800 MHz
    CLK_LOCKED: DISABLED
    DEEP_SLEEP: ENABLED
  GFX_2:
    CLK: 112 MHz
    MIN_CLK: 500 MHz
    MAX_CLK: 1800 MHz
    CLK_LOCKED: DISABLED
    DEEP_SLEEP: ENABLED
  GFX_3:
    CLK: 113 MHz

```

(continues on next page)

(continued from previous page)

```
MIN_CLK: 500 MHz
MAX_CLK: 1800 MHz
CLK_LOCKED: DISABLED
DEEP_SLEEP: ENABLED
GFX_4:
  CLK: 113 MHz
  MIN_CLK: 500 MHz
  MAX_CLK: 1800 MHz
  CLK_LOCKED: DISABLED
  DEEP_SLEEP: ENABLED
GFX_5:
  CLK: 113 MHz
  MIN_CLK: 500 MHz
  MAX_CLK: 1800 MHz
  CLK_LOCKED: DISABLED
  DEEP_SLEEP: ENABLED
GFX_6:
  CLK: 113 MHz
  MIN_CLK: 500 MHz
  MAX_CLK: 1800 MHz
  CLK_LOCKED: DISABLED
  DEEP_SLEEP: ENABLED
GFX_7:
  CLK: 113 MHz
  MIN_CLK: 500 MHz
  MAX_CLK: 1800 MHz
  CLK_LOCKED: DISABLED
  DEEP_SLEEP: ENABLED
MEM_0:
  CLK: 900 MHz
  MIN_CLK: 900 MHz
  MAX_CLK: 1200 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: DISABLED
VCLK_0:
  CLK: 29 MHz
  MIN_CLK: 914 MHz
  MAX_CLK: 1480 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
VCLK_1:
  CLK: 29 MHz
  MIN_CLK: 914 MHz
  MAX_CLK: 1480 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
VCLK_2:
  CLK: 29 MHz
  MIN_CLK: 914 MHz
  MAX_CLK: 1480 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
```

(continues on next page)

(continued from previous page)

```

VCLK_3:
  CLK: 29 MHz
  MIN_CLK: 914 MHz
  MAX_CLK: 1480 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
DCLK_0:
  CLK: 22 MHz
  MIN_CLK: 711 MHz
  MAX_CLK: 1233 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
DCLK_1:
  CLK: 22 MHz
  MIN_CLK: 711 MHz
  MAX_CLK: 1233 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
DCLK_2:
  CLK: 22 MHz
  MIN_CLK: 711 MHz
  MAX_CLK: 1233 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED
DCLK_3:
  CLK: 22 MHz
  MIN_CLK: 711 MHz
  MAX_CLK: 1233 MHz
  CLK_LOCKED: N/A
  DEEP_SLEEP: ENABLED

```

- **Added deferred ecc counts.**

Added deferred error correctable counts to `amd-smi metric --ecc --ecc-blocks`

```

$ amd-smi metric --ecc --ecc-blocks
GPU: 0
ECC:
  TOTAL_CORRECTABLE_COUNT: 0
  TOTAL_UNCORRECTABLE_COUNT: 0
  TOTAL_DEFERRED_COUNT: 0
  CACHE_CORRECTABLE_COUNT: 0
  CACHE_UNCORRECTABLE_COUNT: 0
ECC_BLOCKS:
  UMC:
    CORRECTABLE_COUNT: 0
    UNCORRECTABLE_COUNT: 0
    DEFERRED_COUNT: 0
  SDMA:
    CORRECTABLE_COUNT: 0
    UNCORRECTABLE_COUNT: 0
    DEFERRED_COUNT: 0
...

```

- **Updated `amd-smi topology --json` to align with host/guest.**

Topology's `--json` output now is changed to align with output host/guest systems. Additionally, users can select/filter specific topology details as desired (refer to `amd-smi topology -h` for full list). See examples shown below.

Previous format:

