
AMD SMI
Release 51.0.0

Advanced Micro Devices, Inc.

Mar 25, 2026

INSTALL

1	AMD SMI LIBRARY AND TOOL BUILD	3
1.1	Requirements	3
1.2	Build commands	3
1.3	AMD SMI LIBRARY Build Options	3
1.4	Folder structure	4
1.5	Python wrapper	4
1.6	Python package	5
1.7	AMD SMI tool build	5
2	AMD SMI LIBRARY AND TOOL INSTALLATION	9
2.1	Note on GIM Package Installation:	9
2.2	Installation commands	9
3	AMD SMI C library usage and examples	11
3.1	Key Steps for Using the AMD SMI C Library	11
3.2	AMD SMI C example	11
4	Python package usage	15
5	AMD SMI CLI Tool - Usage Guide	17
5.1	Overview	17
5.2	Return Codes	17
5.3	Commands	18
5.4	Basic Usage	22
5.5	Command Examples with Sample Outputs	25
6	AMD SMI C API reference	57
6.1	File List	57
6.2	Globals	57
6.3	Data Structures	57
6.4	Data Fields	57
7	AMD SMI Python API reference	59
7.1	Requirements	59
7.2	Overview	59
7.3	Amdsmi usage	60
7.4	Amdsmi Exceptions	60
7.5	Amdsmi API	61
8	Versioning Guide for AMD SMI Library and Tool	139
8.1	Overview	139

8.2	SMI Library Versioning	139
8.3	Version Number Components	139
8.4	SMI Tool Versioning	140
8.5	Version Number Components	140
8.6	Version Usage	141
8.7	Version Compatibility	141
8.8	Examples	141
9	License	143

AMD SMI LIB is a library that enables you to manage and monitor AMD Virtualization Enabled GPUs. It is a thread safe, extensible C based library. The library exposes both C and Python API interface.

Note

This is the AMD SMI for SR-IOV Linux host only. If you are looking for Linux baremetal or SR-IOV Linux guest AMD SMI, please go to the [AMD SMI documentation](#).

Some of the features that are exposed in the library are:

- Query static information about the GPU (ASIC, framebuffer etc.)
- Query information about the FW on the physical function
- Query information about the virtual functions on the GPU
- Query GPU metrics (temperature, clocks, usage etc.)
- Set GPU configuration (partitioning modes, FB sharing modes, power cap etc.)
- GPU reset and clear VF FB (on MI products)
- To run the AMD SMI library, the Linux PF driver needs to be installed.

There are two parts to the AMD SMI library interface:

- **C interface**
 - Consists of C function declarations.
 - The client can call these to query information and configure settings on the Linux host machine.
 - The C interface can be used by the clients to build applications in C/C++ by using this interface and library binary.
- **Python interface**
 - Consists of Python function declarations.
 - These directly call the C interface.
 - The client can use the Python interface to build applications in Python.

AMD SMI tool is a command line utility that utilizes AMD SMI Library APIs to monitor and configure AMD GPUs. The tool is used to monitor AMD's GPUs status in a virtualization environment in both host OS and guest VM, as well as in a bare metal environment for both Windows and Linux Operating Systems. The tool outputs GPU/driver information in plain text, in JSON, or in CSV formats while it can also show the info in the console or save to the specified output file.

The tool can be used to:

- Query static information about the GPU (ASIC, framebuffer etc.)
- Query information about the FW on the physical function
- Query information about the virtual functions on the GPU
- Query GPU metrics (temperature, clocks, usage etc.)
- Set GPU configuration (partitioning modes, FB sharing modes, power cap etc.)
- Reset GPU and clear VF FB (on MI products)

For additional information on build, installation, usage, versioning and API references, please refer to the sections below:

Install

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

How to

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

Reference

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

General

- *Library and CLI tool versioning*

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 © 2025 Advanced Micro Devices, Inc. All rights reserved.

AMD SMI LIBRARY AND TOOL BUILD

1.1 Requirements

Before building the integration and unit tests, ensure that `gtest` and `gmock` are installed on your system. These libraries are required for building and running the tests. You can install them using your system's package manager or build them from source. Additionally, ensure that the `lcov` package is installed on your system before running the `gen_coverage` command, as it is necessary for generating code coverage reports.

Minimum supported `lcov` version is 1.15.

1.2 Build commands

1.2.1 AMD SMI library build

When running `make` inside the `gim` folder, the AMD SMI library is built as well. Here are some useful commands for building the AMD SMI library:

- Run `make` in the `smi-lib` folder to build the library.
- Run `make package` to create the AMD SMI Python package.
- Run `make test` to build and run the integration and unit tests.
- Run `make all` to build everything mentioned above.
- Run `make gen_coverage` to calculate the code coverage of the AMD SMI library.
- If any changes are made to the `interface` folder, regenerate the Python wrapper by running `make python_wrapper` and replace the `amdsmi_wrapper.py` file in the `py/interface` folder with the one generated in the build folder `build/amdsmi/amdsmi_wrapper/amdsmi_wrapper.py`.

1.3 AMD SMI LIBRARY Build Options

These options allow you to customize the build process, such as specifying the build type, enabling thread safety, enabling logging, and using the Thread Sanitizer.

- **BUILD_TYPE:**

This option specifies the type of build you want to perform. Common values are `Release` and `Debug`. `Release` builds are optimized for performance and do not include debugging information. `Debug` builds include debugging information and are not optimized, making them suitable for development and debugging. Default: `Release`

- **THREAD_SAFE:**

This option indicates whether the build should include thread safety features. When set to True, thread safety mechanisms (e.g., mutexes, locks) are enabled. When set to False, thread safety mechanisms are disabled, which might improve performance but can lead to race conditions in multi-threaded environments. Default: True

- LOGGING:

This option controls whether logging is enabled in the build. When set to True, logging code is included, which can help with debugging and monitoring. When set to False, logging code is excluded, which might improve performance. Default: False

- THREAD_SANITIZER:

This option indicates whether the Thread Sanitizer should be enabled. Thread Sanitizer is a tool that detects data races in multi-threaded programs. When set to True, the build includes Thread Sanitizer instrumentation. When set to False, Thread Sanitizer is not included. Default: False

- ADDRESS_SANITIZER:

This option indicates whether the Address Sanitizer should be enabled. Address Sanitizer is a tool that detects memory errors such as buffer overflows, use-after-free, and memory leaks. When set to True, the build includes Address Sanitizer instrumentation. When set to False, Address Sanitizer is not included. Default: False

1.4 Folder structure

Library folder structure is shown below:

```

smi-lib/
├── build/                # Contains all generated files during build
├── cli/
│   └── cpp/             # CLI C++ source code
├── dl/                  # Contains downloaded packages during build
├── drv/
│   ├── core/           # Core driver SMI code
│   ├── inc/            # Driver SMI headers
│   └── linux/          # Linux platform-specific driver code
├── examples/           # Examples of using C APIs
├── inc/                 # Internal include files
│   ├── common/        # Interface between driver and SMI library
│   └── linux/         # Header files specific to the Linux platform
├── interface/         # C API interface for clients
├── py/
│   └── interface/     # Python interface and wrapper around C APIs
├── src/                # C implementation
├── tests/              # Library gtest/gmock tests
│   ├── integration/   # Integration tests
│   └── unit/          # Unit tests
├── utils/
│   └── scripts/       # Utility scripts

```

1.5 Python wrapper

AMD SMI stack contains a wrapper around C SMI Library. The Python API is a one-to-one mapping to the C interface of SMI Library. It exposes library functionality into Python language, allowing for fast and easy scripting.

1.5.1 Code

The Python Wrapper source code can be found in `smi-lib/py/interface` folder.

1.5.2 Build

The wrapper is built together with the SMI Library. For detailed instructions, refer to the *AMD SMI LIBRARY BUILD* section.

1.5.3 Code style

The code style follows the Python PEP-8 standard. See [PEP-8](#) for details.

1.5.4 Dependencies

- Requires Python 3.10 or higher.
- Python Wrapper relies on `ctypes` extension of Python language. See [CTYPES]<https://docs.python.org/3/library/ctypes.html> for details. By using this extension python code can load shared library and call its functions. The extension comes by default with Python 3.

1.5.5 Flow

1. The flow starts with the C library `libamdsmi.so`, which contains the functionality to be used in Python.
2. A Python wrapper (`amdsmi_wrapper.py`) loads the `libamdsmi.so` library and provides a Python interface to its functions.
3. Another wrapper (`amdsmi_interface.py`) provides a user-friendly interface for interacting with the library.

1.5.6 Regenerate wrapper

If any changes are made to the C interface (`smi-lib/interface/amdsmi.h`), regenerate the Python wrapper by running `make python_wrapper` and replace the `amdsmi_wrapper.py` file in the `py/interface` folder with the one generated in the build folder `build/amdsmi/amdsmi_wrapper/amdsmi_wrapper.py`.

1.6 Python package

The AMD SMI Python package is a Python package that provides a wrapper around the AMD System Management Interface (SMI) library. It allows you to interact with AMD GPUs (Graphics Processing Units) and retrieve various information and metrics related to GPU performance, temperature, power consumption, and more. To use the AMD SMI Python package, you need to have the AMD GPU driver installed on your system. The package interacts with the AMD SMI C library, to communicate with the GPU hardware. On a Linux platform, go to the `smi-lib` directory `gim/smi-lib/` and run the following commands:

```
make clean - cleaning build/ directory
make package -j$(nproc) - building AMD SMI library and getting AMD SMI Python package on the following path: gim/smi-lib/build/amdsmi/package/Release/amdsmi
```

1.7 AMD SMI tool build

The AMD SMI CLI tool is a command line utility built in C++ that utilizes AMD SMI Library APIs to monitor and configure AMD GPUs and NICs on Linux host systems.

1.7.1 Tool Source Code Structure

The CLI tool source code is organized in a structured hierarchy designed for maintainability and platform-specific implementations.

Folder Structure

```
cli/
├── cpp/
│   ├── cmake/                # Contains all CMake files used in the build
│   │   └── linux/
│   ├── docs/
│   │   └── external/        # Contains documents
│   ├── inc/                  # Internal include files
│   ├── src/                  # Source files
│   │   ├── guest/          # Windows Guest-specific source files
│   │   └── host/           # Host-specific source files
│   └── utils/
│       ├── scripts/        # Utility scripts
│       └── third_party/
│           ├── inc/        # Third-party libraries
│           └── json/
│               └── tabulate/
```

Key Components

Include files (inc/) Contains all header files that define the CLI tool's interfaces, including:

- Command parsers and handlers for GPU and NIC operations
- API interface definitions for GPU and NIC management
- Template definitions for output formatting
- Helper functions and utilities

Source files (src/)

- **host/**: Contains Linux host-specific implementations for GPU and NIC management and monitoring
- **guest/**: Contains Windows guest-specific source files (for cross-platform compatibility)

Build system (cmake/)

- **linux/**: Linux-specific CMake configuration files
- Platform-specific build configurations and dependencies

Third-party libraries (utils/third_party/)

- **json/**: JSON parsing and formatting library. Converts internal data structures to properly formatted JSON objects for machine-readable output.
- **tabulate/**: Table formatting library for structured output. Handles the alignment, spacing, and visual formatting of tabular data (like the monitor command output showing GPU metrics in neat columns).

1.7.2 Build Requirements

Prerequisites

- Modern C++ compiler (g++11)
- CMake 3.16 or higher
- AMD SMI Library development files
- Linux kernel headers (for host functionality)

Dependencies

- AMD SMI Library (libamdsmi)
- Standard C++ libraries
- JSON library (bundled in third_party)
- Tabulate library (bundled in third_party)

1.7.3 Build Process

Basic build steps

- Run `make` in the `smi-lib/cli/cpp` folder to build the tool.
- Run `make clean` to remove all files generated during the build process, such as object files and executables, to ensure clean build environment.

1.7.4 Build Options

The CLI tool build system supports various configuration options to customize the build for different environments and hardware support requirements.

NIC support configuration

The CLI tool can be built with or without NIC (Network Interface Card) support depending on your system requirements:

Build with NIC support (default)

```
cd smi-lib/cli/cpp
make
```

Build with NIC support using build flag

```
cd smi-lib/cli/cpp
make AMD_SMI_NIC_SUPPORT=True ..
```

Build without NIC support

```
cd smi-lib/cli/cpp
make AMD_SMI_NIC_SUPPORT=False ..
```

Available build options

- `AMD_SMI_NIC_SUPPORT`: Enable/disable NIC monitoring functionality (Default: True)

Notes:

- When NIC support is disabled (`AMD_SMI_NIC_SUPPORT=False`), all NIC-related commands and functionality will be excluded from the build

1.7.5 Output Location

After successful compilation, the `amd-smi` binary will be generated in: `smi-lib/cli/cpp/build/` (build directory).

1.7.6 Runtime Requirements

Library dependencies The CLI tool requires the AMD SMI Library (`libamdsmi.so`) to be available either:

- In the same directory as the `amd-smi` binary
- In the system library path (`/usr/local/lib`)
- Via `LD_LIBRARY_PATH` environment variable

Driver requirements

- AMD SR-IOV Host driver must be installed and loaded
- For SRIOV functionality: SR-IOV must be enabled in the system

NIC support requirements

- AMD Pensando NIC drivers must be installed and loaded
- Appropriate permissions for network device access

1.7.7 Development Notes

Code organization

- Each command is implemented as a separate class inheriting from a base command interface
- Template-based output formatting ensures consistent display across all commands
- Platform-specific code is isolated in respective directories (`host/` vs `guest/`)

Extending functionality To add new commands or features:

1. Create new command class in appropriate `src/` subdirectory
2. Add corresponding header file in `inc/`
3. Update command parser to recognize new commands

1.7.8 Troubleshooting

Common build issues

- **Missing AMD SMI library:** Ensure `libamdsmi.so` is built and available
- **CMake version:** Verify CMake 3.16+ is installed

Runtime issues

- **Library not found:** Check `LD_LIBRARY_PATH` includes AMD SMI Library location
- **Permission errors:** Ensure proper permissions for GPU device access
- **Driver issues:** Verify AMD SR-IOV Host driver is properly installed and loaded

NIC-related issues

- **NIC not detected:** Verify AMD Pensando NIC drivers are installed and loaded
- **Network permission errors:** Ensure proper network device access permissions

AMD SMI LIBRARY AND TOOL INSTALLATION

2.1 Note on GIM Package Installation:

Installing GIM from the package will automatically build and install the AMD SMI library and tool for the user. This automation eliminates the need for manual execution of the tool build and installation steps, simplifying the setup process.

- After GIM package installation, the `libamdsmi.so` library is located in the system library directory, which is `/usr/local/lib/`. This ensures that the library is accessible system-wide, allowing applications and tools to link against it without requiring additional configuration.
- The `amd-smi` tool will be installed in `/usr/local/bin`, making it available for execution from any directory in the terminal without needing to specify the full path.

If user prefers to manually build and install SMI library and tool from the source code, follow the steps in the following sections.

2.2 Installation commands

After a successful build of AMD SMI library and tool, the following commands will install them on the system:

2.2.1 AMD SMI library:

- Run `sudo make install` in the `smi-lib` folder to install the compiled library (`libamdsmi.so`) and the header file (`amdsmi.h`) to the system library folder (`/usr/local/lib/`) and system include folder (`/usr/local/inc/`), respectively.
- Run `sudo make uninstall` in the `smi-lib` folder to remove the installed library (`libamdsmi.so`) and header file (`amdsmi.h`) from the system library folder (`/usr/local/lib/`) and system include folder (`/usr/local/inc/`), respectively.

2.2.2 AMD SMI tool:

- Run `sudo make install` in the `smi-lib/cli/cpp` folder to install the compiled tool (`amd-smi`) to the system bin folder (`/usr/local/bin/`).
- Run `sudo make uninstall` in the `smi-lib/cli/cpp` folder to uninstall the installed tool (`amd-smi`) from the system bin folder (`/usr/local/bin/`).

AMD SMI C LIBRARY USAGE AND EXAMPLES

This section is dedicated to explaining how to use the AMD System Management Interface (SMI) C library. It provides guidance on the basic setup and teardown of the library, which is essential for interacting with AMD hardware through the SMI API.

The AMD SMI C library allows developers to query and control various aspects of AMD hardware, such as monitoring power usage, temperature, and performance metrics. To effectively use the library, it is important to follow the correct initialization and cleanup procedures.

3.1 Key Steps for Using the AMD SMI C Library

1. **Initialization:** Before making any calls to the AMD SMI API, the library must be initialized using the `amdsmi_init()` function. This function sets up the necessary internal data structures and prepares the library for use.
2. **Performing Operations:** Once the library is initialized, you can use the various API functions provided by AMD SMI to interact with the hardware. These functions allow you to retrieve system information, monitor hardware metrics, and perform other management tasks.
3. **Cleanup:** After completing all operations, it is crucial to call `amdsmi_shut_down()` to properly release resources and close the connection to the driver.

By following these steps, you can ensure that your application interacts with AMD hardware efficiently and safely.

For a detailed example, refer to the code snippet in the next section, which demonstrates the basic structure of an application using the AMD SMI C library.

3.2 AMD SMI C example

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.