```
$ amd-smi topology --json
[
  {
    "gpu": 0,
    "link_accessibility": {
      "gpu_0": "ENABLED",
      "gpu_1": "DISABLED"
    },
    "weight": {
      "gpu_0": 0,
      "gpu_1": 40
    },
    "hops": {
      "gpu_0": 0,
      "gpu_1": 2
    },
    "link_type": {
      "gpu_0": "SELF",
      "gpu_1": "PCIE"
    },
    "numa_bandwidth": {
      "gpu_0": "N/A",
      "gpu_1": "N/A"
    }
  },
  {
    "gpu": 1,
    "link_accessibility": {
      "gpu_0": "DISABLED",
      "gpu_1": "ENABLED"
    },
    "weight": {
      "gpu_0": 40,
      "gpu_1": 0
    },
    "hops": {
      "gpu_0": 2,
      "gpu_1": 0
    },
    "link_type": {
      "gpu_0": "PCIE",
      "gpu_1": "SELF"
    },
    "numa_bandwidth": {
      "gpu_0": "N/A",
      "gpu_1": "N/A"
    }
  }
]
```

(continues on next page)

(continued from previous page)

```
}
]
```

New format:

```
$ amd-smi topology --json
[
  {
    "gpu": 0,
    "bdf": "0000:01:00.0",
    "links": [
      {
        "gpu": 0,
        "bdf": "0000:01:00.0",
        "weight": 0,
        "link_status": "ENABLED",
        "link_type": "SELF",
        "num_hops": 0,
        "bandwidth": "N/A",
      },
      {
        "gpu": 1,
        "bdf": "0001:01:00.0",
        "weight": 15,
        "link_status": "ENABLED",
        "link_type": "XGMI",
        "num_hops": 1,
        "bandwidth": "50000-100000",
      },
      ...
    ]
  },
  ...
]
```

```
$ /opt/rocm/bin/amd-smi topology -a -t --json
[
  {
    "gpu": 0,
    "bdf": "0000:08:00.0",
    "links": [
      {
        "gpu": 0,
        "bdf": "0000:08:00.0",
        "link_status": "ENABLED",
        "link_type": "SELF"
      },
      {
        "gpu": 1,
        "bdf": "0000:44:00.0",
        "link_status": "DISABLED",
        "link_type": "PCIE"
      }
    ]
  }
]
```

(continues on next page)

```

    }
  ]
},
{
  "gpu": 1,
  "bdf": "0000:44:00.0",
  "links": [
    {
      "gpu": 0,
      "bdf": "0000:08:00.0",
      "link_status": "DISABLED",
      "link_type": "PCIE"
    },
    {
      "gpu": 1,
      "bdf": "0000:44:00.0",
      "link_status": "ENABLED",
      "link_type": "SELF"
    }
  ]
}
]
}
]

```

8.6.2 Resolved issues

- **Fix for GPU reset error on non-amdgpu cards.**
Previously our reset could attempt to reset non-amd GPUS- resulting in “Unable to reset non-amd GPU” error. Fix updates CLI to target only AMD ASICs.
- **Fix for `amd-smi static --pcie` and `amdsmi_get_pcie_info()` Navi32/31 cards.**
Updated API to include `amdsmi_card_form_factor_t.AMDSMI_CARD_FORM_FACTOR_CEM`. Previously, this would report “UNKNOWN”. This fix provides the correct board `SLOT_TYPE` associated with these ASICs (and other Navi cards).
- **Fix for `amd-smi process`.**
Fixed output results when getting processes running on a device.
- **Improved Error handling for `amd-smi process`.**
Fixed Attribute Error when getting process in csv format

8.6.3 Known issues

- `amd-smi bad-pages` can result with “ValueError: NULL pointer access” with certain PM FW versions.

8.7 `amd_smi_lib` for ROCm 6.1.0

8.7.1 Added

- **Added Monitor Command.**
Provides users the ability to customize GPU metrics to capture, collect, and observe. Output is provided in a table view. This aligns closer to ROCm SMI `rocm-smi` (no argument), additionally allows users to customize what data is helpful for their use-case.

```
$ amd-smi monitor -h
usage: amd-smi monitor [-h] [--json | --csv] [--file FILE] [--loglevel LEVEL]
                        [-g GPU [GPU ...] | -U CPU [CPU ...] | -O CORE [CORE ...]]
                        [-w INTERVAL] [-W TIME] [-i ITERATIONS] [-p] [-t] [-u] [-m] [-n]
                        [-d] [-s] [-e] [-v] [-r]
```

Monitor a target device **for** the specified arguments.
 If no arguments are provided, all arguments will be enabled.
 Use the watch arguments to run continuously

Monitor Arguments:

```
-h, --help                show this help message and exit
-g, --gpu GPU [GPU ...]  Select a GPU ID, BDF, or UUID from the possible choices:
                        ID: 0 | BDF: 0000:01:00.0 | UUID: <redacted>
                        all | Selects all devices
-U, --cpu CPU [CPU ...]  Select a CPU ID from the possible choices:
                        ID: 0
                        all | Selects all devices
-O, --core CORE [CORE ...] Select a Core ID from the possible choices:
                        ID: 0 - 23
                        all | Selects all devices
-w, --watch INTERVAL     Reprint the command in a loop of INTERVAL seconds
-W, --watch_time TIME    The total TIME to watch the given command
-i, --iterations ITERATIONS Total number of ITERATIONS to loop on the given command
-p, --power-usage        Monitor power usage in Watts
-t, --temperature        Monitor temperature in Celsius
-u, --gfx                Monitor graphics utilization (%) and clock (MHz)
-m, --mem                Monitor memory utilization (%) and clock (MHz)
-n, --encoder            Monitor encoder utilization (%) and clock (MHz)
-d, --decoder            Monitor decoder utilization (%) and clock (MHz)
-s, --throttle-status    Monitor thermal throttle status
-e, --ecc                Monitor ECC single bit, ECC double bit, and PCIe replay
↪error counts
-v, --vram-usage         Monitor memory usage in MB
-r, --pcie               Monitor PCIe Tx/Rx in MB/s
```

Command Modifiers:

```
--json                    Displays output in JSON format (human readable by
↪default).
--csv                     Displays output in CSV format (human readable by default).
--file FILE               Saves output into a file on the provided path (stdout by
↪default).
--loglevel LEVEL          Set the logging level from the possible choices:
                        DEBUG, INFO, WARNING, ERROR, CRITICAL
```

```
$ amd-smi monitor -ptumv
GPU POWER GPU_TEMP MEM_TEMP GFX_UTIL GFX_CLOCK MEM_UTIL MEM_CLOCK VRAM_USED ↵
↪VRAM_TOTAL
0 171 W 32 °C 33 °C 0 % 114 MHz 0 % 900 MHz 283 MB ↵
↪196300 MB
1 175 W 33 °C 34 °C 0 % 113 MHz 0 % 900 MHz 283 MB ↵
↪196300 MB
```

(continues on next page)

(continued from previous page)

| | | | | | | | | | | |
|---|--------|----|-------|-------|-----|---------|-----|---------|--------|---|
| 2 | 177 | W | 31 °C | 33 °C | 0 % | 113 MHz | 0 % | 900 MHz | 283 MB | ↳ |
| ↳ | 196300 | MB | | | | | | | | |
| 3 | 172 | W | 33 °C | 32 °C | 0 % | 113 MHz | 0 % | 900 MHz | 283 MB | ↳ |
| ↳ | 196300 | MB | | | | | | | | |
| 4 | 178 | W | 32 °C | 32 °C | 0 % | 113 MHz | 0 % | 900 MHz | 284 MB | ↳ |
| ↳ | 196300 | MB | | | | | | | | |
| 5 | 176 | W | 33 °C | 35 °C | 0 % | 113 MHz | 0 % | 900 MHz | 283 MB | ↳ |
| ↳ | 196300 | MB | | | | | | | | |
| 6 | 176 | W | 32 °C | 32 °C | 0 % | 113 MHz | 0 % | 900 MHz | 283 MB | ↳ |
| ↳ | 196300 | MB | | | | | | | | |
| 7 | 175 | W | 34 °C | 32 °C | 0 % | 113 MHz | 0 % | 900 MHz | 283 MB | ↳ |
| ↳ | 196300 | MB | | | | | | | | |

- **Integrated ESMI Tool.**

Users can get CPU metrics and telemetry through our API and CLI tools. This information can be seen in `amd-smi static` and `amd-smi metric` commands. Only available for limited target processors. As of ROCm 6.0.2, this is listed as:

- AMD Zen3 based CPU Family 19h Models 0h-Fh and 30h-3Fh
- AMD Zen4 based CPU Family 19h Models 10h-1Fh and A0-AFh

See a few examples listed below.