Example:

```
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#include "amdsmi.h"

int main(void)
```

(continues on next page)

(continued from previous page)

```

{
    amdsmi_processor_handle *processors = NULL;
    uint32_t gpu_count = 0;
    amdsmi_bdf_t bdf;
    amdsmi_asic_info_t asic;
    int ret = 0;

    // Initialize the AMD SMI library
    ret = amdsmi_init(AMDSMI_INIT_ALL_PROCESSORS);
    if (ret != AMDSMI_STATUS_SUCCESS) {
        fprintf(stderr, "Failed to initialize AMD SMI library: %d\n", ret);
        return ret;
    }

    // Get the number of GPU processors
    ret = amdsmi_get_processor_handles(NULL, &gpu_count, NULL);
    if (ret != AMDSMI_STATUS_SUCCESS) {
        fprintf(stderr, "Failed to get GPU count: %d\n", ret);
        goto fini;
    }

    // Allocate memory for processor handles
    processors = (amdsmi_processor_handle *)malloc(sizeof(amdsmi_processor_handle) * gpu_
↪count);
    if (!processors) {
        fprintf(stderr, "Memory allocation failed\n");
        ret = AMDSMI_STATUS_OUT_OF_RESOURCES;
        amdsmi_shut_down();
        return ret;
    }

    // Retrieve processor handles
    ret = amdsmi_get_processor_handles(NULL, &gpu_count, processors);
    if (ret != AMDSMI_STATUS_SUCCESS) {
        fprintf(stderr, "Failed to get processor handles: %d\n", ret);
        goto fini;
    }

    // Iterate through GPUs and retrieve ASIC information
    for (uint32_t i = 0; i < gpu_count; i++) {
        ret = amdsmi_get_gpu_device_bdf(processors[i], &bdf);
        if (ret != AMDSMI_STATUS_SUCCESS) {
            fprintf(stderr, "Failed to get GPU[%u] BDF: %d\n", i, ret);
            goto fini;
        }
        ret = amdsmi_get_gpu_asic_info(processors[i], &asic);
        if (ret != AMDSMI_STATUS_SUCCESS) {
            fprintf(stderr, "Failed to get ASIC info for GPU[%u]: %d\n", ret, i);
            goto fini;
        }
        printf("GPU[%u] BDF:[%02" PRIx64 "]:[%02" PRIx64 "].%" PRIu64 "] Vendor ID: 0x%04x,
↪Device ID: 0x%04" PRIx64 " Market name: %s\n",

```

(continues on next page)

(continued from previous page)

```
        i,
        (uint64_t)bdf.bdf.bus_number, (uint64_t)bdf.bdf.device_number, (uint64_t)bdf.
↪bdf.function_number,
        asic.vendor_id,
        asic.device_id,
        asic.market_name);
    }

    fini:
    // Free allocated resources and clean up internal library resources
    free(processors);
    amdsmi_shut_down();
    return ret;
}
```


PYTHON PACKAGE USAGE

Before creating python package make sure that python version on your system is at least 3.10

Run `make package` to create the AMD SMI Python package. Open terminal and navigate to `gim/smi-lib/build/amdsmi/package/Release`. Run `sudo python3` command. This command will launch the Python interpreter, allowing you to execute Python code and interact with the Python environment. First, you will need to execute the following lines of code:

```
from amdsmi import * # imports all the functions and classes defined in the amdsmi_
↳package, allowing you to use them directly without specifying the package name.
amdsmi_init() # function is then called to initialize the AMD System Management_
↳Interface (SMI) library. This function sets up the necessary environment and
# resources for interacting with the AMD GPU and accessing its management information.
```

By executing these lines, you can access the functionality provided by the `amdsmi` package and start using the AMD SMI library. Most functions require you to specify the processor handle for which you want to retrieve information. In order to do that, it is necessary to call `amdsmi_get_processor_handles()` function, which will return the list of GPU device handle objects on the current machine. Executing `len(processors)` function, will be able to see the number of elements of the mentioned list:

```
processors = amdsmi_get_processor_handles()
len(processors)
```

Once you have obtained the processor handle, you can call any API that works with it. For example:

```
amdsmi_get_gpu_asic_info(processors[0])
```

Finally, call `amdsmi_shut_down()` to release the SMI library resources and close the connection to the driver:

```
amdsmi_shut_down()
```

```
from amdsmi import *

try:
    amdsmi_init()

    # amdsmi calls ...

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

(continues on next page)

(continued from previous page)

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

AMD SMI CLI TOOL - USAGE GUIDE

5.1 Overview

AMD SMI tool is a command line utility that utilizes AMD SMI Library APIs to monitor and configure AMD GPUs and NICs on Linux host systems. The tool is used to monitor AMD GPUs and NICs status in virtualization environments, providing comprehensive GPU management capabilities for host administrators. The tool outputs GPU and NIC information in plain text, in JSON, or in CSV formats while it can also show the info in the console or save to the specified output file.

5.2 Return Codes

The **AMD SMI CLI tool** uses specific return codes to indicate the status of command execution:

Return Code	Description
0	Success. Does not display a message
-1	Invalid Command “Command [command_user_wrote] is invalid. Run ‘help’ for more info.”
-2	Invalid Parameter “Parameter [command_user_wrote] is invalid. Run ‘help’ for more info.”
-3	Device Not Found “Device [index_from_list BDF UUID inputted by user] cannot be found on the system. Run ‘help’ for more info.”
-4	Invalid File Path “Path [path_user_wrote] cannot be found.”
-5	Invalid Parameter Value “Value [value_user_wrote] is not of valid type or format. Run ‘help’ for more info.”
-6	Missing Parameter Value “Parameter [parameter_which_requires_a_value] requires a value. Run ‘help’ for more info.”
-7	Command Not Supported “Command [command_user_wrote] is not supported on the system. Run ‘help’ for more info.”
-8	Parameter Not Supported “Parameter [parameter_user_wrote] is not supported on the system. Run ‘help’ for more info.”
-9	Required command “Command [command_user_wrote] requires a target argument. Run ‘help’ for more info.”
-10	Invalid subcommand “Command [command_user_wrote] is invalid. Must receive valid AMD-SMI Command first. Run ‘help’ for more info.”
-11	Permission Denied “Permission denied. This action requires elevated privileges.”
-100	Unknown Error “An unknown error has occurred. Run ‘help’ for more info.”

Library Error Codes: (-1014) – (-1001) – SMI-LIB Error “SMI-LIB has returned error [smi_lib_error_code] - [smi_lib_error_code_string]”

5.3 Commands

Commands take arguments that help to specify the type of information to be displayed. Note that some commands such as help, list and version do not have arguments. The commands and respective arguments that they accept are described as follows:

1. **help** Display information about the tool.
2. **version** Display information about current version of the library and the tool.
3. **list** (discovery) Lists all GPUs, GPU VFs, and NICs on the system and their general information.
4. **static** Gets static information about the specified GPU or VF. If no target is specified, returns information for all devices on the system.

GPU Parameters:

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Parameters for a specific GPU
- `--asic`: All asic information.
- `--bus`: All bus information.
- `--vbios`: All video bios information (if available).
- `--board`: All board information.
- `--limit`: All limit metric values (i.e. power and thermal limits).
- `--driver`: Displays driver version.
- `--ras`: Displays ras features information.
- `--dfc-ucode`: All dfc ucode table information.
- `--fb-info`: All fb information.
- `--num-vf`: Displays number of supported and enabled VFs.
- `--vram`: All vram information.
- `--cache`: All cache info.
- `--partition`: All partition information.
- `--ifwi`: All IFWI/video bios information.
- `--numa`: All NUMA information.

VF Parameters:

- `--vf=<gpu_index:vf_index from list, vf_bdf, vf_uuid>`: Gets general information about the specified VF (e.g. timeslice, fb info)

NIC Parameters:

- `--nic=<nic_index from list, nic_bdf>`: Parameters for a specific NIC
 - `--asic`: All asic information.
 - `--bus`: All bus information.
 - `--driver`: NIC driver information.
 - `--numa`: All NUMA information.
 - `--port`: All port information.
 - `--rdma-devices`: All RDMA devices information.

5. **firmware** (ucode) Gets firmware information about the specified GPU, VF, or NIC. If no target is specified, returns information for all GPUs on the system.

GPU Parameters:

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Parameters for a specific GPU
- `--fw-list`: All firmware list information.
- `--error-records`: All error records information.

VF Parameters:

- `--vf=<gpu_index:vf_index from list, vf_bdf, vf_uuid>`: Parameters for a specific VF
- `--fw-list`: All firmware list information.

6. **bad-pages**

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Gets bad page information about the specified GPU. If no argument is provided, returns information for all GPUs on the system.

7. **metric** Gets metric information about the specified GPU, VF, or NIC. If no target is specified, returns information for all devices on the system.

GPU Parameters:

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Parameters for a specific GPU
- `--usage`: All usage information.
- `--power`: All power readings information.
- `--clock`: All frequency sensor readings.
- `--temperature`: All thermal sensor readings.
- `--ecc`: All ecc information.
- `--ecc-block`: Number of ECC errors per block.
- `--pcie`: Current pcie information.
- `--energy`: Amount of energy consumed.

VF Parameters:

- `--vf=<gpu_index:vf_index from list, vf_bdf, vf_uuid>`: Parameters for a specific VF
- `--schedule`: All scheduling info.
- `--guard`: All guard information.
- `--guest-data`: All guest data information.
- `--per-partition`: Per-partition metrics information.

NIC Parameters:

- `--nic=<nic_index from list, nic_bdf>`: Parameters for a specific NIC
 - `--port`: All port information.
 - `--rdma-devices`: All RDMA devices information.

NIC Parameters:

- `--nic=<nic_index from list, nic_bdf>`: Parameters for a specific NIC
 - `--port`: All port information.

- `--rdma-devices`: All RDMA devices information.

Note: When using the `--csv` format modifier with the `metric` command, only one argument is supported per command (e.g., `metric -usage`). For all other formats (plain text and json), multiple arguments are supported. The per-partition command does not support the `--csv` format modifier.

8. `event`

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Displays event information for GPU. If no argument is provided, returns event information for all GPUs on the system.

Note: The `watch`, `watch_time` and `iterations` modifiers are not supported for the `event` command.

9. `topology`

- `--gpu= <gpu_index from list, gpu_bdf, gpu_uuid>`: Displays link topology information. If no argument is provided, returns information for all GPUs on the system.

Topology arguments for the GPU are the following:

- `--weight`: Current weight information.
- `--hops`: Current hops information.
- `--fb-sharing`: Current framebuffer sharing information.
- `--link-type`: Link type information.
- `--coherent`: Cache coherent information.
- `--atomics`: 32 and 64-bit atomic link capability information.
- `--bi-dir`: bi-directional link capability information.
- `--dma`: dma link capability information.

Note: The `topology` command does not support the `--csv` format modifier.

10. `xgmi`

- `--gpu= <gpu_index from list, gpu_bdf, gpu_uuid>`: Displays XGMI capabilities, framebuffer sharing and metric information. If no argument is provided, returns information for all GPUs on the system.

XGMI arguments for the GPU are the following:

- `--caps`: XGMI capabilities.
- `--fb-sharing`: Framebuffer sharing for each mode.
- `--metric`: Metric XGMI information.
- `--source-status`: Port status information.
- `--link-status`: Link status information.

Note: The `xgmi` command does not support the `--csv` format modifier.

11. **reset** Reset or cleanup operations for GPUs and VFs. Available only in plain text. If no target is provided, returns tool exception.

GPU Parameters:

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Parameters for a specific GPU
 - `--gpureset`: Reset all GPUs.

VF Parameters:

- `--vf=<gpu_index:vf_index from list, vf_bdf, vf_uuid>`: Parameters for a specific VF (requires SR-IOV)
 - `--vf-fb`: Cleanup VF FB for the specified VF.

12. set

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Set options for devices. If no argument is provided, returns tool exception. Available only in plain text.

Set arguments for the GPU are the following:

- `--memory-partition=<AmdSmiMemoryPartitionSetting>`: Sets memory partition mode. Run `amd-smi partition` to list memory-partition modes supported on current platform.
- `--accelerator-partition=<profile_index>`: Sets accelerator partition mode to a mode based on `profile_index` from partition command. Run `amd-smi partition` to list accelerator partition modes supported on current platform.
- `--power-cap=<power_cap_value>`: Sets the power cap to the provided cap value. **Note:** Cap value must be between the minimum (`min_power_cap`) and maximum (`max_power_cap`) power cap values which can be retrieved from `amd-smi static --limit` command.
- `--xgmi-plpd=<policy>`: Sets XGMI Per-Link Power Down (PLPD) to enabled or disabled.
- `--num-vf=<number_of_vfs>`: Sets the number of Virtual Functions (VFs) to be enabled on the specified GPU. The number must be within the supported range for the GPU. Use `amd-smi static --gpu=<gpu> --num-vf` to check current VF configuration and supported limits.
- `--soc-pstate=<pstate_level>`: Sets the SOC (System on Chip) performance state level to control power and performance characteristics.
- `--xgmi --fb-sharing-mode=<AmdSmiXgmiFbSharingMode>`: Sets framebuffer sharing mode from list [“MODE_1”, “MODE_2”, “MODE_4”, “MODE_8”] where, MODE_X represents that X GPUs will be in the same group, linked together: MODE_1 (one GPU in a group), MODE_2 (two GPUs in a group), MODE_4 (four GPUs in a group), MODE_8 (eight GPUs in a group). All possible configurations can be seen by running the `amd-smi xgmi` command, not all of them are supported on all systems.
- `--ptl-status=<STATUS>`: Enable or disable the PTL on a GPU processor (ENABLED/DISABLED).
- `--ptl-format=<FRMT1,FRMT2>`: Set the PTL format on a GPU processor. For example, `-ptl-format 18,F32`.

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

GPU Parameters:

- `--gpu=<gpu_index from list, gpu_bdf, gpu_uuid>`: Monitor a specific GPU
 - `--gfx`: Monitor graphics utilization (%) and clock (MHz).
 - `--mem`: Monitor memory utilization (%) and clock (MHz).
 - `--encoder`: Monitor encoder utilization (%) and clock (MHz).
 - `--ecc`: Monitor ECC single bit, ECC double bit.
 - `--pcie`: Monitor PCIe bandwidth in Mb/s and PCIe replay error count.
 - `--power-usage`: Monitor power usage in Watts.
 - `--temperature`: Monitor temperature in Celsius.
 - `--decoder`: Monitor decoder utilization (%) and clock (MHz).

14. partition

- `--gpu= <gpu_index from list, gpu_bdf, gpu_uuid>`: Displays capabilities and current information for memory and accelerator partition. If no argument is provided, returns information for all GPUs on the system.

Partition arguments for the GPU are the following:

- `--current`: Current memory and accelerator partition information.
- `--memory`: Memory partition information.
- `--accelerator`: Accelerator partition information.
- `--global`: Global partition configuration settings.

Note: The partition command does not support the `--csv` or `--json` format modifiers.

15. **ras** Retrieves RAS (Reliability, Availability, Serviceability) error information. (MI300 host systems only, human-readable output only - `cper|afid`, all formats allowed - `policy`)

RAS arguments (mutually exclusive):

- `--cper --severity=<fatal|nonfatal-uncorrected|nonfatal-corrected|all> [--folder=FOLDER] [--file-limit=NUMBER] [--follow]`: Get CPER (Common Platform Error Record) entries based on severity level. Supports GPU filtering with `--gpu`. Optional folder saves error files. File limit controls maximum saved files. Follow enables continuous monitoring.
- `--afid --cper-file=FILE`: Extract AFID (AMD Field ID) list from existing CPER file. GPU filtering not supported.
- `--policy`: Get RAS policy information. Supports GPU filtering with `--gpu`.

16. **node** Displays baseboard informations and node power management informations for a NODE. (MI350 host system only).

Node arguments are the following:

- `-b, --baseboard`: Show baseboard information.
- `-p, --power-management`: Show power management information.

5.4 Basic Usage

5.4.1 Command Syntax

```
sudo amd-smi <command> <options>
```

- `<command>` is the primary command to execute. It must be the first argument after **amd-smi**.
- `<options>` can include subcommands, modifiers, or other arguments relevant to the specified command.

5.4.2 Getting Help

To get detailed information about the available commands and options, you can run help command. The help command provides a comprehensive overview of the tool's functionalities and usage instructions. Simply run tool without arguments or with command help.

```
$ sudo amd-smi help
```

Copyright 2023-2025 Advanced Micro Devices, Inc. All rights reserved.

(continues on next page)

(continued from previous page)

```
usage: amd-smi help

AMD System Management Interface | AMD SMI tool version 32.6.0

AMD-SMI Commands:
```

	Descriptions
list	List device information
static	Gets static information about the specified device
metric	Gets metric information about the specified device
version	Display version information (GPU only)
monitor	Monitor metrics for target devices (GPU only)
bad-pages	Gets bad page information about the specified device (GPU only)
event	Displays event information for the given device (GPU only)
firmware	Gets firmware information about the specified device (GPU only)
set	Set options for devices (GPU only)
reset	Reset options for devices (GPU only)
xgmi	Displays xgmi information of the devices (GPU only)
topology	Displays topology information of the devices (GPU only)
partition	Displays partition information of the devices (GPU only)
ras	Displays ras information of the devices (GPU only)
node	Displays node information of the devices (GPU only)

From help message you can see which subcommands are supported on the system and a short description for each command.

To access the help documentation for a specific command, simply use that command name followed by the help command. For example, if you want to get help for “list” command you can use the tool the following way.

```
$ sudo amd-smi list --help

Copyright 2023-2025 Advanced Micro Devices, Inc. All rights reserved.

usage: amd-smi list [-h | --help] [--json | --csv] [--file FILE] [-g | --gpu [GPU ...]]

List all GPUs and VFs on the system and their most basic general information.
If no GPU is specified, returns basic information for all GPUs on the system.

List arguments:
```

	Description:
-h, --help	show this help message and exit
-g, --gpu=[GPU ...]	Select a GPU ID, BDF or UUID, if not selected it will return ↵ ↵for all GPUs

```
Command Modifiers:
```

	Description:
--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).

5.4.3 Output Formats

The AMD SMI CLI tool supports three output formats:

Human-readable (Default)

```
$ sudo amd-smi list

GPU: 0
  BDF: 0000:0c:00.0
  UUID: 67ff74a1-0000-1000-8081-b5b9fd6edd00
  VF: 0
    BDF: 0000:0c:02.0
    UUID: 670074a1-0000-1000-8081-b5b9fd6edd00
```

JSON Format

```
$ sudo amd-smi list --json

[
  {
    "gpu": 0,
    "bdf": "0000:0c:00.0",
    "uuid": "67ff74a1-0000-1000-8081-b5b9fd6edd00",
    "vfs": [
      {
        "vf": 0,
        "bdf": "0000:0c:02.0",
        "uuid": "670074a1-0000-1000-8081-b5b9fd6edd00"
      }
    ]
  }
]
```

CSV Format

```
$ sudo amd-smi list --csv

gpu,gpu_bdf,gpu_uuid,vf,vf_bdf,vf_uuid
0,0000:0c:00.0,67ff74a1-0000-1000-8081-b5b9fd6edd00,0,0000:0c:02.0,670074a1-0000-1000-
→8081-b5b9fd6edd00
```

5.4.4 Saving Output to File

All outputs can be saved to a file using the `--file` parameter:

```
sudo amd-smi list --file=output.txt
```

5.5 Command Examples with Sample Outputs

5.5.1 1. Version Information

```
$ sudo amd-smi version
```

Output:

```
VERSION:
  TOOL_NAME: AMD SMI tool
  TOOL_VERSION: 29.0.0
  LIB_VERSION: 35.0.0
```

5.5.2 2. Static Information

Get all static information for GPU 0:

```
$ sudo amd-smi static --gpu=0
```

Output:

```
GPU: 0
  ASIC:
    MARKET_NAME: MI300X
    VENDOR_ID: 0x1002
    VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
    SUBVENDOR_ID: 0x1002
    DEVICE_ID: 0x74A1
    SUBSYSTEM_ID: 0x74A1
    REV_ID: 0x0
    ASIC_SERIAL: 0xF33397508B72EAAF
    OAM_ID: 2
    NUM_OF_COMPUTE_UNITS: 304
  BUS:
    BDF: 0000:0c:00.0
    MAX_PCIE_WIDTH: 16
    MAX_PCIE_SPEED: 32 GT/s
    PCIE_INTERFACE_VERSION: Gen 5
    SLOT_TYPE: OAM
    MAX_PCIE_INTERFACE_VERSION: Gen 5
  VBIOS:
    NAME: AMD MI300X_PRODUCTION_1VF
    BUILD_DATE: 2024/03/15 14:30
    PART_NUMBER: 113-MI3PRD-001
    VERSION: 022.040.003.036.000001
  BOARD:
    MODEL_NUMBER: 102-G30201-0B
    PRODUCT_SERIAL: PCB068560-0020
    FRU_ID: 113-AMDG302010B14
    PRODUCT_NAME: Instinct MI300X
    MANUFACTURER_NAME: AMD
  LIMIT:
    MAX_POWER: 750 W
```

(continues on next page)

(continued from previous page)