```
$ amd-smi static -U all
CPU: 0
SMU:
  FW_VERSION: 85.90.0
  INTERFACE_VERSION:
  PROTO_VERSION: 6
```

```
$ amd-smi metric -O 0 1 2
CORE: 0
  BOOST_LIMIT:
    VALUE: 400 MHz
  CURR_ACTIVE_FREQ_CORE_LIMIT:
    VALUE: 400 MHz
  CORE_ENERGY:
    VALUE: N/A

CORE: 1
  BOOST_LIMIT:
    VALUE: 400 MHz
  CURR_ACTIVE_FREQ_CORE_LIMIT:
    VALUE: 400 MHz
  CORE_ENERGY:
    VALUE: N/A

CORE: 2
  BOOST_LIMIT:
    VALUE: 400 MHz
  CURR_ACTIVE_FREQ_CORE_LIMIT:
    VALUE: 400 MHz
```

(continues on next page)

(continued from previous page)

```
CORE_ENERGY:
  VALUE: N/A
```

```
$ amd-smi metric -U all
CPU: 0
POWER_METRICS:
  SOCKET POWER: 102675 mW
  SOCKET POWER LIMIT: 550000 mW
  SOCKET MAX POWER LIMIT: 550000 mW
PROCHOT:
  PROCHOT_STATUS: 0
FREQ_METRICS:
  FCLKMEMCLK:
    FCLK: 2000 MHz
    MCLK: 1300 MHz
  CCLKFREQLIMIT: 400 MHz
  SOC_CURRENT_ACTIVE_FREQ_LIMIT:
    FREQ: 400 MHz
    FREQ_SRC: [HSMP Agent]
  SOC_FREQ_RANGE:
    MAX_SOCKET_FREQ: 3700 MHz
    MIN_SOCKET_FREQ: 400 MHz
C0_RESIDENCY:
  RESIDENCY: 4 %
SVI_TELEMETRY_ALL_RAILS:
  POWER: 102673 mW
METRIC_VERSION:
  VERSION: 11
METRICS_TABLE:
  CPU_FAMILY: 25
  CPU_MODEL: 144
RESPONSE:
  MTBL_ACCUMULATION_COUNTER: 2887162626
  MTBL_MAX_SOCKET_TEMPERATURE: 41.0 °C
  MTBL_MAX_VR_TEMPERATURE: 39.0 °C
  MTBL_MAX_HBM_TEMPERATURE: 40.0 °C
  MTBL_MAX_SOCKET_TEMPERATURE_ACC: 108583340881.125 °C
  MTBL_MAX_VR_TEMPERATURE_ACC: 109472702595.0 °C
  MTBL_MAX_HBM_TEMPERATURE_ACC: 111516663941.0 °C
  MTBL_SOCKET_POWER_LIMIT: 550.0 W
  MTBL_MAX_SOCKET_POWER_LIMIT: 550.0 W
  MTBL_SOCKET_POWER: 102.678 W
  MTBL_TIMESTAMP_RAW: 288731677361880
  MTBL_TIMESTAMP_READABLE: Tue Mar 19 12:32:21 2024
  MTBL_SOCKET_ENERGY_ACC: 166127.84 kJ
  MTBL_CCD_ENERGY_ACC: 3317.837 kJ
  MTBL_XCD_ENERGY_ACC: 21889.147 kJ
  MTBL_AID_ENERGY_ACC: 121932.397 kJ
  MTBL_HBM_ENERGY_ACC: 18994.108 kJ
  MTBL_CCLK_FREQUENCY_LIMIT: 3.7 GHz
  MTBL_GFCLK_FREQUENCY_LIMIT: 0.0 MHz
  MTBL_FCLK_FREQUENCY: 1999.988 MHz
```

(continues on next page)

(continued from previous page)

```

MTBL_UCLK_FREQUENCY: 1299.993 MHz
MTBL_SOCCLK_FREQUENCY: [35.716, 35.715, 35.714, 35.714] MHz
MTBL_VCLK_FREQUENCY: [0.0, 53.749, 53.749, 53.749] MHz
MTBL_DCLK_FREQUENCY: [7.143, 44.791, 44.791, 44.791] MHz
MTBL_LCLK_FREQUENCY: [20.872, 18.75, 35.938, 599.558] MHz
MTBL_FCLK_FREQUENCY_TABLE: [1200.0, 1600.0, 1900.0, 2000.0] MHz
MTBL_UCLK_FREQUENCY_TABLE: [900.0, 1100.0, 1200.0, 1300.0] MHz
MTBL_SOCCLK_FREQUENCY_TABLE: [800.0, 1000.0, 1142.857, 1142.857] MHz
MTBL_VCLK_FREQUENCY_TABLE: [914.286, 1300.0, 1560.0, 1720.0] MHz
MTBL_DCLK_FREQUENCY_TABLE: [711.111, 975.0, 1300.0, 1433.333] MHz
MTBL_LCLK_FREQUENCY_TABLE: [600.0, 844.444, 1150.0, 1150.0] MHz
MTBL_CCLK_FREQUENCY_ACC: [4399751656.639, 4399751656.639, 4399751656.639,
↪4399751656.639,
    4399751656.639, 4399751656.639, 4399751656.639, 4399751656.639, 4399751656.639,
↪4399751656.639,
    4399751656.639, 4399751656.639, 4399751656.639, 4399751656.639,
↪4399751656.639,
    4399751656.639, 4399751656.639, 4399751656.639, 4399751656.639,
↪4399751656.639,
    4399751656.639, 4399751656.639, 4399751656.639, 4399751656.639,
↪4399751656.639,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
↪0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
↪0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
↪0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
↪0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] GHz
MTBL_GFXCLK_FREQUENCY_ACC: [0.0, 0.0, 250534397827.603, 251546257401.82,
↪250811364089.836,
    249999070486.505, 251622633562.855, 251342375116.05] MHz
MTBL_GFXCLK_FREQUENCY: [0.0, 0.0, 31.091, 31.414, 31.141, 31.478, 31.32, 31.
↪453]
    MHz
MTBL_MAX_CCLK_FREQUENCY: 3.7 GHz
MTBL_MIN_CCLK_FREQUENCY: 0.4 GHz
MTBL_MAX_GFXCLK_FREQUENCY: 2100.0 MHz
MTBL_MIN_GFXCLK_FREQUENCY: 500.0 MHz
MTBL_MAX_LCLK_DPM_RANGE: 2
MTBL_MIN_LCLK_DPM_RANGE: 0
MTBL_XGMI_WIDTH: 0.0
MTBL_XGMI_BITRATE: 0.0 Gbps
MTBL_XGMI_READ_BANDWIDTH_ACC: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] Gbps
MTBL_XGMI_WRITE_BANDWIDTH_ACC: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] Gbps
MTBL_SOCKET_C0_RESIDENCY: 4.329 %
MTBL_SOCKET_GFX_BUSY: 0.0 %
MTBL_HBM_BANDWIDTH_UTILIZATION: 0.001 %
MTBL_SOCKET_C0_RESIDENCY_ACC: 311523106.34
MTBL_SOCKET_GFX_BUSY_ACC: 84739.281
MTBL_HBM_BANDWIDTH_ACC: 33231180.073 Gbps

```

(continues on next page)


```
$ amd-smi version
AMDSMI Tool: 23.4.2+505b858 | AMDSMI Library version: 24.2.0.0 | ROCm version: 6.1.0
```

- **Added XGMI table.**

Displays XGMI information for AMD GPU devices in a table format. Only available on supported ASICs (eg. MI300). Here users can view read/write data XGMI or PCIe accumulated data transfer size (in KiloBytes).