```
MIN_POWER: 100 W
SOCKET_POWER: 750 W
SLOWDOWN_EDGE_TEMPERATURE: N/A C
SLOWDOWN_HOTSPOT_TEMPERATURE: 100 C
SLOWDOWN_MEM_TEMPERATURE: 95 C
SHUTDOWN_EDGE_TEMPERATURE: N/A C
SHUTDOWN_HOTSPOT_TEMPERATURE: 110 C
SHUTDOWN_MEM_TEMPERATURE: 105 C
PTL: ENABLED
PTL_FORMAT: I8,BF16
VRAM:
TYPE: HBM3
VENDOR: HYNIX
SIZE: 196592 MB
BIT_WIDTH: 8192
MAX_BANDWIDTH: 5300 GB/s
```

Get specific static information:

```
$ sudo amd-smi static --gpu=0 --asic --limit
```

Output

```
GPU: 0
ASIC:
MARKET_NAME: AMD Instinct MI350X
VENDOR_ID: 0x1002
VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
SUBVENDOR_ID: 0x1002
DEVICE_ID: 0x75A0
SUBSYSTEM_ID: 0x75A0
REV_ID: 0x0
ASIC_SERIAL: 0xBA6524093004B8B2
OAM_ID: 0
NUM_OF_COMPUTE_UNITS: 256
LIMIT:
MAX_POWER: 1000 W
MIN_POWER: 0 W
SOCKET_POWER: 1000 W
SLOWDOWN_EDGE_TEMPERATURE: N/A
SLOWDOWN_HOTSPOT_TEMPERATURE: 100 C
SLOWDOWN_MEM_TEMPERATURE: 115 C
SHUTDOWN_EDGE_TEMPERATURE: N/A
SHUTDOWN_HOTSPOT_TEMPERATURE: 110 C
SHUTDOWN_MEM_TEMPERATURE: 120 C
PTL: ENABLED
PTL_FORMAT: I8,BF16
```

5.5.3 3. Metric Information

Get usage metrics:

```
$ sudo amd-smi metric --gpu=0 --usage
```

Output:

```
GPU: 0
  USAGE:
    GFX_ACTIVITY: 45 %
    UMC_ACTIVITY: 12 %
    MM_ACTIVITY: 3 %
    VCN_ACTIVITY: [ 0 %, 2 %, 0 %, 1 % ]
    JPEG_ACTIVITY: [ 0 %, 0 %, 1 %, 0 %, 0 %, 0 %, 0 %, 0 % ]
```

Get power metrics:

```
$ sudo amd-smi metric --gpu=0 --power
```

Output:

```
GPU: 0
  POWER:
    SOCKET_POWER: 320 W
    GFX_VOLTAGE: 875 mV
    SOC_VOLTAGE: 950 mV
    MEM_VOLTAGE: 1250 mV
    POWER_MANAGEMENT: ENABLED
```

Get temperature metrics:

```
$ sudo amd-smi metric --gpu=0 --temperature
```

Output:

```
GPU: 0
  TEMPERATURE:
    EDGE: 65 C
    HOTSPOT: 75 C
    MEM: 68 C
```

Get clock information:

```
$ sudo amd-smi metric --gpu=0 --clock
```

Output:

```
GPU: 0
  CLOCK:
    GFX:
      CLK: 1800 MHz
      MIN_CLK: 500 MHz
      MAX_CLK: 2100 MHz
      CLK_LOCKED: DISABLED
```

(continues on next page)

(continued from previous page)

```
DEEP_SLEEP: DISABLED
MEM:
  CLK: 1600 MHz
  MIN_CLK: 400 MHz
  MAX_CLK: 1600 MHz
  CLK_LOCKED: DISABLED
  DEEP_SLEEP: DISABLED
```

5.5.4 4. Firmware Information

```
$ sudo amd-smi firmware --gpu=0
```

Output:

```
GPU: 0
FW_LIST:
  FW_0:
    FW_ID: SMU
    FW_VERSION: 0.85.117.1
  FW_1:
    FW_ID: CP_MEC_JT1
    FW_VERSION: 0x80b8
  FW_2:
    FW_ID: CP_MEC1
    FW_VERSION: 0x80b8
  FW_3:
    FW_ID: RLC
    FW_VERSION: 0x45
  FW_4:
    FW_ID: SDMA0
    FW_VERSION: 0x18
  FW_5:
    FW_ID: SDMA1
    FW_VERSION: 0x18
  FW_6:
    FW_ID: SDMA2
    FW_VERSION: 0x18
  FW_7:
    FW_ID: SDMA3
    FW_VERSION: 0x18
  FW_8:
    FW_ID: RLC_V
    FW_VERSION: 0x1a
  FW_9:
    FW_ID: MMSCH
    FW_VERSION: 8.0.19
  FW_10:
    FW_ID: PSP_SYSDRV
    FW_VERSION: 0.36.2.5a
  FW_11:
    FW_ID: PSP_SOSDRV
    FW_VERSION: 0.36.2.5a
```

(continues on next page)

(continued from previous page)

```

FW_12:
  FW_ID: PSP_KEYDB
  FW_VERSION: 5.0.36.0
FW_13:
  FW_ID: DFC
  FW_VERSION: 0.1.0.1
FW_14:
  FW_ID: PSP_BL
  FW_VERSION: 0.a1.2.1e
FW_15:
  FW_ID: REG_ACCESS_WHITELIST
  FW_VERSION: c.2.36.0
FW_16:
  FW_ID: P2S_TABLE
  FW_VERSION: 0x50101
FW_17:
  FW_ID: PSP_SOC
  FW_VERSION: 0.36.2.5a
FW_18:
  FW_ID: PSP_DBG
  FW_VERSION: 0.36.2.5a
FW_19:
  FW_ID: PSP_INTF
  FW_VERSION: 0.36.2.5a
FW_20:
  FW_ID: PSP_RAS
  FW_VERSION: 0.36.2.5a
ERROR_RECORDS:

```

5.5.5 5. Bad Pages Information

```
$ sudo amd-smi bad-pages --gpu=0
```

Output:

```

GPU: 0
BAD_PAGE_1:
  RETIRED_BAD_PAGE: 0x7FFF12345000
  TIMESTAMP: 01/10/2025:08/41/33
  MEM_CHANNEL: 2
  MCUMC_ID: 1
BAD_PAGE_2:
  RETIRED_BAD_PAGE: 0x7FFF12346000
  TIMESTAMP: 03/10/2025:06/11/13
  MEM_CHANNEL: 3
  MCUMC_ID: 1

```

5.5.6 6. Event Information

```
$ sudo amd-smi event --gpu=0
```

Output:

```
EVENT_INFO:
GPU: 0
  MESSAGE: Temperature threshold exceeded
  CATEGORY: THERMAL
  DATE: 2025-10-08:11:23:07.505
GPU: 0
  MESSAGE: ECC single bit error corrected
  CATEGORY: ECC
  DATE: 2025-10-09:10:34:25.237
```

5.5.7 7. Topology Information

```
$ sudo amd-smi topology --weight
```

Output:

```
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
↪ 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
```

5.5.8 8. XGMI Information

```
$ sudo amd-smi xgmi --caps
```

Output:

```
XGMI_CONFIGURATION_SUPPORT_CAPABILITY:
      MODE_1      MODE_2      MODE_4      MODE_8      MODE_CUSTOM
0000:0c:00.0 SUPPORTED    SUPPORTED    SUPPORTED    SUPPORTED    SUPPORTED
0000:22:00.0 SUPPORTED    SUPPORTED    SUPPORTED    SUPPORTED    SUPPORTED
0000:38:00.0 SUPPORTED    SUPPORTED    SUPPORTED    SUPPORTED    SUPPORTED
```

(continues on next page)

(continued from previous page)

```

0000:5c:00.0 SUPPORTED SUPPORTED SUPPORTED SUPPORTED SUPPORTED
0000:9f:00.0 SUPPORTED SUPPORTED SUPPORTED SUPPORTED SUPPORTED
0000:af:00.0 SUPPORTED SUPPORTED SUPPORTED SUPPORTED SUPPORTED
0000:bf:00.0 SUPPORTED SUPPORTED SUPPORTED SUPPORTED SUPPORTED
0000:df:00.0 SUPPORTED SUPPORTED SUPPORTED SUPPORTED SUPPORTED

```

5.5.9 9. Partition Information

```
$ sudo amd-smi partition --gpu=0 --current
```

Output:

```

GPU: 0
PARTITION:
ACCELERATOR_PARTITION: SPX
MEMORY_PARTITION: NPS1
PARTITION_ID: 0

```

Get per-partition metrics:

```
$ sudo amd-smi metric --vf=0:0 --per-partition
```

Output:

```

GPU: 0
VF: 0
PER_PARTITION:
AID_0:
CLK_VCLK: 29 MHz
CLK_VCLK_MIN_LIMIT: 914 MHz
CLK_VCLK_MAX_LIMIT: 1333 MHz
CLK_DCLK_LIMIT: 22 MHz
CLK_DCLK_MIN_LIMIT: 711 MHz
CLK_DCLK_MAX_LIMIT: 1142 MHz
CLK_SCLK_LIMIT: 28 MHz
CLK_SCLK_MIN_LIMIT: 888 MHz
CLK_SCLK_MAX_LIMIT: 1142 MHz
VCN_ACTIVITY: 0 %
JPEG_ACTIVITY: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %]
AID_1:
CLK_VCLK: 29 MHz
CLK_VCLK_MIN_LIMIT: 914 MHz
CLK_VCLK_MAX_LIMIT: 1333 MHz
CLK_DCLK_LIMIT: 22 MHz
CLK_DCLK_MIN_LIMIT: 711 MHz
CLK_DCLK_MAX_LIMIT: 1142 MHz
CLK_SCLK_LIMIT: 28 MHz
CLK_SCLK_MIN_LIMIT: 888 MHz
CLK_SCLK_MAX_LIMIT: 1142 MHz
VCN_ACTIVITY: 0 %
JPEG_ACTIVITY: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %]
AID_2:

```

(continues on next page)

(continued from previous page)

```

CLK_VCLK: 29 MHz
CLK_VCLK_MIN_LIMIT: 914 MHz
CLK_VCLK_MAX_LIMIT: 1333 MHz
CLK_DCLK_LIMIT: 22 MHz
CLK_DCLK_MIN_LIMIT: 711 MHz
CLK_DCLK_MAX_LIMIT: 1142 MHz
CLK_SCLK_LIMIT: 28 MHz
CLK_SCLK_MIN_LIMIT: 888 MHz
CLK_SCLK_MAX_LIMIT: 1142 MHz
VCN_ACTIVITY: 0 %
JPEG_ACTIVITY: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %]
AID_3:
CLK_VCLK: 29 MHz
CLK_VCLK_MIN_LIMIT: 914 MHz
CLK_VCLK_MAX_LIMIT: 1333 MHz
CLK_DCLK_LIMIT: 22 MHz
CLK_DCLK_MIN_LIMIT: 711 MHz
CLK_DCLK_MAX_LIMIT: 1142 MHz
CLK_SCLK_LIMIT: 28 MHz
CLK_SCLK_MIN_LIMIT: 888 MHz
CLK_SCLK_MAX_LIMIT: 1142 MHz
VCN_ACTIVITY: 0 %
JPEG_ACTIVITY: [0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %, 0 %]
XCP_0:
GFX_CLK: [132 MHz]
GFX_MIN_CLK: [500 MHz]
GFX_MAX_CLK: [2100 MHz]
GFX_CLK_LOCKED: [DISABLED]
GFX_USAGE: [0 %]
XCP_1:
GFX_CLK: [132 MHz]
GFX_MIN_CLK: [500 MHz]
GFX_MAX_CLK: [2100 MHz]
GFX_CLK_LOCKED: [DISABLED]
GFX_USAGE: [0 %]
XCP_2:
GFX_CLK: [132 MHz]
GFX_MIN_CLK: [500 MHz]
GFX_MAX_CLK: [2100 MHz]
GFX_CLK_LOCKED: [DISABLED]
GFX_USAGE: [0 %]
XCP_3:
GFX_CLK: [132 MHz]
GFX_MIN_CLK: [500 MHz]
GFX_MAX_CLK: [2100 MHz]
GFX_CLK_LOCKED: [DISABLED]
GFX_USAGE: [0 %]
XCP_4:
GFX_CLK: [132 MHz]
GFX_MIN_CLK: [500 MHz]
GFX_MAX_CLK: [2100 MHz]
GFX_CLK_LOCKED: [DISABLED]

```

(continues on next page)

(continued from previous page)

```

    GFX_USAGE: [0 %]
XCP_5:
    GFX_CLK: [132 MHz]
    GFX_MIN_CLK: [500 MHz]
    GFX_MAX_CLK: [2100 MHz]
    GFX_CLK_LOCKED: [DISABLED]
    GFX_USAGE: [0 %]
XCP_6:
    GFX_CLK: [132 MHz]
    GFX_MIN_CLK: [500 MHz]
    GFX_MAX_CLK: [2100 MHz]
    GFX_CLK_LOCKED: [DISABLED]
    GFX_USAGE: [0 %]
XCP_7:
    GFX_CLK: [132 MHz]
    GFX_MIN_CLK: [500 MHz]
    GFX_MAX_CLK: [2100 MHz]
    GFX_CLK_LOCKED: [DISABLED]
    GFX_USAGE: [0 %]

```

Get global partition configuration:

```
$ sudo amd-smi partition --global
```

Output:

```

GLOBAL_PARTITION_CONFIG:
GPU ACCELERATOR_TYPE SUPPORTED_VF_MODE MEMORY_PARTITION_CAPS
0    SPX                1                NPS1
    DPX                2                NPS1
    QPX                4                NPS1,NPS4
    CPX                1,2,4,8         NPS1,NPS4
1    SPX                1                NPS1
    DPX                2                NPS1
    QPX                4                NPS1,NPS4
    CPX                1,2,4,8         NPS1,NPS4
2    SPX                1                NPS1
    DPX                2                NPS1
    QPX                4                NPS1,NPS4
    CPX                1,2,4,8         NPS1,NPS4
3    SPX                1                NPS1
    DPX                2                NPS1
    QPX                4                NPS1,NPS4
    CPX                1,2,4,8         NPS1,NPS4
4    SPX                1                NPS1
    DPX                2                NPS1
    QPX                4                NPS1,NPS4
    CPX                1,2,4,8         NPS1,NPS4
5    SPX                1                NPS1
    DPX                2                NPS1
    QPX                4                NPS1,NPS4
    CPX                1,2,4,8         NPS1,NPS4

```

(continues on next page)

(continued from previous page)

6	SPX	1	NPS1
	DPX	2	NPS1
	QPX	4	NPS1,NPS4
	CPX	1,2,4,8	NPS1,NPS4
7	SPX	1	NPS1
	DPX	2	NPS1
	QPX	4	NPS1,NPS4
	CPX	1,2,4,8	NPS1,NPS4

5.5.10 10. Monitor Command (Continuous Monitoring)

```
$ sudo amd-smi monitor
```

Output (updates every second):

GPU	POWER	HOTSPOT_TEMP	MEM_TEMP	GFX_UTIL	GFX_CLOCK	MEM_UTIL	MEM_CLOCK	ENC_UTIL
↔	VCLK	DEC_UTIL	DCLK	CORRECTABLE_ECC	UNCORRECTABLE_ECC	PCIE_REPLAY	PCIE_BW	
0	156 W	38 C	33 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	18 Mb/s
1	153 W	38 C	31 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	44 Mb/s
2	149 W	35 C	30 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	18 Mb/s
3	140 W	36 C	31 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	18 Mb/s
4	149 W	33 C	31 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	18 Mb/s
5	151 W	39 C	33 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	41 Mb/s
6	140 W	35 C	31 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	18 Mb/s
7	140 W	38 C	33 C	0 %	132 MHz	0 %	900 MHz	0 %
↔	29 MHz	0 %	22 MHz	0	0	0	0	18 Mb/s

5.5.11 11. Set Commands

Set power cap:

```
$ sudo amd-smi set --gpu=0 --power-cap=600
```

Output:

```
GPU: 0
POWER_CAP: Successfully set power cap to 600 W
```

Set memory partition:

```
$ sudo amd-smi set --memory-partition=NPS2
```

Output

```
Setting memory-partition in progress. This may take a while...
```

```
GPU: 0
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 1
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 2
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 3
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 4
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 5
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 6
    MEMORY_PARTITION: Successfully set memory partition to NPS2
GPU: 7
    MEMORY_PARTITION: Successfully set memory partition to NPS2
```

Set number of VFs enabled:

```
$ sudo amd-smi set --num-vf=1
```

Output

```
GPU: 0
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 1
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 2
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 3
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 4
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 5
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 6
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
GPU: 7
    NUM_VF_ENABLED: Successfully set enabled VFs to 1
```

Set XGMI Per-Link Down Policy

```
$ sudo amd-smi set --xgmi-plpd=0
```

Output

```
GPU: 0
    DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 1
    DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 2
```

(continues on next page)

(continued from previous page)

```
DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 3
DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 4
DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 5
DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 6
DPM_POLICY: Successfully set xgmi per-link power down policy to 0
GPU: 7
DPM_POLICY: Successfully set xgmi per-link power down policy to 0
```

Set SOC Pstate

```
$ sudo amd-smi set --soc-pstate=0
```

Output

```
GPU: 0
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 1
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 2
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 3
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 4
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 5
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 6
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
GPU: 7
SOC_PSTATE: Successfully set dpm soc pstate policy to 0
```

Set PTL format

```
$ sudo amd-smi set --ptl-format=I8,F32 --gpu=0
```

Output

```
GPU: 0
PTL_FORMAT: Successfully set PTL preferred formats to I8,F32
```

Set PTL status

```
$ sudo amd-smi set --ptl-status=DISABLED --gpu=0
```

Output

```
GPU: 0
PTL_STATUS: Successfully set PTL status to DISABLED
```

Set XGMI FB Sharing Mode

```
$ sudo amd-smi set --xgmi --fb-sharing-mode=MODE_1
```

Output

```
XGMI FB_SHARING_MODE: Successfully set mode to MODE_1 for the given group/s
```

5.5.12 12. Reset Commands**Reset GPU:**

```
$ sudo amd-smi reset --gpureset
```

Output

```
GPU: 0
GPU_RESET: Successfully reset GPU
GPU: 1
GPU_RESET: Successfully reset GPU
GPU: 2
GPU_RESET: Successfully reset GPU
GPU: 3
GPU_RESET: Successfully reset GPU
GPU: 4
GPU_RESET: Successfully reset GPU
GPU: 5
GPU_RESET: Successfully reset GPU
GPU: 6
GPU_RESET: Successfully reset GPU
GPU: 7
GPU_RESET: Successfully reset GPU
```

Clean VF framebuffer:

```
$ sudo amd-smi reset --vf=0:0 --vf-fb
```

Output

```
Successfully reset vf fb for vf with id: 0:0
```

5.5.13 13. RAS Error Information (MI300 Host Systems Only)**Get CPER entries with fatal severity:**

```
$ sudo amd-smi ras --cper --severity=fatal --folder=/tmp/ras_logs
```

Output:

timestamp	gpu_id	severity	file_name	list of ↵
↵afids				
07/10/2025 09:03:09	0	fatal	fatal-1.cper	24
07/10/2025 09:04:15	1	fatal	fatal-2.cper	24 29

Get all CPER entries with continuous monitoring:

```
$ sudo amd-smi ras --cper --severity=all --folder=/tmp/ras_logs --file-limit=10 --follow
```

Extract AFID from existing CPER file:

```
$ sudo amd-smi ras --afid --cper-file=/tmp/ras_logs/fatal-2.cper
```

Output:

```
24 29
```

Get RAS policy info:

```
$ sudo amd-smi ras [--policy]
```

Output:

```
POLICY:
  MINOR_VERSION: 4
  MAJOR_VERSION: 0
  DRAM_NON_CRITICAL_REGION_THRESHOLD: int(val)
  DRAM_CRITICAL_REGION_THRESHOLD: int(val)
```

5.5.14 14. JSON and CSV Format Examples

JSON format for metrics:

```
$ sudo amd-smi metric --pcie --gpu=0 --json
```

Output:

```
[
  {
    "gpu": 0,
    "pcie": {
      "width": 16,
      "speed": {
        "value": 32,
        "unit": "GT/s"
      },
      "bandwidth": {
        "value": 18,
        "unit": "Mb/s"
      },
      "replay_count": 0,
      "l0_to_recovery_count": 0,
      "replay_roll_over_count": 0,
      "nak_sent_count": 0,
      "nak_received_count": 0
    }
  }
]
```

CSV format for usage:

```
$ sudo amd-smi metric --pcie --gpu=0 --csv
```

Output:

```
gpu,pcie_current_width,pcie_current_speed,pcie_current_bandwidth,pcie_replay_count,pcie_
↳l0_to_recovery_count,pcie_replay_roll_over_count,pcie_nak_sent_count,pcie_nak_received_
↳count
0,16,32,18,0,0,0,0,0
```

5.5.15 15. NIC Information and Monitoring**List NICs:**

```
$ sudo amd-smi list --nic
```

Output:

```
NIC: 0
BDF: 0000:41:00.0
PERMANENT_ADDRESS: 04:90:81:2c:7b:10
PRODUCT_NAME: POLLARA 1x400G QSFP112
PART_NUMBER: POLLARA-1Q400P
SERIAL_NUMBER: FPL250300C5EC0V2
VENDOR_NAME: AMD Pensando Systems, Inc.
```

Get NIC static information:

```
$ sudo amd-smi static --nic=0
```

Output:

```
NIC: 0
ASIC:
  VENDOR_ID: 0x1DD8
  SUBVENDOR_ID: 0x1DD8
  DEVICE_ID: 0x8
  SUBSYSTEM_ID: 0x5201
  REVISION: 0x0
  PERMANENT_ADDRESS: 04:90:81:2c:7b:10
  PRODUCT_NAME: POLLARA 1x400G QSFP112
  PART_NUMBER: POLLARA-1Q400P
  SERIAL_NUMBER: FPL250300C5EC0V2
  VENDOR_NAME: AMD Pensando Systems, Inc.
BUS:
  BDF: 0000:41:00.0
  MAX_PCIE_WIDTH: 16
  MAX_PCIE_SPEED: 32 GT/s
  PCIE_INTERFACE_VERSION: N/A
  SLOT_TYPE: N/A
DRIVER:
  NAME: ionic
  VERSION: 25.08.1.001
NUMA:
```

(continues on next page)

(continued from previous page)

```

NODE: 0
AFFINITY: [0-95,192-287]
PORTS:
  PORT_0:
    BDF: 0000:44:00.0
    PORT_NUM: 0
    TYPE: Ethernet
    FLAVOUR: N/A
    NETDEV: enp68s0
    IFINDEX: 48
    MAC_ADDRESS: 04:90:81:2c:7b:10
    CARRIER: 0
    MTU: 1500 B
    LINK_STATE: DOWN
    LINK_SPEED: 1000000 Mb/s
    ACTIVE_FEC: 4
    AUTONEG: DISABLED
    PAUSE_AUTONEG: OFF
    PAUSE_RX: OFF
    PAUSE_TX: OFF
RDMA_DEVICES:
  RDMA_DEVICE_0:
    RDMA_DEV: rocep68s0
    NODE_GUID: 0690:81ff:fe2c:7b10
    NODE_TYPE: CA
    SYS_IMAGE_GUID: 0690:81ff:fe2c:7b10
    FW_VER: 1.117.2-a-11
    PORTS:
      PORT_0:
        NETDEV: enp68s0
        STATE: DOWN
        RDMA_PORT: 1
        MAX_MTU: N/A
        ACTIVE_MTU: N/A

```

Get NIC RDMA devices statistics:

```
$ sudo amd-smi metric --nic=0 --rdma_devices
```

Output:

```

NIC: 0
RDMA_DEVICES:
  RDMA_DEVICE_0:
    RDMA_DEV: rocep68s0
    PORTS:
      PORT_0:
        STATISTICS:
          LIFESPAN: 10
          REQ_RX_CQE_ERR: 0
          REQ_RX_CQE_FLUSH: 0
          REQ_RX_DUP_RESPONSE: 0

```

(continues on next page)

(continued from previous page)

```

REQ_RX_IMPL_NAK_SEQ_ERR: 0
REQ_RX_INVALID_PKTS: 0
REQ_RX_OPER_ERR: 0
REQ_RX_PKT_SEQ_ERR: 0
REQ_RX_RMT_ACC_ERR: 0
REQ_RX_RMT_REQ_ERR: 0
REQ_RX_RNR_RETRY_ERR: 0
REQ_TX_LOC_ACC_ERR: 0
REQ_TX_LOC_OPER_ERR: 0
REQ_TX_LOC_SGL_INV_ERR: 0
REQ_TX_MEM_MGMT_ERR: 0
REQ_TX_RETRY_EXCD_ERR: 0
RESP_RX_CQE_ERR: 0
RESP_RX_CQE_FLUSH: 0
RESP_RX_DUP_REQUEST: 0
RESP_RX_INVALID_REQUEST: 0
RESP_RX_LOC_LEN_ERR: 0
RESP_RX_LOC_OPER_ERR: 0
RESP_RX_OUTOF_ATOMIC: 0
RESP_RX_OUTOF_BUF: 0
RESP_RX_OUTOF_SEQ: 0
RESP_RX_S0_TABLE_ERR: 0
RESP_TX_LOC_SGL_INV_ERR: 0
RESP_TX_PKT_SEQ_ERR: 0
RESP_TX_RMT_ACC_ERR: 0
RESP_TX_RMT_INVALID_REQ_ERR: 0
RESP_TX_RMT_OPER_ERR: 0
RESP_TX_RNR_RETRY_ERR: 0
RX_RDMA_CNP_PKTS: 0
RX_RDMA_ECN_PKTS: 0
RX_RDMA_MCAST_BYTES: 0
RX_RDMA_MCAST_PKTS: 0
RX_RDMA_UCAST_BYTES: 0
RX_RDMA_UCAST_PKTS: 0
TX_RDMA_CNP_PKTS: 0
TX_RDMA_MCAST_BYTES: 0
TX_RDMA_MCAST_PKTS: 0
TX_RDMA_UCAST_BYTES: 0
TX_RDMA_UCAST_PKTS: 0

```

Get NIC ports statistics:

```
$ sudo amd-smi metric --nic=0 --port
```

Output:

```

NIC: 0
  PORTS:
    PORT_0:
      NETDEV: enp153s0
      VENDOR_STATISTICS:
        RX_BYTES: 0

```

(continues on next page)

(continued from previous page)

```

RX_CSUM_NONE: 0
RX_PACKETS: 0
TX_BYTES: 0
TX_CSUM: 0
TX_CSUM_NONE: 0
TX_PACKETS: 0
TX_TSO: 0
TX_TSO_BYTES: 0
STATISTICS:
COLLISIONS: 0
MULTICAST: 0
RX_BYTES: 0
RX_COMPRESSED: 0
RX_CRC_ERRORS: 0
RX_DROPPED: 0
RX_ERRORS: 0
RX_FIFO_ERRORS: 0
RX_FRAME_ERRORS: 0
RX_LENGTH_ERRORS: 0
RX_MISSED_ERRORS: 0
RX_NOHANDLER: 0
RX_OVER_ERRORS: 0
RX_PACKETS: 0
TX_ABORTED_ERRORS: 0
TX_BYTES: 0
TX_CARRIER_ERRORS: 0
TX_COMPRESSED: 0
TX_DROPPED: 0
TX_ERRORS: 0
TX_FIFO_ERRORS: 0
TX_HEARTBEAT_ERRORS: 0
TX_PACKETS: 0
TX_WINDOW_ERRORS: 0

```

Get NIC and GPU static information:

```
$ sudo amd-smi static --gpu=0 --gpu=1 --gpu=2 --nic=0
```

Output

```

GPU: 0
  ASIC:
    MARKET_NAME: AMD Instinct MI350X
    VENDOR_ID: 0x1002
    VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
    SUBVENDOR_ID: 0x1002
    DEVICE_ID: 0x75A0
    SUBSYSTEM_ID: 0x75A0
    REV_ID: 0x0
    ASIC_SERIAL: 0xDB992407CFE01E26
    OAM_ID: 6
    NUM_OF_COMPUTE_UNITS: 256

```

(continues on next page)

(continued from previous page)

```
BUS:
  BDF: 0000:05:00.0
  MAX_PCIE_WIDTH: 16
  MAX_PCIE_SPEED: 32 GT/s
  PCIE_INTERFACE_VERSION: Gen 4
  SLOT_TYPE: OAM
  MAX_PCIE_INTERFACE_VERSION: Gen 5
IFWI:
  NAME: AMD MI350X
  BUILD_DATE: 2025/08/31 23:49
  PART_NUMBER: 113-M350-01-1K0-980C
  VERSION: 023.040.001.008.000001
  BOOT_FIRMWARE: N/A
LIMIT:
  MAX_POWER: 1000 W
  MIN_POWER: 0 W
  SOCKET_POWER: 1000 W
  SLOWDOWN_EDGE_TEMPERATURE: N/A
  SLOWDOWN_HOTSPOT_TEMPERATURE: 100 C
  SLOWDOWN_MEM_TEMPERATURE: 115 C
  SHUTDOWN_EDGE_TEMPERATURE: N/A
  SHUTDOWN_HOTSPOT_TEMPERATURE: 110 C
  SHUTDOWN_MEM_TEMPERATURE: 125 C
  PTL: ENABLED
  PTL_FORMAT: I8,BF16
DRIVER:
  NAME: libgv
  VERSION: 8.6.0.K-dev-107-1dfb7bfa73
  DATE: 2025-07-24 10:19:16
  MODEL: N/A
BOARD:
  MODEL_NUMBER: 102-G36212-0C
  PRODUCT_SERIAL: 692517020426
  FRU_ID: 113-AMDG362120C01-100-300000082
  PRODUCT_NAME: AMD Instinct MI350 OAM
  MANUFACTURER_NAME: AMD
RAS:
  EEPROM_VERSION: 0x30000
  BAD_PAGE_THRESHOLD: 128
  PARITY_SCHEMA: ENABLED
  SINGLE_BIT_SCHEMA: ENABLED
  DOUBLE_BIT_SCHEMA: ENABLED
  POISON_SCHEMA: ENABLED
  BLOCK_STATE:
    UMC: ENABLED
    SDMA: ENABLED
    GFX: ENABLED
    MMHUB: ENABLED
    ATHUB: DISABLED
    PCIE_BIF: ENABLED
    HDP: DISABLED
    XGMI_WAFL: ENABLED
```

(continues on next page)

(continued from previous page)

```
DF: DISABLED
SMN: DISABLED
SEM: DISABLED
MP0: DISABLED
MP1: DISABLED
FUSE: DISABLED
MCA: DISABLED
VCN: DISABLED
JPEG: DISABLED
IH: DISABLED
MPIO: DISABLED
FB_INFO:
  TOTAL_FB_SIZE: 257752 MB
  PF_FB_RESERVED: 16 MB
  PF_FB_OFFSET: 0 MB
  FB_ALIGNMENT: 16 MB
  MAX_VF_FB_USABLE: 257736 MB
  MIN_VF_FB_USABLE: 16 MB
NUM_VF:
  SUPPORTED: 1
  ENABLED: 1
VRAM:
  TYPE: HBM3E
  VENDOR: SAMSUNG
  SIZE: 258032 MB
  BIT_WIDTH: 0
  MAX_BANDWIDTH: 6810 GB/s
CACHE_INFO:
  CACHE_0:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 32 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 1
    NUM_CACHE_INSTANCE: 256
  CACHE_1:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 16 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 128
  CACHE_2:
    CACHE_PROPERTIES: INST_CACHE, SIMD_CACHE
    CACHE_SIZE: 64 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 128
  CACHE_3:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 4096 KB
    CACHE_LEVEL: 2
    MAX_NUM_CU_SHARED: 32
    NUM_CACHE_INSTANCE: 8
```

(continues on next page)

(continued from previous page)

```

CACHE_4:
  CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
  CACHE_SIZE: 262144 KB
  CACHE_LEVEL: 3
  MAX_NUM_CU_SHARED: 256
  NUM_CACHE_INSTANCE: 1
XGMI_PLPD:
  NUM_SUPPORTED: 3
  CURRENT_ID: 1
  POLICIES:
    POLICY_ID: 0
    POLICY_DESCRIPTION: plpd_disallow
    POLICY_ID: 1
    POLICY_DESCRIPTION: plpd_default
    POLICY_ID: 2
    POLICY_DESCRIPTION: plpd_optimized
PARTITION:
  ACCELERATOR_PARTITION: SPX
  MEMORY_PARTITION: NPS1
  PARTITION_ID: 0
MODE: HOST
NUMA:
  NODE: 0
  CPU_AFFINITY:
    CPU_LIST_0:
      BITMASK: ffffffffffffffff
      CORE_RANGE: [0-63]
    CPU_LIST_1:
      BITMASK: 0000000000000000
      CORE_RANGE: []
    CPU_LIST_2:
      BITMASK: ffffffffffffffff
      CORE_RANGE: [128-191]
    CPU_LIST_3:
      BITMASK: 0000000000000000
      CORE_RANGE: []
  SOCKET_AFFINITY: N/A
GPU: 1
ASIC:
  MARKET_NAME: AMD Instinct MI350X
  VENDOR_ID: 0x1002
  VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
  SUBVENDOR_ID: 0x1002
  DEVICE_ID: 0x75A0
  SUBSYSTEM_ID: 0x75A0
  REV_ID: 0x0
  ASIC_SERIAL: 0x5F002195F0DFC795
  OAM_ID: 7
  NUM_OF_COMPUTE_UNITS: 256
BUS:
  BDF: 0000:15:00.0
  MAX_PCIE_WIDTH: 16

```

(continues on next page)

(continued from previous page)

```
MAX_PCIE_SPEED: 32 GT/s
PCIE_INTERFACE_VERSION: Gen 4
SLOT_TYPE: OAM
MAX_PCIE_INTERFACE_VERSION: Gen 5
IFWI:
NAME: AMD MI350X
BUILD_DATE: 2025/08/31 23:49
PART_NUMBER: 113-M350-01-1K0-980C
VERSION: 023.040.001.008.000001
BOOT_FIRMWARE: N/A
LIMIT:
MAX_POWER: 1000 W
MIN_POWER: 0 W
SOCKET_POWER: 1000 W
SLOWDOWN_EDGE_TEMPERATURE: N/A
SLOWDOWN_HOTSPOT_TEMPERATURE: 100 C
SLOWDOWN_MEM_TEMPERATURE: 115 C
SHUTDOWN_EDGE_TEMPERATURE: N/A
SHUTDOWN_HOTSPOT_TEMPERATURE: 110 C
SHUTDOWN_MEM_TEMPERATURE: 125 C
PTL: ENABLED
PTL_FORMAT: I8,BF16
DRIVER:
NAME: libgv
VERSION: 8.6.0.K-dev-107-1dfb7bfa73
DATE: 2025-07-24 10:19:16
MODEL: N/A
BOARD:
MODEL_NUMBER: 102-G36212-0C
PRODUCT_SERIAL: 692517020489
FRU_ID: 113-AMDG362120C01-100-300000082
PRODUCT_NAME: AMD Instinct MI350 OAM
MANUFACTURER_NAME: AMD
RAS:
EEPROM_VERSION: 0x30000
BAD_PAGE_THRESHOLD: 128
PARITY_SCHEMA: ENABLED
SINGLE_BIT_SCHEMA: ENABLED
DOUBLE_BIT_SCHEMA: ENABLED
POISON_SCHEMA: ENABLED
BLOCK_STATE:
  UMC: ENABLED
  SDMA: ENABLED
  GFX: ENABLED
  MMHUB: ENABLED
  ATHUB: DISABLED
  PCIE_BIF: ENABLED
  HDP: DISABLED
  XGMI_WAFL: ENABLED
  DF: DISABLED
  SMN: DISABLED
  SEM: DISABLED
```

(continues on next page)

(continued from previous page)

```
MP0: DISABLED
MP1: DISABLED
FUZE: DISABLED
MCA: DISABLED
VCN: DISABLED
JPEG: DISABLED
IH: DISABLED
MPIO: DISABLED
FB_INFO:
  TOTAL_FB_SIZE: 257752 MB
  PF_FB_RESERVED: 16 MB
  PF_FB_OFFSET: 0 MB
  FB_ALIGNMENT: 16 MB
  MAX_VF_FB_USABLE: 257736 MB
  MIN_VF_FB_USABLE: 16 MB
NUM_VF:
  SUPPORTED: 1
  ENABLED: 1
VRAM:
  TYPE: HBM3E
  VENDOR: SAMSUNG
  SIZE: 258032 MB
  BIT_WIDTH: 0
  MAX_BANDWIDTH: 6810 GB/s
CACHE_INFO:
  CACHE_0:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 32 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 1
    NUM_CACHE_INSTANCE: 256
  CACHE_1:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 16 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 128
  CACHE_2:
    CACHE_PROPERTIES: INST_CACHE, SIMD_CACHE
    CACHE_SIZE: 64 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 128
  CACHE_3:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 4096 KB
    CACHE_LEVEL: 2
    MAX_NUM_CU_SHARED: 32
    NUM_CACHE_INSTANCE: 8
  CACHE_4:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 262144 KB
```

(continues on next page)

(continued from previous page)

```

    CACHE_LEVEL: 3
    MAX_NUM_CU_SHARED: 256
    NUM_CACHE_INSTANCE: 1
XGMI_PLPD:
    NUM_SUPPORTED: 3
    CURRENT_ID: 1
    POLICIES:
        POLICY_ID: 0
        POLICY_DESCRIPTION: plpd_disallow
        POLICY_ID: 1
        POLICY_DESCRIPTION: plpd_default
        POLICY_ID: 2
        POLICY_DESCRIPTION: plpd_optimized
PARTITION:
    ACCELERATOR_PARTITION: SPX
    MEMORY_PARTITION: NPS1
    PARTITION_ID: 0
MODE: HOST
NUMA:
    NODE: 0
    CPU_AFFINITY:
        CPU_LIST_0:
            BITMASK: ffffffffffffffff
            CORE_RANGE: [0-63]
        CPU_LIST_1:
            BITMASK: 0000000000000000
            CORE_RANGE: []
        CPU_LIST_2:
            BITMASK: ffffffffffffffff
            CORE_RANGE: [128-191]
        CPU_LIST_3:
            BITMASK: 0000000000000000
            CORE_RANGE: []
    SOCKET_AFFINITY: N/A
GPU: 2
ASIC:
    MARKET_NAME: AMD Instinct MI350X
    VENDOR_ID: 0x1002
    VENDOR_NAME: Advanced Micro Devices Inc. [AMD/ATI]
    SUBVENDOR_ID: 0x1002
    DEVICE_ID: 0x75A0
    SUBSYSTEM_ID: 0x75A0
    REV_ID: 0x0
    ASIC_SERIAL: 0x20285E597D0C8872
    OAM_ID: 5
    NUM_OF_COMPUTE_UNITS: 256
BUS:
    BDF: 0000:65:00.0
    MAX_PCIE_WIDTH: 16
    MAX_PCIE_SPEED: 32 GT/s
    PCIE_INTERFACE_VERSION: Gen 4
    SLOT_TYPE: OAM

```

(continues on next page)

(continued from previous page)