```
$ amd-smi xgmi
LINK METRIC TABLE:
      bdf          bit_rate max_bandwidth link_type 0000:0c:00.0 0000:22:00.0
↪0000:38:00.0 0000:5c:00.0 0000:9f:00.0 0000:af:00.0 0000:bf:00.0 0000:df:00.0
GPU0 0000:0c:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 1 KB          2 KB          1 KB          2 KB          N/A          2 KB          2 KB
  Write
↪ 1 KB          1 KB          1 KB          1 KB          N/A          1 KB          1 KB
GPU1 0000:22:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 2 KB          1 KB          2 KB          1 KB          0 KB          2 KB          N/A
  Write
↪ 1 KB          1 KB          1 KB          1 KB          0 KB          N/A          1 KB
GPU2 0000:38:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 2 KB          1 KB          2 KB          0 KB          0 KB          1 KB          N/A
  Write
↪ 1 KB          1 KB          1 KB          1 KB          0 KB          1 KB          N/A
GPU3 0000:5c:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ N/A           1 KB          0 KB          0 KB          0 KB          2 KB          2 KB
  Write
↪ N/A           1 KB          1 KB          1 KB          0 KB          1 KB          1 KB
GPU4 0000:9f:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 0 KB          N/A          2 KB          0 KB          0 KB          1 KB          0 KB
  Write
↪ 1 KB          N/A          1 KB          1 KB          0 KB          1 KB          1 KB
GPU5 0000:af:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 0 KB          0 KB          N/A          2 KB          0 KB          2 KB          0 KB
  Write
↪ 1 KB          1 KB          N/A          1 KB          0 KB          1 KB          1 KB
GPU6 0000:bf:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 0 KB          0 KB          0 KB          N/A          0 KB          0 KB          0 KB
  Write
↪ 1 KB          1 KB          1 KB          N/A          0 KB          1 KB          1 KB
GPU7 0000:df:00.0 32 Gb/s 512 Gb/s XGMI
  Read
↪ 0 KB          0 KB          0 KB          0 KB          0 KB          0 KB          0 KB
  Write
↪ 1 KB          1 KB          1 KB          1 KB          0 KB          1 KB          1 KB
```

- **Added units of measure to JSON output.**

We added unit of measure to JSON/CSV `amd-smi metric`, `amd-smi static`, and `amd-smi monitor` com-

mands.

Ex.

```
amd-smi metric -p --json
[
  {
    "gpu": 0,
    "power": {
      "socket_power": {
        "value": 10,
        "unit": "W"
      },
      "gfx_voltage": {
        "value": 6,
        "unit": "mV"
      },
      "soc_voltage": {
        "value": 918,
        "unit": "mV"
      },
      "mem_voltage": {
        "value": 1250,
        "unit": "mV"
      },
      "power_management": "ENABLED",
      "throttle_status": "UNTHROTTLED"
    }
  }
]
```

8.7.2 Changed

- **Topology is now left-aligned with BDF of each device listed individual table's row/columns.**

We provided each device's BDF for every table's row/columns, then left aligned data. We want AMD SMI Tool output to be easy to understand and digest for our users. Having users scroll up to find this information made it difficult to follow, especially for devices which have many devices associated with one ASIC.