```

MAX_PCIE_INTERFACE_VERSION: Gen 5
IFWI:
  NAME: AMD MI350X
  BUILD_DATE: 2025/08/31 23:49
  PART_NUMBER: 113-M350-01-1K0-980C
  VERSION: 023.040.001.008.000001
  BOOT_FIRMWARE: N/A
LIMIT:
  MAX_POWER: 1000 W
  MIN_POWER: 0 W
  SOCKET_POWER: 1000 W
  SLOWDOWN_EDGE_TEMPERATURE: N/A
  SLOWDOWN_HOTSPOT_TEMPERATURE: 100 C
  SLOWDOWN_MEM_TEMPERATURE: 115 C
  SHUTDOWN_EDGE_TEMPERATURE: N/A
  SHUTDOWN_HOTSPOT_TEMPERATURE: 110 C
  SHUTDOWN_MEM_TEMPERATURE: 125 C
  PTL: ENABLED
  PTL_FORMAT: I8,BF16
DRIVER:
  NAME: libgv
  VERSION: 8.6.0.K-dev-107-1dfb7bfa73
  DATE: 2025-07-24 10:19:16
  MODEL: N/A
BOARD:
  MODEL_NUMBER: 102-G36212-0C
  PRODUCT_SERIAL: 692517020515
  FRU_ID: 113-AMDG362120C01-100-300000082
  PRODUCT_NAME: AMD Instinct MI350 OAM
  MANUFACTURER_NAME: AMD
RAS:
  EEPROM_VERSION: 0x30000
  BAD_PAGE_THRESHOLD: 128
  PARITY_SCHEMA: ENABLED
  SINGLE_BIT_SCHEMA: ENABLED
  DOUBLE_BIT_SCHEMA: ENABLED
  POISON_SCHEMA: ENABLED
  BLOCK_STATE:
    UMC: ENABLED
    SDMA: ENABLED
    GFX: ENABLED
    MMHUB: ENABLED
    ATHUB: DISABLED
    PCIE_BIF: ENABLED
    HDP: DISABLED
    XGMI_WAFL: ENABLED
    DF: DISABLED
    SMN: DISABLED
    SEM: DISABLED
    MP0: DISABLED
    MP1: DISABLED
    FUSE: DISABLED

```

(continues on next page)

(continued from previous page)

```
MCA: DISABLED
VCN: DISABLED
JPEG: DISABLED
IH: DISABLED
MPIO: DISABLED
FB_INFO:
  TOTAL_FB_SIZE: 257752 MB
  PF_FB_RESERVED: 16 MB
  PF_FB_OFFSET: 0 MB
  FB_ALIGNMENT: 16 MB
  MAX_VF_FB_USABLE: 257736 MB
  MIN_VF_FB_USABLE: 16 MB
NUM_VF:
  SUPPORTED: 1
  ENABLED: 1
VRAM:
  TYPE: HBM3E
  VENDOR: SAMSUNG
  SIZE: 258032 MB
  BIT_WIDTH: 0
  MAX_BANDWIDTH: 6810 GB/s
CACHE_INFO:
  CACHE_0:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 32 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 1
    NUM_CACHE_INSTANCE: 256
  CACHE_1:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 16 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 128
  CACHE_2:
    CACHE_PROPERTIES: INST_CACHE, SIMD_CACHE
    CACHE_SIZE: 64 KB
    CACHE_LEVEL: 1
    MAX_NUM_CU_SHARED: 2
    NUM_CACHE_INSTANCE: 128
  CACHE_3:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 4096 KB
    CACHE_LEVEL: 2
    MAX_NUM_CU_SHARED: 32
    NUM_CACHE_INSTANCE: 8
  CACHE_4:
    CACHE_PROPERTIES: DATA_CACHE, SIMD_CACHE
    CACHE_SIZE: 262144 KB
    CACHE_LEVEL: 3
    MAX_NUM_CU_SHARED: 256
    NUM_CACHE_INSTANCE: 1
```

(continues on next page)

(continued from previous page)

```

XGMI_PLPD:
  NUM_SUPPORTED: 3
  CURRENT_ID: 1
  POLICIES:
    POLICY_ID: 0
    POLICY_DESCRIPTION: plpd_disallow
    POLICY_ID: 1
    POLICY_DESCRIPTION: plpd_default
    POLICY_ID: 2
    POLICY_DESCRIPTION: plpd_optimized
  PARTITION:
    ACCELERATOR_PARTITION: SPX
    MEMORY_PARTITION: NPS1
    PARTITION_ID: 0
  MODE: HOST
  NUMA:
    NODE: 0
    CPU_AFFINITY:
      CPU_LIST_0:
        BITMASK: ffffffffffffffff
        CORE_RANGE: [0-63]
      CPU_LIST_1:
        BITMASK: 0000000000000000
        CORE_RANGE: []
      CPU_LIST_2:
        BITMASK: ffffffffffffffff
        CORE_RANGE: [128-191]
      CPU_LIST_3:
        BITMASK: 0000000000000000
        CORE_RANGE: []
    SOCKET_AFFINITY: N/A
  NIC: 0
  ASIC:
    VENDOR_ID: 0x1DD8
    SUBVENDOR_ID: 0x1DD8
    DEVICE_ID: 0x8
    SUBSYSTEM_ID: 0x5201
    REVISION: 0x0
    PERMANENT_ADDRESS: 04:90:81:34:12:e0
    PRODUCT_NAME: POLLARA 1x400G QSFP112
    PART_NUMBER: POLLARA-1Q400P
    SERIAL_NUMBER: FPL25170119EC0V2
    VENDOR_NAME: AMD Pensando Systems, Inc.
  BUS:
    BDF: 0000:96:00.0
    MAX_PCIE_WIDTH: 16
    MAX_PCIE_SPEED: 32 GT/s
    PCIE_INTERFACE_VERSION: N/A
    SLOT_TYPE: N/A
  DRIVER:
    NAME: ionic
    VERSION: 25.08.4.003

```

(continues on next page)

(continued from previous page)

```

NUMA:
  NODE: 1
  AFFINITY: [64-127,192-255]
PORTS:
  PORT_0:
    BDF: 0000:99:00.0
    PORT_NUM: 0
    TYPE: Ethernet
    FLAVOUR: N/A
    NETDEV: enp153s0
    IFINDEX: 10
    MAC_ADDRESS: 04:90:81:34:12:e0
    CARRIER: 0
    MTU: 1500 B
    LINK_STATE: DOWN
    LINK_SPEED: 4000000 Mb/s
    ACTIVE_FEC: 4
    AUTONEG: OFF
    PAUSE_AUTONEG: OFF
    PAUSE_RX: ON
    PAUSE_TX: ON
RDMA_DEVICES:
  RDMA_DEVICE_0:
    RDMA_DEV: ionic_0
    NODE_GUID: 0690:81ff:fe34:12e0
    NODE_TYPE: CA
    SYS_IMAGE_GUID: 0690:81ff:fe34:12e0
    FW_VER: 1.117.1-a-42
    PORTS:
      PORT_0:
        NETDEV: enp153s0
        STATE: DOWN
        RDMA_PORT: 1
        MAX_MTU: N/A
        ACTIVE_MTU: N/A

```

5.5.16 16. NODE informatins

```
$ sudo amd-smi node -p
```

```

NODE:
  POWER_MANAGEMENT:
    LIMIT: 7000 W
    STATUS: ENABLED

```

Use Case Scenarios

This section provides practical workflows for common administrative tasks using `**amd-smi**`.

Memory Partition Management

(continues on next page)

(continued from previous page)

```

**Scenario**: Configure memory partitioning modes

```bash
Step 1: Check current memory partition configuration
$ sudo amd-smi partition --gpu=0 --current

Step 2: List available memory partition modes
$ sudo amd-smi partition --gpu=0 --memory-partition

Step 3: Set memory partition to NPS2
$ sudo amd-smi set --memory-partition=NPS2

Step 4: Verify the partition change
$ sudo amd-smi partition --gpu=0 --current

```

### 5.5.17 Accelerator Partition Configuration

**Scenario**: Set up accelerator partitioning modes

```

Step 1: Check available accelerator partition profiles
$ sudo amd-smi partition --gpu=0 --accelerator-partition

Step 2: View current accelerator partition setting
$ sudo amd-smi partition --gpu=0 --current

Step 3: Set accelerator partition to profile 2 (example)
$ sudo amd-smi set --accelerator-partition=2

Step 4: Verify the partition configuration
$ sudo amd-smi partition --gpu=0 --current

```

### 5.5.18 XGMI Framebuffer Sharing Setup

**Scenario**: Configure framebuffer sharing

```

Step 1: Check XGMI capabilities and current configuration
$ sudo amd-smi xgmi --caps
$ sudo amd-smi xgmi --fb-sharing

Step 2: View topology to understand GPU connections
$ sudo amd-smi topology

Step 3: Set framebuffer sharing mode for 2-GPU group
$ sudo amd-smi set --xgmi --fb-sharing-mode=MODE_2

Step 4: Verify the framebuffer sharing configuration
$ sudo amd-smi topology --fb-sharing

```

## 5.5.19 System Health Monitoring

**Scenario:** Perform comprehensive system health check

```
Step 1: Check for bad pages and ECC errors
$ sudo amd-smi bad-pages
$ sudo amd-smi metric --ecc

Step 2: Monitor temperatures and power consumption
$ sudo amd-smi metric --temperature
$ sudo amd-smi metric --power

Step 3: Check thermal and power limits
$ sudo amd-smi static --limit

Step 4: Monitor real-time performance metrics (updates continuously)
$ sudo amd-smi monitor --gpu=0 --temperature --power-usage --ecc --watch=1
```

## 5.5.20 Performance Analysis Workflow

**Scenario:** Analyze GPU performance and utilization patterns

```
Step 1: Get baseline static information
$ sudo amd-smi static --gpu=0 --asic --vram --cache

Step 2: Monitor GPU utilization over time (updates every second for 60 seconds)
$ sudo amd-smi monitor --gpu=0 --gfx --mem --power-usage --watch=1 --watch-time=60

Step 3: Check detailed metrics for bottleneck analysis
$ sudo amd-smi metric --gpu=0 --usage --clock --temperature --pcie

Step 4: Export performance data for analysis
$ sudo amd-smi metric --gpu=0 --usage --clock --power --json > performance_data.json
```

## 5.5.21 VF Management in SR-IOV Environment

**Scenario:** Manage Virtual Functions for GPU virtualization

```
Step 1: List all GPUs and available VFs
$ sudo amd-smi list

Step 2: Check VF capabilities and current configuration
$ sudo amd-smi static --gpu=0 --num-vf

Step 3: Configure the number of VFs (if modification needed)
$ sudo amd-smi set --gpu=0 --num-vf=8

Step 4: Verify VF configuration after change
$ sudo amd-smi static --gpu=0 --num-vf

Step 5: Monitor VF performance metrics
$ sudo amd-smi metric --vf=0:0 --schedule --guard
```

## 5.5.22 Firmware and Driver Validation

**Scenario:** Validate firmware versions and driver compatibility

```
Step 1: Check current firmware versions
$ sudo amd-smi firmware --gpu=0 --fw-list

Step 2: Verify driver version and compatibility
$ sudo amd-smi static --gpu=0 --driver

Step 3: Check for firmware error records
$ sudo amd-smi firmware --gpu=0 --error-records
```

## 5.5.23 RAS Error Monitoring (MI300 Host Systems)

**Scenario:** Monitor and analyze hardware reliability errors

```
Step 1: Check for critical fatal errors and save to files
$ sudo amd-smi ras --cper --severity=fatal --folder=/var/log/gpu_errors --file-limit=50

Step 2: Monitor all error types with continuous monitoring
$ sudo amd-smi ras --cper --severity=all --folder=/var/log/gpu_errors --follow --file-
→limit=100

Step 3: Analyze existing error files to extract firmware component IDs
$ sudo amd-smi ras --afid --cper-file=/var/log/gpu_errors/fatal-1.cper

Step 4: Monitor specific GPU for non-fatal corrected errors
$ sudo amd-smi ras --gpu=0 --cper --severity=nonfatal-corrected --folder=/var/log/gpu0_
→errors
```

## 5.5.24 Network Interface Management

**Scenario:** Monitor and manage network interface performance

```
Step 1: List all available NICs on the system
$ sudo amd-smi list --nic

Step 2: Check NIC static configuration and capabilities
$ sudo amd-smi static --nic=0 --asic --port --bus

Step 3: Check network port statistics
$ sudo amd-smi metric --nic=0 --port
```

## 5.5.25 Multi-Device System Monitoring

**Scenario:** Comprehensive monitoring of GPUs and NICs in a multi-device system

```
Step 1: List all system devices (GPUs, VFs, and NICs)
$ sudo amd-smi list

Step 2: Generate comprehensive system report
$ sudo amd-smi static --json --gpu=0 > system_gpu_static.json
```

(continues on next page)

(continued from previous page)

```
$ sudo amd-smi metric --json --gpu=0 > system_gpu_metrics.json
$ sudo amd-smi static --json --nic=0 > system_nic_static.json
$ sudo amd-smi metric --json --nic=0 > system_nic_metrics.json
$ sudo amd-smi static --json > system_static.json
$ sudo amd-smi metric --json > system_metrics.json
```

## **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.

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

### **6.1 File List**

### **6.2 Globals**

### **6.3 Data Structures**

### **6.4 Data Fields**

#### **6.4.1 Data Fields**

#### **6.4.2 Data Fields - Variables**



## AMD SMI PYTHON API REFERENCE

Python interface – Consists of Python function declarations, which directly call the C interface. The client can use the Python interface to build applications in Python.

### 7.1 Requirements

- python 3.10+ 64-bit

### 7.2 Overview

#### 7.2.1 Folder structure

File Name	Note
<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 exceptions python file
<code>README.md</code>	Documentation

#### 7.2.2 Build steps

Navigate to project's root folder and run Makefile command:

- `make package`

Build process will create a folder `build/amdsmi/package/BUILD_MODE/amdsmi`, where `BUILD_MODE` can be Release or Debug. The folder will contain the following files:

- `__init__.py`
- `amdsmi_interface.py`
- `amdsmi_wrapper.py`
- `amdsmi_exception.py`
- `libamdsmi.so`
- `README.md`

## 7.3 Amdsmi usage

Generated amdsmi folder should be copied and placed next to importing script. It should be imported as:

```
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)
```

To initialize amdsmi lib, amdsmi\_init() must be called before all other calls to amdsmi lib.

To close connection to driver, amdsmi\_shut\_down() must be the last call.

## 7.4 Amdsmi Exceptions

All exceptions are in amdsmi\_exception.py file. Exceptions that can be thrown are:

- **AmdSmiException**: base smi exception class
- **AmdSmiLibraryException**: derives base **AmdSmiException** class and represents errors that can occur in smi-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 smi-lib and **err\_info** is a string that explains the error that occurred. Example:

```
try:
 amdsmi_init()
 processors = amdsmi_get_processor_handles()
 num_of_GPUs = len(processors)
 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 == AmdSmiRetCode.AMDSMI_STATUS_RETRY:
 print("Error info: {}".format(e.err_info))
finally:
 try:
 amdsmi_shut_down()
 except AmdSmiException as e:
 print(e)
```

- **AmdSmiRetryException** : Derives **AmdSmiLibraryException** class and signals processor 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.

## 7.5 Amdsmi API

### 7.5.1 amdsmi\_init

Description: Initialize smi lib and connect to driver

Input parameters: `init_flags` (*Optional parameter, if no value provided, default value is `AMDSMI_INIT_ALL_PROCESSORS` value*)

`init_flags` is `AmdSmiInitFlags` enum:

Field	Description
<code>INIT_ALL_PROCESSORS</code>	all processors
<code>INIT_AMD_CPUS</code>	amd cpus
<code>INIT_AMD_GPUS</code>	amd gpus
<code>INIT_NON_AMD_CPUS</code>	non amd cpus
<code>INIT_NON_AMD_GPUS</code>	non amd gpus
<code>INIT_AMD_APUS</code>	amd apus

Output: None

Exceptions that can be thrown by `amdsmi_init` function:

- `AmdSmiLibraryException`

Example:

```
try:
 amdsmi_init()
 # continue with amdsmi
except AmdSmiException as e:
 print("Init failed")
 print(e)
```

### 7.5.2 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_shut_down()
except AmdSmiException as e:
 print("Fini failed")
 print(e)
```

### 7.5.3 amdsmi\_get\_processor\_handles

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

Note: This function currently supports only AMD GPUs. To enumerate other devices, such as AMD NICs, use `amdsmi_get_processor_handles_by_type()`.

Input parameters: None

Output: List of GPU device handles

Exceptions that can be thrown by `amdsmi_get_processor_handles` function:

- `AmdSmiLibraryException`

Example:

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

### 7.5.4 amdsmi\_get\_processor\_handles\_by\_type

Description: Returns a list of processor handles of the specified type in the system.

Input parameters:

- `processor_type` one of `AmdSmiProcessorType` enum values:

Field	Description
UNKNOWN	Unknown processor type
AMD_GPU	AMD GPU device
AMD_CPU	AMD CPU device ( <i>Not supported yet</i> )
NON_AMD_GPU	Non-AMD GPU device ( <i>Not supported yet</i> )
NON_AMD_CPU	Non-AMD CPU device ( <i>Not supported yet</i> )
AMD_CPU_CORE	AMD CPU core ( <i>Not supported yet</i> )
AMD_APU	AMD Accelerated Processing Unit (APU) ( <i>Not supported yet</i> )
AMD_NIC	AMD Network Interface Card (NIC)

Output: List of processor handles of the chosen type

Exceptions that can be thrown by `amdsmi_get_processor_handles_by_type` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 processor_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_GPU)
 if len(processor_handles) == 0:
```

(continues on next page)

(continued from previous page)

```

 print("No GPUs on machine")
 else:
 for processor in processor_handles:
 print(amdsmi_get_gpu_device_uuid(processor))

 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 print(amdsmi_get_nic_asic_info(nic))
except AmdSmiException as e:
 print(e)

```

### 7.5.5 amdsmi\_get\_processor\_handle\_from\_bdf

Description: Returns processor 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: processor handle object

Exceptions that can be thrown by amdsmi\_get\_processor\_handle\_from\_bdf function:

- AmdSmiLibraryException
- AmdSmiBdfFormatException

Example:

```

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

```

### 7.5.6 amdsmi\_get\_gpu\_device\_bdf

Description: Returns BDF of the given processor (GPU PF).

Input parameters:

- GPU device 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:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
 print("GPU bdf:", amdsmi_get_gpu_device_bdf(processor))
except AmdSmiException as e:
 print(e)
```

### 7.5.7 `amdsmi_get_nic_device_bdf`

Description: Returns BDF of the given processor (NIC).

Input parameters:

- NIC device 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_nic_device_bdf` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
 print("NIC bdf:", amdsmi_get_nic_device_bdf(processor))
except AmdSmiException as e:
 print(e)
```

### 7.5.8 `amdsmi_get_processor_bdf`

Description: Returns BDF of the given processor.

Input parameters:

- Processor 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_processor_bdf` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
 print("processor bdf:", amdsmi_get_processor_bdf(processor))
except AmdSmiException as e:
 print(e)
```

### 7.5.9 `amdsmi_get_index_from_processor_handle`

Description: Returns the index of the given processor handle

Input parameters:

- GPU device for which to query

Output: GPU device index

Exceptions that can be thrown by `amdsmi_get_index_from_processor_handle` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 print("Processor's index:", amdsmi_get_index_from_processor_
handle(processor))
except AmdSmiException as e:
 print(e)
```

### 7.5.10 `amdsmi_get_processor_handle_from_index`

Description: Returns the processor handle from the given processor index

Input parameters:

- Function processor index to query

Output: processor handle object

Exceptions that can be thrown by `amdsmi_get_processor_handle_from_index` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 num_of_GPUs = len(processors)
 if num_of_GPUs == 0:
 print("No GPUs on machine")
 else:
 for index in range(num_of_GPUs):
 print("Processor handle:", amdsmi_get_processor_handle_from_index(index))
except AmdSmiException as e:
 print(e)

```

### 7.5.11 amdsmi\_get\_vf\_bdf

Description: Returns BDF of the given VF

Input parameters:

- VF 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\_vf\_bdf function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```

try:
 vf = amdsmi_get_vf_handle_from_bdf("0000:23:02.0")
 print("VF's bdf:", amdsmi_get_vf_bdf(vf))
except AmdSmiException as e:
 print(e)

```

### 7.5.12 amdsmi\_get\_vf\_handle\_from\_bdf

Description: Returns processor handle (VF) 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: processor handle object

Exceptions that can be thrown by amdsmi\_get\_vf\_handle\_from\_bdf function:

- AmdSmiLibraryException
- AmdSmiBdfFormatException

Example:

```
try:
 vf = amdsmi_get_vf_handle_from_bdf("0000:23:02.0")
 print(amdsmi_get_vf_uuid(vf))
except AmdSmiException as e:
 print(e)
```

### 7.5.13 amdsmi\_get\_processor\_handle\_from\_uuid

Description: Returns processor handle from the given UUID

Input parameters: uuid string Output: processor handle object

Exceptions that can be thrown by amdsmi\_get\_processor\_handle\_from\_uuid function:

- AmdSmiLibraryException

Example:

```
try:
 processor = amdsmi_get_processor_handle_from_uuid("fcff7460-0000-1000-80e9-
↪b388cfe84658")
 print("Processor's UUID: ", amdsmi_get_gpu_device_uuid(processor))
except AmdSmiException as e:
 print(e)
```

### 7.5.14 amdsmi\_get\_vf\_handle\_from\_uuid

Description: Returns the handle of a virtual function (VF) from the given UUID

Input parameters: uuid string Output: vf object

Exceptions that can be thrown by amdsmi\_get\_vf\_handle\_from\_uuid function:

- AmdSmiLibraryException

Example:

```
try:
 vf = amdsmi_get_vf_handle_from_uuid("87007460-0000-1000-8059-3ae746ab9206")
 print("VF's UUID: ", amdsmi_get_vf_uuid(vf))
except AmdSmiException as e:
 print(e)
```

### 7.5.15 amdsmi\_get\_gpu\_device\_uuid

Description: Returns the UUID of the device

Input parameters:

- GPU device 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:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
 print("Device UUID: ", amdsmi_get_gpu_device_uuid(processor))
except AmdSmiException as e:
 print(e)
```

### 7.5.16 amdsmi\_get\_vf\_uuid

Description: Returns the UUID of the device

Input parameters:

- VF handle for which to query

Output: UUID string unique to the device

Exceptions that can be thrown by amdsmi\_get\_vf\_uuid function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```
try:
 vf_id = amdsmi_get_vf_handle_from_vf_index(processor_handle, 0)
 print("VF UUID: ", amdsmi_get_vf_uuid(vf_id))
except AmdSmiException as e:
 print(e)
```

### 7.5.17 amdsmi\_get\_gpu\_driver\_info

Description: Returns the version string of the driver

Input parameters:

- processor\_handle GPU device for which to query

Output:

- driver\_name Driver name string that is handling the GPU device
- driver\_version Driver version string that is handling the GPU device
- driver\_date Driver date string that is handling the GPU device

Exceptions that can be thrown by amdsmi\_get\_gpu\_driver\_info function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```

try:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
 driver_info = amdsmi_get_gpu_driver_info(processor)
 print("Driver name: ", driver_info.driver_name)
 print("Driver version: ", driver_info.driver_version)
 print("Driver date: ", driver_info.driver_date)
except AmdSmiException as e:
 print(e)

```

### 7.5.18 amdsmi\_get\_gpu\_driver\_model

Description: Returns driver model information

Input parameters:

- `processor_handle` GPU device for which to query

Output:

- current driver model from `AmdSmiDriverModelType` enum

`AmdSmiDriverModelType` enum:

Field	Description
WDDM	Windows Display Driver Model
WDM	Windows Driver Model
MCDM	Compute Driver Model

Exceptions that can be thrown by `amdsmi_get_gpu_driver_model` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```

try:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23:00.0")
 driver_model = amdsmi_get_gpu_driver_model(processor)
 print("Driver model: ", driver_model)
except AmdSmiException as e:
 print(e)

```

### 7.5.19 amdsmi\_get\_vf\_handle\_from\_vf\_index

Description: Returns VF id of the VF referenced by its index (in partitioning info)

Input parameters:

- `processor` handle PF or child VF of a GPU device for which to query
- `VF's index` Index of VF (0-31) in GPU's partitioning info

Output:

- VF id

Exceptions that can be thrown by `amdsmi_get_vf_handle_from_vf_index` function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```
try:
 vf_id = amdsmi_get_vf_handle_from_vf_index(processor_handle, 0)
 print(amdsmi_get_vf_info(vf_id))
except AmdSmiException as e:
 print(e)
```

### 7.5.20 amdsmi\_get\_gpu\_total\_ecc\_count

Description: Returns the number of ECC errors on the GPU device

Input parameters:

- processor\_handle GPU device which to query

Output: Dictionary with fields

Field	Description
correctable_count	Count of ECC correctable errors
uncorrectable_count	Count of ECC uncorrectable errors
deferred_count	Count of ECC deferred errors

Exceptions that can be thrown by amdsmi\_get\_gpu\_total\_ecc\_count function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 ecc_errors = amdsmi_get_gpu_total_ecc_count(processor)
 print(ecc_errors['correctable_count'])
 print(ecc_errors['uncorrectable_count'])
 print(ecc_errors['deferred_count'])
except AmdSmiException as e:
 print(e)
```

### 7.5.21 amdsmi\_get\_gpu\_ecc\_count

Description: Returns the number of ECC errors on the GPU device for the given block

Input parameters:

- processor\_handle GPU device which to query

- `block` The block for which error counts should be retrieved

`block` is `AmdSmiGpuBlock` enum:

Field	Description
UMC	UMC block
SDMA	SDMA block
GFX	GFX block
MMHUB	MMHUB block
ATHUB	ATHUB block
PCIE_BIF	PCIE_BIF block
HDP	HDP block
XGMI_WAFL	XGMI_WAFL block
DF	DF block
SMN	SMN block
SEM	SEM block
MP0	MP0 block
MP1	MP1 block
FUSE	FUSE block
MCA	MCA block
VCN	VCN block
JPEG	JPEG block
IH	IH block
MPIO	MPIO block

Output: Dictionary with fields

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

Exceptions that can be thrown by `amdsmi_get_gpu_ecc_count` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 ecc_errors = amdsmi_get_gpu_ecc_count(processor, AmdSmiGpuBlock.UMC)
 print(ecc_errors['correctable_count'])
 print(ecc_errors['uncorrectable_count'])
 print(ecc_errors['deferred_count'])
except AmdSmiException as e:
 print(e)
```

## 7.5.22 amdsmi\_get\_gpu\_ecc\_enabled

Description: Returns ECC capabilities (disable/enable) for each GPU block.

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary of each GPU block and its value (False if the block is not enabled, True if the block is enabled)

Each GPU block in the dictionary is from `AmdSmiGpuBlock` enum:

Field	Description
UMC	UMC block
SDMA	SDMA block
GFX	GFX block
MMHUB	MMHUB block
ATHUB	ATHUB block
PCIE_BIF	PCIE_BIF block
HDP	HDP block
XGMI_WAFL	XGMI_WAFL block
DF	DF block
SMN	SMN block
SEM	SEM block
MP0	MP0 block
MP1	MP1 block
FUSE	FUSE block
MCA	MCA block
VCN	VCN block
JPEG	JPEG block
IH	IH block
MPIO	MPIO block

Exceptions that can be thrown by `amdsmi_get_gpu_ecc_enabled` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 ecc_status = amdsmi_get_gpu_ecc_enabled(processor, AmdSmiGpuBlock.UMC)
 print(ecc_status)
except AmdSmiException as e:
 print(e)
```

### 7.5.23 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(AmdSmiRetCode.SUCCESS)
 print(status_str)
except AmdSmiException as e:
 print(e)
```

### 7.5.24 amdsmi\_get\_gpu\_ras\_feature\_info

Description: Returns RAS feature info

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Description
<code>ras_eeeprom_version</code>	RAS EEPROM version
<code>ecc_correction_schema</code>	ecc correction schema mask used with <code>AmdSmiEccCorrectionSchemaSupport</code> enum

Exceptions that can be thrown by `amdsmi_get_gpu_ras_feature_info` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 ras_feature = amdsmi_get_gpu_ras_feature_info(processor)
 print(ras_feature['ras_eeeprom_version'])
 print(ras_feature['ecc_correction_schema'])
except AmdSmiException as e:
 print(e)
```

### 7.5.25 amdsmi\_get\_bad\_page\_threshold

Description: Returns bad page threshold

Input parameters:

- `processor_handle` GPU device which to query

Output: Bad page threshold value

Exceptions that can be thrown by `amdsmi_get_bad_page_threshold` function:

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

Example:

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

### 7.5.26 amdsmi\_get\_gpu\_bad\_page\_info

Description: Returns bad page info.

Input parameters:

- `processor handle` object PF of a GPU device to query

Output: list of dictionaries with fields for each bad page

Field	Description
<code>retired_page</code>	64K/4K Driver managed location that is blocked from further use
<code>ts</code>	Marks the last time when the RAS event was observed
<code>mem_channel</code>	this value identifies the memory channel the issue has been reported on
<code>mcumc_id</code>	this value identifies the memory controller the issue has been reported on

Exceptions that can be thrown by `amdsmi_get_gpu_bad_page_info` function:

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

Example:

```
try:
 processor = amdsmi_get_processor_handle_from_bdf("0000:23.00.0")
 bad_page_info = amdsmi_get_gpu_bad_page_info(processor)
```

(continues on next page)

(continued from previous page)

```

if len(bad_page_info) == 0:
 print("no bad pages")
else:
 for table_record in bad_page_info:
 print(hex(table_record["retired_page"]))
 print(datetime.fromtimestamp(table_record['ts']).strftime('%Y/%m/%d:%H/%M/%S
→'))
 print(table_record['mem_channel'])
 print(table_record['mcumc_id'])

except AmdSmiException as e:
 print(e)

```

### 7.5.27 amdsmi\_get\_gpu\_asic\_info

Description: Returns asic information for the given GPU

Input parameters:

- processor\_handle GPU device which to query

Output: Dictionary with fields

Field	Content
market_name	market name
vendor_id	vendor id
vendor_name	vendor name
subvendor_id	subsystem vendor id
device_id	unique id of a GPU
rev_id	revision id
asic_serial	asic serial
oam_id	xgmi physical id
num_of_compute_units	num of compute units ( <i>Not supported yet, currently hardcoded to 0</i> )
target_graphics_version	target graphics version ( <i>Not supported yet, currently hardcoded to 0</i> )
subsystem_id	subsystem device id

Exceptions that can be thrown by amdsmi\_get\_gpu\_asic\_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 asic_info = amdsmi_get_gpu_asic_info(processor)
 print(asic_info['market_name'])

```

(continues on next page)

(continued from previous page)

```

print(asic_info['vendor_id'])
print(asic_info['vendor_name'])
print(asic_info['subvendor_id'])
print(asic_info['device_id'])
print(asic_info['subsystem_id'])
print(asic_info['rev_id'])
print(asic_info['asic_serial'])
print(asic_info['oam_id'])
print(asic_info['num_of_compute_units'])
print(asic_info['target_graphics_version'])
except AmdSmiException as e:
 print(e)

```

### 7.5.28 amdsmi\_get\_pcie\_info

Description: Returns static and metric information about PCIe link for the given GPU

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Content
<code>pcie</code>	Subfield Description <code>max_pcie_width</code> Maximum number of PCIe lanes <code>max_pcie_speed</code> Maximum PCIe speed <code>pcie_interface_version</code> PCIe interface version <code>slot_type</code> Card form factor from <code>AmdSmiCardFormFactor</code> enum <code>max_pcie_interface_version</code> Maximum PCIe link generation
<code>pcie</code>	Subfield Description <code>pcie_speed</code> Current PCIe speed in MT/s <code>pcie_width</code> Current number of PCIe lanes <code>pcie_bandwidth</code> Current PCIe bandwidth in Mb/s <code>pcie_replay_count</code> Total number of the replays issued on the PCIe link <code>pcie_l0_to_recovery_count</code> Total number of times the PCIe link transitioned from L0 to the recovery state <code>pcie_replay_roll_over_count</code> Total number of replay rollovers issued on the PCIe link <code>pcie_nak_sent_count</code> Total number of NAKs issued on the PCIe link by the device <code>pcie_nak_received_count</code> Total number of NAKs issued on the PCIe link by the receiver <code>pcie_lc_perf_other_end_recovery_count</code> PCIe other end recovery counter

Exceptions that can be thrown by `amdsmi_get_pcie_info` function:

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

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 pcie_info = amdsmi_get_pcie_info(processor)
 print(pcie_info['pcie_static']['max_pcie_width'])
 print(pcie_info['pcie_static']['max_pcie_speed'])

```

(continues on next page)

(continued from previous page)

```

print(pcie_info['pcie_static']['slot_type'])
print(pcie_info['pcie_static']['max_pcie_interface_version'])
print(pcie_info['pcie_metric']['pcie_speed'])
print(pcie_info['pcie_metric']['pcie_width'])
print(pcie_info['pcie_metric']['pcie_bandwidth'])
print(pcie_info['pcie_metric']['pcie_interface_version'])
print(pcie_info['pcie_metric']['pcie_replay_count'])
print(pcie_info['pcie_metric']['pcie_l0_to_recovery_count'])
print(pcie_info['pcie_metric']['pcie_replay_roll_over_count'])
print(pcie_info['pcie_metric']['pcie_nak_sent_count'])
print(pcie_info['pcie_metric']['pcie_nak_received_count'])
print(pcie_info['pcie_metric']['pcie_lc_perf_other_end_recovery_count'])

```

```

except AmdSmiException as e:
 print(e)

```

### 7.5.29 amdsmi\_get\_power\_cap\_info

Description: Returns dictionary of power capabilities as currently configured on the given GPU

Input parameters:

- `processor_handle` GPU device which to query
- `sensor_ind` sensor index. Normally, this will be 0. If a processor has more than one sensor, it could be greater than 0. Parameter `sensor_ind` is unused on `@platform{host}`. It is an optional parameter and is set to 0 by default.

Output: Dictionary with fields

Field	Description
<code>power_cap</code>	power capability
<code>default_power_cap</code>	default power capability
<code>dpm_cap</code>	dynamic power management capability
<code>min_power_cap</code>	minimum power capability
<code>max_power_cap</code>	maximum power capability

Exceptions that can be thrown by `amdsmi_get_power_cap_info` function:

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

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 power_info = amdsmi_get_power_cap_info(processor)
 print(power_info['power_cap'])

```

(continues on next page)

(continued from previous page)

```

 print(power_info['default_power_cap'])
 print(power_info['dpm_cap'])
 print(power_info['min_power_cap'])
 print(power_info['max_power_cap'])
except AmdSmiException as e:
 print(e)

```

### 7.5.30 amdsmi\_get\_fb\_layout

Description: Returns framebuffer related information for the given GPU

Input parameters:

- `processor` handle PF of a GPU device for which to query

Output: Dictionary with field

Field	Description
<code>total_fb_size</code>	total framebuffer size in MB
<code>pf_fb_reserved</code>	framebuffer reserved space in MB
<code>pf_fb_offset</code>	framebuffer offset in MB
<code>fb_alignment</code>	framebuffer alignment in MB
<code>max_vf_fb_usable</code>	maximum usable framebuffer size in MB
<code>min_vf_fb_usable</code>	minimum usable framebuffer size in MB

Exceptions that can be thrown by `amdsmi_get_fb_layout` function:

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

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 fb_info = amdsmi_get_fb_layout(processor)
 print(fb_info['total_fb_size'])
 print(fb_info['pf_fb_reserved'])
 print(fb_info['pf_fb_offset'])
 print(fb_info['fb_alignment'])
 print(fb_info['max_vf_fb_usable'])
 print(fb_info['min_vf_fb_usable'])
except AmdSmiException as e:
 print(e)

```

### 7.5.31 amdsmi\_get\_gpu\_activity

Description: Returns the engine usage for the given GPU.