```
$ amd-smi topology
ACCESS TABLE:
      0000:0c:00.0 0000:22:00.0 0000:38:00.0 0000:5c:00.0 0000:9f:00.0 0000:af:00.
↪ 0 0000:bf:00.0 0000:df:00.0
0000:0c:00.0 ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ↵
↪   ENABLED   ENABLED
0000:22:00.0 ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ↵
↪   ENABLED   ENABLED
0000:38:00.0 ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ↵
↪   ENABLED   ENABLED
0000:5c:00.0 ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ↵
↪   ENABLED   ENABLED
0000:9f:00.0 ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ↵
↪   ENABLED   ENABLED
0000:af:00.0 ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ENABLED   ↵
↪   ENABLED   ENABLED
```

(continues on next page)

(continued from previous page)

| | | | | | | | | |
|--------------|---------|---------|---------|---------|---------|---------|---------|---|
| 0000:bf:00.0 | ENABLED | ENABLED | ENABLED | ENABLED | ENABLED | ENABLED | ENABLED | ↳ |
| ↳ | ENABLED | ENABLED | | | | | | |
| 0000:df:00.0 | ENABLED | ENABLED | ENABLED | ENABLED | ENABLED | ENABLED | ENABLED | ↳ |
| ↳ | ENABLED | ENABLED | | | | | | |

WEIGHT TABLE:

| | 0000:0c:00.0 | 0000:22:00.0 | 0000:38:00.0 | 0000:5c:00.0 | 0000:9f:00.0 | 0000:af:00.0 | |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|---|
| ↳ 0000:bf:00.0 | 0000:df:00.0 | | | | | | |
| 0000:0c:00.0 | 0 | 15 | 15 | 15 | 15 | 15 | ↳ |
| ↳ | 15 | 15 | | | | | |
| 0000:22:00.0 | 15 | 0 | 15 | 15 | 15 | 15 | ↳ |
| ↳ | 15 | 15 | | | | | |
| 0000:38:00.0 | 15 | 15 | 0 | 15 | 15 | 15 | ↳ |
| ↳ | 15 | 15 | | | | | |
| 0000:5c:00.0 | 15 | 15 | 15 | 0 | 15 | 15 | ↳ |
| ↳ | 15 | 15 | | | | | |
| 0000:9f:00.0 | 15 | 15 | 15 | 15 | 0 | 15 | ↳ |
| ↳ | 15 | 15 | | | | | |
| 0000:af:00.0 | 15 | 15 | 15 | 15 | 15 | 0 | ↳ |
| ↳ | 15 | 15 | | | | | |
| 0000:bf:00.0 | 15 | 15 | 15 | 15 | 15 | 15 | ↳ |
| ↳ | 0 | 15 | | | | | |
| 0000:df:00.0 | 15 | 15 | 15 | 15 | 15 | 15 | ↳ |
| ↳ | 15 | 0 | | | | | |

HOPS TABLE:

| | 0000:0c:00.0 | 0000:22:00.0 | 0000:38:00.0 | 0000:5c:00.0 | 0000:9f:00.0 | 0000:af:00.0 | |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|---|
| ↳ 0000:bf:00.0 | 0000:df:00.0 | | | | | | |
| 0000:0c:00.0 | 0 | 1 | 1 | 1 | 1 | 1 | ↳ |
| ↳ | 1 | 1 | | | | | |
| 0000:22:00.0 | 1 | 0 | 1 | 1 | 1 | 1 | ↳ |
| ↳ | 1 | 1 | | | | | |
| 0000:38:00.0 | 1 | 1 | 0 | 1 | 1 | 1 | ↳ |
| ↳ | 1 | 1 | | | | | |
| 0000:5c:00.0 | 1 | 1 | 1 | 0 | 1 | 1 | ↳ |
| ↳ | 1 | 1 | | | | | |
| 0000:9f:00.0 | 1 | 1 | 1 | 1 | 0 | 1 | ↳ |
| ↳ | 1 | 1 | | | | | |
| 0000:af:00.0 | 1 | 1 | 1 | 1 | 1 | 0 | ↳ |
| ↳ | 1 | 1 | | | | | |
| 0000:bf:00.0 | 1 | 1 | 1 | 1 | 1 | 1 | ↳ |
| ↳ | 0 | 1 | | | | | |
| 0000:df:00.0 | 1 | 1 | 1 | 1 | 1 | 1 | ↳ |
| ↳ | 1 | 0 | | | | | |

LINK TYPE TABLE:

| | 0000:0c:00.0 | 0000:22:00.0 | 0000:38:00.0 | 0000:5c:00.0 | 0000:9f:00.0 | 0000:af:00.0 | |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|---|
| ↳ 0000:bf:00.0 | 0000:df:00.0 | | | | | | |
| 0000:0c:00.0 | SELF | XGMI | XGMI | XGMI | XGMI | XGMI | ↳ |
| ↳ | XGMI | XGMI | | | | | |
| 0000:22:00.0 | XGMI | SELF | XGMI | XGMI | XGMI | XGMI | ↳ |
| ↳ | XGMI | XGMI | | | | | |

(continues on next page)

(continued from previous page)

| | | | | | | | |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|---|
| 0000:38:00.0 | XGMI | XGMI | SELF | XGMI | XGMI | XGMI | ↳ |
| ↳ | XGMI | XGMI | | | | | |
| 0000:5c:00.0 | XGMI | XGMI | XGMI | SELF | XGMI | XGMI | ↳ |
| ↳ | XGMI | XGMI | | | | | |
| 0000:9f:00.0 | XGMI | XGMI | XGMI | XGMI | SELF | XGMI | ↳ |
| ↳ | XGMI | XGMI | | | | | |
| 0000:af:00.0 | XGMI | XGMI | XGMI | XGMI | XGMI | SELF | ↳ |
| ↳ | XGMI | XGMI | | | | | |
| 0000:bf:00.0 | XGMI | XGMI | XGMI | XGMI | XGMI | XGMI | ↳ |
| ↳ | SELF | XGMI | | | | | |
| 0000:df:00.0 | XGMI | XGMI | XGMI | XGMI | XGMI | XGMI | ↳ |
| ↳ | XGMI | SELF | | | | | |
| NUMA BW TABLE: | | | | | | | |
| | 0000:0c:00.0 | 0000:22:00.0 | 0000:38:00.0 | 0000:5c:00.0 | 0000:9f:00.0 | 0000:af:00.0 | |
| ↳ | 0000:bf:00.0 | 0000:df:00.0 | | | | | |
| 0000:0c:00.0 | N/A | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | 50000- | |
| ↳ | 50000 | 50000-50000 | 50000-50000 | | | | |
| 0000:22:00.0 | 50000-50000 | N/A | 50000-50000 | 50000-50000 | 50000-50000 | 50000- | |
| ↳ | 50000 | 50000-50000 | 50000-50000 | | | | |
| 0000:38:00.0 | 50000-50000 | 50000-50000 | N/A | 50000-50000 | 50000-50000 | 50000- | |
| ↳ | 50000 | 50000-50000 | 50000-50000 | | | | |
| 0000:5c:00.0 | 50000-50000 | 50000-50000 | 50000-50000 | N/A | 50000-50000 | 50000- | |
| ↳ | 50000 | 50000-50000 | 50000-50000 | | | | |
| 0000:9f:00.0 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | N/A | 50000- | |
| ↳ | 50000 | 50000-50000 | 50000-50000 | | | | |
| 0000:af:00.0 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | N/A | ↳ |
| ↳ | 50000-50000 | 50000-50000 | | | | | |
| 0000:bf:00.0 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | 50000- | |
| ↳ | 50000 | N/A | 50000-50000 | | | | |
| 0000:df:00.0 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | 50000-50000 | 50000- | |
| ↳ | 50000 | 50000-50000 | N/A | | | | |

8.7.3 Resolved issues

- **Fix for Navi3X/Navi2X/MI100 `amdsmi_get_gpu_pci_bandwidth()` in `frequencies_read` tests.**
Devices which do not report (eg. Navi3X/Navi2X/MI100) we have added checks to confirm these devices return `AMDSMI_STATUS_NOT_SUPPORTED`. Otherwise, tests now display a return string.
- **Fix for devices which have an older `pyyaml` installed.**
Platforms which are identified as having an older `pyyaml` version or `pip`, we no manually update both `pip` and `pyyaml` as needed. This corrects issues identified below. Fix impacts the following CLI commands:
 - `amd-smi list`
 - `amd-smi static`
 - `amd-smi firmware`
 - `amd-smi metric`
 - `amd-smi topology`