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Description
<code>gfx_activity</code>	graphics engine usage/activity percentage (0 - 100)
<code>umc_activity</code>	memory/UMC engine usage/activity percentage (0 - 100)
<code>mm_activity</code>	average multimedia engine usages/activities in percentage (0 - 100)

Exceptions that can be thrown by `amdsmi_get_gpu_activity` function:

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

Example:

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

### 7.5.32 amdsmi\_get\_power\_info

Description: Returns the current power, power limit, and voltage for the given GPU

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Description
Note: <code>socket_power</code> can rarely spike above the socket power limit in some cases	
<code>socket_power</code>	socket power
<code>gfx_voltage</code>	gfx voltage
<code>soc_voltage</code>	socket voltage
<code>mem_voltage</code>	memory voltage

Exceptions that can be thrown by `amdsmi_get_power_info` function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 power_info = amdsmi_get_power_info(processor)
 print(power_info['socket_power'])
 print(power_info['gfx_voltage'])
 print(power_info['soc_voltage'])
 print(power_info['mem_voltage'])
except AmdSmiException as e:
 print(e)
```

### 7.5.33 amdsmi\_set\_power\_cap

Description: Sets GPU power cap.

Input parameters:

- processor handle processor handle
- sensor\_ind sensor index. Normally, this will be 0. If a processor has more than one sensor, it could be greater than 0. Parameter sensor\_ind is unused on @platform{host}.
- cap value representing power cap to set. The value must be between the minimum (min\_power\_cap) and maximum (max\_power\_cap) power cap values, which can be obtained from ::amdsmi\_power\_cap\_info\_t.

Output:

- None

Exceptions that can be thrown by amdsmi\_set\_power\_cap function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 sensor_ind = 0
 for processor in processors:
 power_info = amdsmi_get_power_cap_info(processor)
 power_limit = random.randint(power_info['min_power_cap'], power_info['max_
↪power_cap'])
 amdsmi_set_power_cap(processor, sensor_ind, power_limit)
```

(continues on next page)

(continued from previous page)

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

### 7.5.34 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:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 is_power_management_enabled = amdsmi_is_gpu_power_management_
 enabled(processor)
 print(is_power_management_enabled)
except AmdSmiException as e:
 print(e)
```

### 7.5.35 amdsmi\_get\_temp\_metric

Description: Returns the current temperature or limit temperature for the given processor

Input parameters:

- `processor_handle` GPU device which to query
- `thermal_domain` one of `AmdSmiTemperatureType` enum values:

Field	Description
EDGE	edge thermal domain
HOTSPOT	hotspot/junction thermal domain
VRAM	memory/vram thermal domain
PLX	plx thermal domain ( <i>Not supported yet</i> )
HBM_0	HBM 0 thermal domain ( <i>Not supported yet</i> )
HBM_1	HBM 1 thermal domain ( <i>Not supported yet</i> )
HBM_2	HBM 2 thermal domain ( <i>Not supported yet</i> )
HBM_3	HBM 3 thermal domain ( <i>Not supported yet</i> )

- `thermal_metric` one of `AmdSmiTemperatureMetric` enum values:

Field	Description
CURRENT	current thermal metric
MAX	max thermal metric ( <i>Not supported yet</i> )
MIN	min thermal metric ( <i>Not supported yet</i> )
MAX_HYST	max hyst thermal metric ( <i>Not supported yet</i> )
MIN_HYST	min hyst thermal metric ( <i>Not supported yet</i> )
CRITICAL	limit thermal metric
CRITICAL_HYST	critical hyst metric ( <i>Not supported yet</i> )
EMERGENCY	emergency thermal metric ( <i>Not supported yet</i> )
EMERGENCY_HYST	emergency hyst thermal metric ( <i>Not supported yet</i> )
CRIT_MIN	critical min thermal metric ( <i>Not supported yet</i> )
CRIT_MIN_HYST	critical min hyst thermal metric ( <i>Not supported yet</i> )
OFFSET	offset thermal metric ( <i>Not supported yet</i> )
LOWEST	lowest thermal metric ( <i>Not supported yet</i> )
HIGHEST	highest thermal metric ( <i>Not supported yet</i> )
SHUTDOWN	shutdown thermal metric

Output: Temperature value

Exceptions that can be thrown by `amdsmi_get_temp_metric` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for device in processors:
 print("===== EDGE THERMAL DOMAIN =====")
 thermal_measure = amdsmi_get_temp_metric(device, AmdSmiTemperatureType.EDGE, ↵
↵AmdSmiTemperatureMetric.CURRENT)
 print("Current temperature:")
 print(thermal_measure)
 thermal_measure = amdsmi_get_temp_metric(device, AmdSmiTemperatureType.EDGE, ↵
↵AmdSmiTemperatureMetric.CRITICAL)
 print("Limit temperature:")
 print(thermal_measure)
 print("===== HOTSPOT THERMAL DOMAIN =====")
 thermal_measure = amdsmi_get_temp_metric(device, AmdSmiTemperatureType.
↵HOTSPOT, AmdSmiTemperatureMetric.CURRENT)
 print("Current temperature:")
 print(thermal_measure)
 thermal_measure = amdsmi_get_temp_metric(device, AmdSmiTemperatureType.
↵HOTSPOT, AmdSmiTemperatureMetric.CRITICAL)
 print("Limit temperature:")
 print(thermal_measure)
```

(continues on next page)

(continued from previous page)

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

### 7.5.36 amdsmi\_get\_gpu\_cache\_info

Description: Returns the cache info for the given processor

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Description
<code>cache_properties</code>	array of <code>AmdSmiCacheProperty</code>
<code>cache_size</code>	cache size
<code>cache_level</code>	cache level
<code>max_num_cu_shared</code>	number of Compute Units
<code>num_cache_instance</code>	number of instance of this cache type

`AmdSmiCacheProperty` enum values

Field	Description
<code>ENABLED</code>	Cache enabled
<code>DATA_CACHE</code>	Data cache
<code>INST_CACHE</code>	Inst cache
<code>CPU_CACHE</code>	CPU cache
<code>SIMD_CACHE</code>	SIMD cache

Exceptions that can be thrown by `amdsmi_get_gpu_cache_info` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for device in processors:
 cache_info = amdsmi_get_gpu_cache_info(device)
 for cache in cache_info["cache"]:
 print(cache["cache_properties"])
 print(cache["cache_size"])
 print(cache["cache_level"])
 print(cache["max_num_cu_shared"])
 print(cache["num_cache_instance"])
except AmdSmiException as e:
 print(e)
```

### 7.5.37 amdsmi\_get\_clock\_info

Description: Returns the clock measurements for the given GPU

Input parameters:

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

Field	Description
SYS	system clock domain
GFX	gfx clock domain
DF	Data Fabric clock (for ASICs running on a separate clock) domain ( <i>Not supported yet</i> )
DCEF	Display Controller Engine clock domain ( <i>Not supported yet</i> )
SOC	SOC clock domain ( <i>Not supported yet</i> )
MEM	memory clock domain
PCIE	PCIe clock domain ( <i>Not supported yet</i> )
VCLK0	first multimedia engine (VCLK0) clock domain
VCLK1	second multimedia engine (VCLK1) clock domain
DCLK0	DCLK0 clock domain
DCLK1	DCLK1 clock domain

Output: Dictionary with fields

Field	Description
<code>clk</code>	current clock value for the given domain
<code>min_clk</code>	minimum clock value for the given domain
<code>max_clk</code>	maximum clock value for the given domain
<code>clk_locked</code>	clock locked 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:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 print("===== GFX CLOCK DOMAIN =====")
 clock_measure = amdsmi_get_clock_info(processor, 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'])
```

(continues on next page)

(continued from previous page)

```

print("===== MEM CLOCK DOMAIN =====")
clock_measure = amdsmi_get_clock_info(processor, AmdSmiClkType.MEM)
print(clock_measure['clk'])
print(clock_measure['min_clk'])
print(clock_measure['max_clk'])
print(clock_measure['clk_deep_sleep'])
print("===== SYS CLOCK DOMAIN =====")
clock_measure = amdsmi_get_clock_info(processor, AmdSmiClkType.SYS)
print(clock_measure['clk'])
print(clock_measure['min_clk'])
print(clock_measure['max_clk'])
print(clock_measure['clk_deep_sleep'])
print("===== VCLK0 CLOCK DOMAIN =====")
clock_measure = amdsmi_get_clock_info(processor, AmdSmiClkType.VCLK0)
print(clock_measure['clk'])
print(clock_measure['min_clk'])
print(clock_measure['max_clk'])
print(clock_measure['clk_deep_sleep'])
print("===== VCLK1 CLOCK DOMAIN =====")
clock_measure = amdsmi_get_clock_info(processor, AmdSmiClkType.VCLK1)
print(clock_measure['clk'])
print(clock_measure['min_clk'])
print(clock_measure['max_clk'])
print(clock_measure['clk_deep_sleep'])
print("===== DCLK0 CLOCK DOMAIN =====")
clock_measure = amdsmi_get_clock_info(processor, AmdSmiClkType.DCLK0)
print(clock_measure['clk'])
print(clock_measure['min_clk'])
print(clock_measure['max_clk'])
print(clock_measure['clk_deep_sleep'])
print("===== DCLK1 CLOCK DOMAIN =====")
clock_measure = amdsmi_get_clock_info(processor, AmdSmiClkType.DCLK1)
print(clock_measure['clk'])
print(clock_measure['min_clk'])
print(clock_measure['max_clk'])
print(clock_measure['clk_deep_sleep'])
except AmdSmiException as e:
 print(e)

```

### 7.5.38 amdsmi\_get\_gpu\_vram\_info

Description: Returns the static information for the VRAM info

Input parameters:

- processor\_handle GPU device which to query

Output: Dictionary with fields

Field	Description
vram_type	VRAM type from <code>AmdSmiVramType</code> enum
vram_vendor	VRAM vendor from <code>AmdSmiVramVendor</code> enum
vram_size	VRAM size in MB
vram_bit_width	VRAM bit width

`AmdSmiVramType` enum:

Field	Description
UNKNOWN	UNKNOWN VRAM type
HBM	HBM VRAM type
HBM2	HBM2 VRAM type
HBM2E	HBM2E VRAM type
HBM3	HBM3 VRAM type
HBM3E	HBM3E VRAM type
DDR2	DDR2 VRAM type
DDR3	DDR3 VRAM type
DDR4	DDR4 VRAM type
GDDR1	GDDR1 VRAM type
GDDR2	GDDR2 VRAM type
GDDR3	GDDR3 VRAM type
GDDR4	GDDR4 VRAM type
GDDR5	GDDR5 VRAM type
GDDR6	GDDR6 VRAM type
GDDR7	GDDR7 VRAM type

`AmdSmiVramVendor` enum:

Field	Description
SAMSUNG	SAMSUNG VRAM vendor
INFINEON	INFINEON VRAM vendor
ELPIDA	ELPIDA VRAM vendor
ETRON	ETRON VRAM vendor
NANYA	NANYA VRAM vendor
HYNIX	HYNIX VRAM vendor
MOSEL	MOSEL VRAM vendor
WINBOND	WINBOND VRAM vendor
ESMT	ESMT VRAM vendor
MICRON	MICRON VRAM vendor
UNKNOWN	UNKNOWN VRAM vendor

Exceptions that can be thrown by `amdsmi_get_gpu_vram_info` function:

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

Example:

```

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

```

### 7.5.39 amdsmi\_get\_gpu\_vbios\_info

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

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Description
<code>name</code>	vbios name
<code>build_date</code>	vbios build date
<code>part_number</code>	vbios part number
<code>version</code>	vbios version string
<code>boot_firmware</code>	boot firmware info

Exceptions that can be thrown by `amdsmi_get_gpu_vbios_info` function:

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

Example:

```

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

```

### 7.5.40 amdsmi\_get\_fw\_info

Description: Returns the firmware information for the given GPU.

Input parameters:

- `processor_handle` GPU device which to query

Output: Dictionary with fields

Field	Description
<code>fw_info_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:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 firmware_list = amdsmi_get_fw_info(processor)
 for firmware_block in firmware_list:
 print(firmware_block['fw_id'])
 print(firmware_block['fw_version'])
except AmdSmiException as e:
 print(e)
```

### 7.5.41 amdsmi\_get\_gpu\_board\_info

Description: Returns board related information for the given GPU

Input parameters:

- GPU device handle object

Output: Dictionary with fields

Field	Description
<code>model_number</code>	board model number
<code>product_serial</code>	board product serial number
<code>fru_id</code>	fru (field-replaceable unit) id
<code>product_name</code>	board product name
<code>manufacturer_name</code>	board manufacturer name

Exceptions that can be thrown by `amdsmi_get_gpu_board_info` function:

- `AmdSmiLibraryException`

- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 board_info = amdsmi_get_gpu_board_info(processor)
 print(board_info['model_number'])
 print(board_info['product_serial'])
 print(board_info['fru_id'])
 print(board_info['manufacturer_name'])
 print(board_info['product_name'])
except AmdSmiException as e:
 print(e)
```

### 7.5.42 amdsmi\_get\_num\_vf

Description: Returns number of enabled VFs and number of supported VFs for the given GPU

Input parameters:

- processor handle PF of a GPU device for which to query

Output: Dictionary with fields

Field	Description
num_vf_enabled	number of enabled VFs
num_vf_supported	number of supported VFs

Exceptions that can be thrown by amdsmi\_get\_num\_vf function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 num_vf = amdsmi_get_num_vf(processor)
 print(num_vf['num_vf_enabled'])
 print(num_vf['num_vf_supported'])
except AmdSmiException as e:
 print(e)
```

### 7.5.43 amdsmi\_get\_vf\_partition\_info

Description: Returns array of the current framebuffer partitioning structures on the given GPU

Input parameters:

- `processor_handle` object PF of a GPU device for which to query

Output: Array of dictionary with fields

Field	Description
<code>vf_id</code>	VF handle
<code>fb</code>	Subfield Description <code>fb_size</code> framebuffer size <code>fb_offset</code> framebuffer offset

Exceptions that can be thrown by `amdsmi_get_vf_partition_info` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 partitions = amdsmi_get_vf_partition_info(processor)
 print(partitions[0]['fb']['fb_size'])
 # partitions[0]['fb']['fb_size'] is frame buffer size of the first VF on the
 ↪ given GPU
 # we can access any VF from the array via its index in partitions list
except AmdSmiException as e:
 print(e)
```

### 7.5.44 amdsmi\_set\_num\_vf

Description: Set number of enabled VFs for the given GPU

Input parameters:

- `processor_handle` GPU device which to query
- number of enabled VFs to be set

Output: None

Exceptions that can be thrown by `amdsmi_set_num_vf` function:

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

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 amdsmi_set_num_vf(processor,2)
except AmdSmiException as e:
 print(e)

```

### 7.5.45 amdsmi\_clear\_vf\_fb

Description: Clears framebuffer of the given VF on the given GPU.

If trying to clear the framebuffer of an active function, the call will fail

Input parameters:

- VF device handle

Output: None

Exceptions that can be thrown by amdsmi\_clear\_vf\_fb function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for device in devices:
 partitions = amdsmi_get_vf_partition_info(device)
 amdsmi_clear_vf_fb(partitions[0]['vf_id'])
 # partitions[0]['vf_id'] is handle of the first VF on the given GPU
except AmdSmiException as e:
 print(e)

```

### 7.5.46 amdsmi\_get\_vf\_data

Description: Returns the scheduler information and guard structure for the given VF.

Input parameters:

- VF handle

Output: Dictionary with fields

Field	Description
flr_count	function level reset counter
boot_up_t	boot up time in microseconds
shutdown_	shutdown time in microseconds
reset_tim	reset time in microseconds
state	vf state
last_boot	last boot start time
last_boot	last boot end time
last_shut	last shutdown start time
last_shut	last shutdown end time
last_rese	last reset start time
last_rese	last reset end time
current_a	current session active time, reset after guest reload
current_r	current session running time, reset after guest reload
total_act	total active time, reset after host reload
total_run	total running time, reset after host reload
enabled	show if guard info is enabled for VF
guard	Subfield Descriptionstate vf guard state amount amount of monitor events after enabled interval (dictiona: interval in seconds (sliding window) in which events are counted threshold maximum number of of events that will be processed in the given sliding window interval. Additional events during the interval elements) will be ignored.active current number of events in the interval

AmdSmiGuardType enum values are keys in guard dictionary

Field	Description
FLR	function level reset status
EXCLUSIVE_MOD	exclusive access mode status
EXCLUSIVE_TIMEOUT	exclusive access time out status
ALL_INT	generic interrupt status

State is AmdSmiGuardState enum object with values

Field	Description
NORMAL	the event number is within the threshold
FULL	the event number hits the threshold
OVERFLOW	the event number is bigger than the threshold

State is AmdSmiVfState enum object with values

Field	Description
UNAVAILABLE	vf state unavailable
AVAILABLE	vf state available
ACTIVE	vf state active
SUSPENDED	vf state suspended
FULLACCESS	vf state fullaccess
DEFAULT_AVAILABLE	same as available, indicates this is a default VF

Exceptions that can be thrown by amdsmi\_get\_vf\_data function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 partitions = amdsmi_get_vf_partition_info(processor)
 vf_data = amdsmi_get_vf_data(partitions[0]['vf_id'])
 sched_info = vf_data['sched']
 guard_info = vf_data['guard']
 print(sched_info['boot_up_time'])
 print(sched_info['flr_count'])
 print(sched_info['state'].name)
 print(sched_info['last_boot_start'])
 print(sched_info['last_boot_end'])
 print(sched_info['last_shutdown_start'])
 print(sched_info['last_shutdown_end'])
 print(sched_info['shutdown_time'])
 print(sched_info['last_reset_start'])
 print(sched_info['last_reset_end'])
 print(sched_info['reset_time'])
 print(sched_info['current_active_time'])
 print(sched_info['current_running_time'])
 print(sched_info['total_active_time'])
 print(sched_info['total_running_time'])

 print(guard_info['enabled'])
 for guard_type in guard_info['guard']:
 print("type: {}".format(guard_type))
 print("state: {}".format(guard_info['guard'][guard_type]['state']))
 print("amount: {}".format(guard_info['guard'][guard_type]['amount']))
 print("interval: {}".format(guard_info['guard'][guard_type]['interval']))
 print("threshold: {}".format(guard_info['guard'][guard_type]['threshold
↪']))
 print("active: {}".format(guard_info['guard'][guard_type]['active']))
 print("=====")
except AmdSmiException as e:
 print(e)

```

### 7.5.47 amdsmi\_get\_vf\_info

Description: Returns the configuration structure for a given VF

Input parameters:

- VF handle

Output: Dictionary with fields

Field	Description
fb	Subfield Description fb_offsetframebuffer offsetfb_sizeframebuffer size
gfx_timeslice	gfx timeslice in us

Exceptions that can be thrown by `amdsmi_get_vf_info` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 partitions = amdsmi_get_vf_partition_info(processor)
 # partitions[0]['vf_id'] is handle of the first VF on the given GPU
 config = amdsmi_get_vf_info(partitions[0]['vf_id'])
 print("fb_offset: {}".format(config['fb']['fb_offset']))
 print("fb_size: {}".format(config['fb']['fb_size']))

 print("gfx_timeslice : {}".format(config['gfx_timeslice']))
except AmdSmiException as e:
 print(e)
```

### 7.5.48 amdsmi\_get\_guest\_data

Description: Gets guest OS information of the queried VF

Input parameters:

- `processor` handle VF of a GPU device

Output: Dictionary with fields

Field	Description
driver_version	driver version
fb_usage	fb usage in MB

Exceptions that can be thrown by `amdsmi_get_guest_data` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
```

(continues on next page)

(continued from previous page)

```

 print("No GPUs on machine")
else:
 for processor in processors:
 num_vf_enabled = amdsmi_get_num_vf(processor)['num_vf_enabled']
 partitions = amdsmi_get_vf_partition_info(processor)
 for i in range(0, num_vf_enabled):
 guest_data = amdsmi_get_guest_data(partitions[i]['vf_id'])
 print(guest_data)
except AmdSmiException as e:
 print(e)

```

### 7.5.49 amdsmi\_get\_fw\_error\_records

Description: Gets firmware error records

Input parameters:

- processor handle PF of a GPU device

Output: Dictionary with field err\_records, which is list of elements

Field	Description
timestamp	system time in seconds
vf_idx	vf index
fw_id	firmware id
status	firmware load status

Exceptions that can be thrown by amdsmi\_get\_fw\_error\_records function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 err_records = amdsmi_get_fw_error_records(processor)
 print(err_records)
except AmdSmiException as e:
 print(e)

```

### 7.5.50 amdsmi\_get\_dfc\_fw\_table

Description: Gets dfc firmware table

Input parameters:

- processor handle PF of a GPU device

Output: Dictionary with field header, and data which is a list of elements

Each header is a dictionary with following fields:

Field	Description
dfc_fw_version	dfc firmware version
dfc_fw_total_entries	number of entries in the dfc table
dfc_gart_wr_guest_min	gart wr guest min
dfc_gart_wr_guest_max	gart wr guest max

Each data entry is a dictionary with following fields:

Field	Description
dfc_fw_type	dfc firmware type
verification_enabled	verification enabled
customer_ordinal	customer ordinal
white_list	white list
black_list	black list

Each white list entry is a dictionary with following fields:

Field	Description
latest	latest
oldest	oldest

Exceptions that can be thrown by `amdsmi_get_dfc_fw_table` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

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

### 7.5.51 amdsmi\_get\_vf\_fw\_info

Description: Returns GPU firmware related information.