```
TypeError: dump_all() got an unexpected keyword argument 'sort_keys'
```

- **Fix for crash when user is not a member of video/render groups.**
AMD SMI now uses same mutex handler for devices as rocm-smi. This helps avoid crashes when DRM/device data is inaccessible to the logged in user.

8.8 amd_smi_lib for ROCm 6.0.0

8.8.1 Added

- **Integrated the E-SMI (EPYC-SMI) library.**
You can now query CPU-related information directly through AMD SMI. Metrics include power, energy, performance, and other system details.
- **Added support for gfx942 metrics.**
You can now query MI300 device metrics to get real-time information. Metrics include power, temperature, energy, and performance.
- **Compute and memory partition support.**
Users can now view, set, and reset partitions. The topology display can provide a more in-depth look at the device's current configuration.

8.8.2 Changed

- **GPU index sorting made consistent with other tools.**
To ensure alignment with other ROCm software tools, GPU index sorting is optimized to use Bus:Device.Function (BDF) rather than the card number.
- **Topology output is now aligned with GPU BDF table.**
Earlier versions of the topology output were difficult to read since each GPU was displayed linearly. Now the information is displayed as a table by each GPU's BDF, which closer resembles rocm-smi output.

8.8.3 Optimized

- Updated to C++17, gtest-1.14, and cmake 3.14

8.8.4 Resolved issues

- **Fix for driver not initialized.**
If driver module is not loaded, user retrieve error reponse indicating amdgpu module is not loaded.

LICENSE

Copyright (c) 2019-2024 Advanced Micro Devices, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.