Input parameters:

- `processor` handle VF of a GPU device for which to query

Output: Dictionary with field `fw_list`, which is list of elements

If microcode of certain type is not loaded, version will be 0.

Field	Description
<code>fw_info_list</code>	Subfield Description <code>fw_id</code> <code>AmdSmiFwBlock</code> enum <code>fw_version</code> fw version which is integer

Exceptions that can be thrown by `amdsmi_get_vf_fw_info` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 partitions = amdsmi_get_vf_partition_info(processor)
 fw_info = amdsmi_get_vf_fw_info(partitions[0]['vf_id'])
 fw_num = len(fw_info['fw_list'])
 for j in range(0, fw_num):
 fw = fw_info['fw_list'][j]
 print(fw['fw_name'].name)
 print(fw['fw_version'])
except AmdSmiException as e:
 print(e)
```

### 7.5.52 amdsmi\_get\_partition\_profile\_info

Description: Gets partition profile info

Input parameters:

- `processor` handle PF of a GPU device

Output: Dictionary with fields, `current_profile` and `profiles`

Field	Description
<code>current_profile_index</code>	current profile index
<code>profiles</code>	list of all profiles

Where, `profiles` is a list containing

Field	Description
<code>vf_count</code>	number of vfs
<code>profile_caps</code> (dictionary)	Subfield Description <code>total</code> <code>total available</code> <code>available optimal</code> <code>optimal min_value</code> <code>minimum</code> <code>max_value</code> <code>maximum</code>

Keys for `profile_caps` dictionary are in `AmdSmiProfileCapabilityType` enum

Field	Description
MEMORY	memory
ENCODE	encode engine
DECODE	decode engine
COMPUTE	compute engine

Exceptions that can be thrown by `amdsmi_get_partition_profile_info` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 profile_info = amdsmi_get_partition_profile_info(processor)
 print(profile_info)

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

### 7.5.53 amdsmi\_get\_link\_metrics

Description: Gets link metric information

Input parameters:

- `processor handle` PF of a GPU device

Output: links list of dictionaries with fields for each link

Field	Description
<code>bdf</code>	BDF of the given processor
<code>bit_rate</code>	current link speed in Gb/s
<code>max_bandwidth</code>	max bandwidth of the link
<code>link_type</code>	type of the link from <code>AmdSmiLinkType</code> enum
<code>read</code>	total data received for each link in KB
<code>write</code>	total data transferred for each link in KB

`AmdSmiLinkType` enum:

Field	Description
INTERNAL	Unknown
XGMI	XGMI link type
PCIE	PCIe link type
NOT_APPLICABLE	Link not applicable
UNKNOWN	Unknown

Exceptions that can be thrown by `amdsmi_get_link_metrics` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 link_metrics = amdsmi_get_link_metrics(processor)
 print(link_metrics)

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

### 7.5.54 amdsmi\_get\_link\_topology

Description: Gets link topology information between two connected processors

Input parameters:

- `source_processor` handle PF of a source GPU device
- `destination_processor` handle PF of a destination GPU device

Output: Dictionary with fields

Field	Description
<code>weight</code>	link weight between two GPUs
<code>link_status</code>	HW status of the link
<code>link_type</code>	type of the link from <code>AmdSmiLinkType</code> enum
<code>num_hops</code>	number of hops between two GPUs
<code>fb_sharing</code>	framebuffer sharing between two GPUs

`AmdSmiLinkType` enum:

Field	Description
INTERNAL	Unknown
XGMI	XGMI link type
PCIE	PCIe link type
NOT_APPLICABLE	Link not applicable
UNKNOWN	Unknown

Exceptions that can be thrown by `amdsmi_get_link_topology` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for src_processor in processors:
 for dst_processor in processors:
 link_topology = amdsmi_get_link_topology(src_processor, dst_processor)
 print(link_topology)

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

### 7.5.55 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

`AmdSmiLinkType` enum:

Field	Description
INTERNAL	Unknown
XGMI	XGMI link type
PCIE	PCIe link type
NOT_APPLICABLE	Link not applicable
UNKNOWN	Unknown

Output: Dictionary holding the following fields.

- `processor_list` list of all nearest processor handles found

Exceptions that can be thrown by `amdsmi_get_link_topology_nearest` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 nearest_gpu = amdsmi_get_link_topology_nearest(processor, AmdSmiLinkType.
←PCIE)
 if (len(nearest_gpus['processor_list'])) == 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)

```

### 7.5.56 amdsmi\_get\_xgmi\_fb\_sharing\_caps

Description: Gets XGMI capabilities

Input parameters:

- processor handle PF of a GPU device

Output: Dictionary with fields

Field	Description
mode_custom_cap	flag that indicates if custom mode is supported ( <i>Not supported yet</i> )
mode_1_cap	flag that indicates if mode_1 is supported
mode_2_cap	flag that indicates if mode_2 is supported
mode_4_cap	flag that indicates if mode_4 is supported
mode_8_cap	flag that indicates if mode_8 is supported

Exceptions that can be thrown by amdsmi\_get\_xgmi\_fb\_sharing\_caps function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 caps = amdsmi_get_xgmi_fb_sharing_caps(processor)
 print(caps)

```

(continues on next page)

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

### 7.5.57 amdsmi\_get\_xgmi\_fb\_sharing\_mode\_info

Description: Gets XGMI framebuffer sharing information between two GPUs

Input parameters:

- source processor handle PF of a source GPU device
- destination processor handle PF of a destination GPU device
- mode framebuffer sharing mode from `AmdSmiXgmiFbSharingMode` enum

`AmdSmiXgmiFbSharingMode` enum:

Field	Description
CUSTOM	custom framebuffer sharing mode ( <i>Not supported yet</i> )
MODE_1	framebuffer sharing mode_1
MODE_2	framebuffer sharing mode_2
MODE_4	framebuffer sharing mode_4
MODE_8	framebuffer sharing mode_8

Output: Value indicating whether framebuffer sharing is enabled between two GPUs

Exceptions that can be thrown by `amdsmi_get_xgmi_fb_sharing_mode_info` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for src_processor in processors:
 for dst_processor in processors:
 fb_sharing = amdsmi_get_xgmi_fb_sharing_mode_info(src_processor, dst_
↪processor, AmdSmiXgmiFbSharingMode.MODE_4)
 print(fb_sharing)

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

### 7.5.58 amdsmi\_set\_xgmi\_fb\_sharing\_mode

Description: Sets framebuffer sharing mode

Note: This API will only work if there's no guest VM running.

Input parameters:

- processor handle PF of a GPU device

- mode framebuffer sharing mode from `AmdSmiXgmiFbSharingMode` enum

`AmdSmiXgmiFbSharingMode` enum:

Field	Description
CUSTOM	custom framebuffer sharing mode ( <i>Not supported yet</i> )
MODE_1	framebuffer sharing mode_1
MODE_2	framebuffer sharing mode_2
MODE_4	framebuffer sharing mode_4
MODE_8	framebuffer sharing mode_8

Output:

- None

Exceptions that can be thrown by `amdsmi_set_xgmi_fb_sharing_mode` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 amdsmi_set_xgmi_fb_sharing_mode(processor, AmdSmiXgmiFbSharingMode.MODE_4)
except AmdSmiException as e:
 print(e)
```

### 7.5.59 `amdsmi_set_xgmi_fb_sharing_mode_v2`

Description: Sets framebuffer sharing mode

Note: This API will only work if there's no guest VM running. This api can be used for custom and auto setting of xgmi frame buffer sharing. In case of custom mode: - All processors in the list must be on the same NUMA node. Otherwise, api will return error. - If any processor from the list already belongs to an existing group, the existing group will be released automatically. In case of auto mode(MODE\_X): - The input parameter `processor_list[0]` should be valid. Only the first element of `processor_list` is taken into account and it can be any `gpu0,gpu1,...`

Input parameters:

- `processor_list` list of PFs of a GPU devices
- mode framebuffer sharing mode from `AmdSmiXgmiFbSharingMode` enum

`AmdSmiXgmiFbSharingMode` enum:



Field	Description
val	value of the metric
unit	unit of the metric from AmdSmiMetricUnit enum
name	name of the metric from AmdSmiMetricName enum
category	category of the metric from AmdSmiMetricCategory enum
flags	list of types of the metric from AmdSmiMetricType enum
vf_mask	mask of all active VFs + PF that this metric applies to
res_group	resource group from AmdSmiResGroup enum
res_subgroup	resource group from AmdSmiResSubGroup enum
res_instance	resource instance number

AmdSmiMetricUnit enum:

Field	Description
COUNTER	counter
UINT	unsigned integer
BOOL	boolean
MHZ	megahertz
PERCENT	percentage
MILLIVOLT	millivolt
CELSIUS	celsius
WATT	watt
JOULE	joule
GBPS	gigabyte per second
MBITPS	megabit per second
PCIE_GEN	PCIe generation
PCIE_LANES	PCIe lanes
15_625_MILLIJOULE	millijoule
UNKNOWN	unknown unit

AmdSmiMetricName enum:

Field	Description
ACC_COUNTER	accumulated counter
FW_TIMESTAMP	firmware timestamp
CLK_GFX	gfx clock
CLK_SOC	socket clock
CLK_MEM	memory clock
CLK_VCLK	vclk clock
CLK_DCLK	dclk clock
USAGE_GFX	gfx usage
USAGE_MEM	memory usage
USAGE_MM	mm usage
USAGE_VCN	vcn usage
USAGE_JPEG	jpeg usage
VOLT_GFX	gfx voltage
VOLT_SOC	socket voltage
VOLT_MEM	memory voltage
TEMP_HOTSPOT_CURR	current hotspot temperature

continues on next page

Table 7.1 – continued from previous page

Field	Description
TEMP_HOTSPOT_LIMIT	hotspot temperature limit
TEMP_MEM_CURR	current memory temperature
TEMP_MEM_LIMIT	memory temperature limit
TEMP_VR_CURR	current vr temperature
TEMP_SHUTDOWN	shutdown temperature
POWER_CURR	current power
POWER_LIMIT	power limit
ENERGY_SOCKET	socket energy
ENERGY_CCD	ccd energy
ENERGY_XCD	xcd energy
ENERGY_AID	aid energy
ENERGY_MEM	memory energy
THROTTLE_SOCKET_ACTIVE	active socket throttle
THROTTLE_VR_ACTIVE	active vr throttle
THROTTLE_MEM_ACTIVE	active memory throttle
PCIE_BANDWIDTH	pcie bandwidth
PCIE_L0_TO_RECOVERY_COUNT	pcie l0 recovery count
PCIE_REPLAY_COUNT	pcie replay count
PCIE_REPLAY_ROLLOVER_COUNT	pcie replay rollover count
PCIE_NAK_SENT_COUNT	pcie nak sent count
PCIE_NAK_RECEIVED_COUNT	pcie nak received count
CLK_GFX_MAX_LIMIT	maximum gfx clock limit
CLK_SOC_MAX_LIMIT	maximum socket clock limit
CLK_MEM_MAX_LIMIT	maximum memory clock limit
CLK_VCLK_MAX_LIMIT	maximum vclk clock limit
CLK_DCLK_MAX_LIMIT	maximum dclk clock limit
CLK_GFX_MIN_LIMIT	minimum gfx clock limit
CLK_SOC_MIN_LIMIT	minimum socket clock limit
CLK_MEM_MIN_LIMIT	minimum memory clock limit
CLK_VCLK_MIN_LIMIT	minimum vclk clock limit
CLK_DCLK_MIN_LIMIT	minimum dclk clock limit
CLK_GFX_LOCKED	gfx clock locked
CLK_GFX_DS_DISABLED	gfx deep sleep
CLK_MEM_DS_DISABLED	memory deep sleep
CLK_SOC_DS_DISABLED	socket deep sleep
CLK_VCLK_DS_DISABLED	vclk deep sleep
CLK_DCLK_DS_DISABLED	dclk deep sleep
PCIE_LINK_SPEED	pcie link speed
PCIE_LINK_WIDTH	pcie link width
DRAM_BANDWIDTH	dram bandwidth
MAX_DRAM_BANDWIDTH	maximum dram bandwidth
GFX_CLK_BELOW_HOST_LIMIT_PPT	gfx clock below host limit ppt
GFX_CLK_BELOW_HOST_LIMIT_THM	gfx clock below host limit thermal
GFX_CLK_BELOW_HOST_LIMIT_TOTAL	gfx clock below host limit total
GFX_CLK_LOW_UTILIZATION	gfx clock low utilization
INPUT_TELEMETRY_VOLTAGE	input telemetry voltage
PLDM_VERSION	pldm version
TEMP_XCD	xcd temperature
TEMP_AID	aid temperature
TEMP_HBM	hbm temperature

continues on next page

Table 7.1 – continued from previous page

Field	Description
SYS_METRIC_ACC_COUNTER	system metric accumulated counter
SYSTEM_TEMP_UBB_FPGA	system temperature ubb fpga
SYSTEM_TEMP_UBB_FRONT	system temperature ubb front
SYSTEM_TEMP_UBB_BACK	system temperature ubb back
SYSTEM_TEMP_UBB_OAM7	system temperature ubb oam7
SYSTEM_TEMP_UBB_IBC	system temperature ubb ibc
SYSTEM_TEMP_UBB_UFPGA	system temperature ubb ufpga
SYSTEM_TEMP_UBB_OAM1	system temperature ubb oam1
SYSTEM_TEMP_OAM_0_1_HSC	system temperature oam 0 1 hsc
SYSTEM_TEMP_OAM_2_3_HSC	system temperature oam 2 3 hsc
SYSTEM_TEMP_OAM_4_5_HSC	system temperature oam 4 5 hsc
SYSTEM_TEMP_OAM_6_7_HSC	system temperature oam 6 7 hsc
SYSTEM_TEMP_UBB_FPGA_0V72_VR	system temperature ubb fpga 0v72 vr
SYSTEM_TEMP_UBB_FPGA_3V3_VR	system temperature ubb fpga 3v3 vr
SYSTEM_TEMP_RETIMER_0_1_2_3_1V2_VR	system temperature retimer 0 1 2 3 1v2 vr
SYSTEM_TEMP_RETIMER_4_5_6_7_1V2_VR	system temperature retimer 4 5 6 7 1v2 vr
SYSTEM_TEMP_RETIMER_0_1_0V9_VR	system temperature retimer 0 1 0v9 vr
SYSTEM_TEMP_RETIMER_4_5_0V9_VR	system temperature retimer 4 5 0v9 vr
SYSTEM_TEMP_RETIMER_2_3_0V9_VR	system temperature retimer 2 3 0v9 vr
SYSTEM_TEMP_RETIMER_6_7_0V9_VR	system temperature retimer 6 7 0v9 vr
SYSTEM_TEMP_OAM_0_1_2_3_3V3_VR	system temperature oam 0 1 2 3 3v3 vr
SYSTEM_TEMP_OAM_4_5_6_7_3V3_VR	system temperature oam 4 5 6 7 3v3 vr
SYSTEM_TEMP_IBC_HSC	system temperature ibc hsc
SYSTEM_TEMP_IBC	system temperature ibc
NODE_TEMP_RETIMER	node temperature retimer
NODE_TEMP_IBC_TEMP	node temperature ibc temp
NODE_TEMP_IBC_2_TEMP	node temperature ibc 2 temp
NODE_TEMP_VDD18_VR_TEMP	node temperature vdd18 vr temp
NODE_TEMP_04_HBM_B_VR_TEMP	node temperature 04 hbm b vr temp
NODE_TEMP_04_HBM_D_VR_TEMP	node temperature 04 hbm d vr temp
VR_TEMP_VDDCR_VDD0	vr temperature vddcr vdd0
VR_TEMP_VDDCR_VDD1	vr temperature vddcr vdd1
VR_TEMP_VDDCR_VDD2	vr temperature vddcr vdd2
VR_TEMP_VDDCR_VDD3	vr temperature vddcr vdd3
VR_TEMP_VDDCR_SOC_A	vr temperature vddcr soc a
VR_TEMP_VDDCR_SOC_C	vr temperature vddcr soc c
VR_TEMP_VDDCR_SOCIO_A	vr temperature vddcr socio a
VR_TEMP_VDDCR_SOCIO_C	vr temperature vddcr socio c
VR_TEMP_VDD_085_HBM	vr temperature vdd 085 hbm
VR_TEMP_VDDCR_11_HBM_B	vr temperature vddcr 11 hbm b
VR_TEMP_VDDCR_11_HBM_D	vr temperature vddcr 11 hbm d
VR_TEMP_VDD_USR	vr temperature vdd usr
VR_TEMP_VDDIO_11_E32	vr temperature vddio 11 e32
UNKNOWN	unknown name

AmdSmiMetricCategory enum:

Field	Description
ACC_COUNTER	counter
FREQUENCY	frequency
ACTIVITY	activity
TEMPERATURE	temperature
POWER	power
ENERGY	energy
THROTTLE	throttle
PCIE	pcie
STATIC	static
SYS_ACC_COUNTER	system accumulated counter
SYS_BASEBOARD_TEMP	system baseboard temperature
SYS_GPUBOARD_TEMP	system gpu board temperature
UNKNOWN	unknown category

AmdSmiMetricType enum:

Field	Description
COUNTER	counter
CHIPLET	chiplet
INST	instantaneous data
ACC	accumulated data

AmdSmiMetricResGroup enum:

Field	Description
NA	resource group is not applicable
GPU	gpu resource group
XCP	xcp resource group
AID	aid resource group
MID	mid resource group
SYSTEM	system resource group
UNKNOWN	unknown resource group

AmdSmiMetricResSubgroup enum:

Field	Description
NA	resource subgroup is not applicable
XCC	xcc resource subgroup
ENGINE	engine resource subgroup
HBM	hbm resource subgroup
BASEBOARD	baseboard resource subgroup
GPUBOARD	gpuboard resource subgroup
UNKNOWN	unknown resource subgroup

Exceptions that can be thrown by `amdsmi_get_gpu_metrics` function:

- `AmdSmiLibraryException`

- AmdSmiParameterException

Example:

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

### 7.5.61 amdsmi\_get\_soc\_pstate

Description: Gets the soc pstate policy for the processor

Input parameters:

- processor handle PF of a GPU device

Output: Dictionary with fields

Field	Description
cur	current policy index
policies	List of policies

Each policies list entry is a dictionary with following fields:

Field	Description
policy_id	policy id
policy_description	policy description

Exceptions that can be thrown by amdsmi\_get\_soc\_pstate function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 dpm_policy = amdsmi_get_soc_pstate(processor)
 print(dpm_policy)
```

(continues on next page)

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

### 7.5.62 amdsmi\_set\_soc\_pstate

Description: Sets the soc pstate policy for the processor

Input parameters:

- `processor` handle PF of a GPU device
- `policy_id` policy id represents one of the values we get from the policies list from `amdsmi_get_soc_pstate`.

Output:

- None

Exceptions that can be thrown by `amdsmi_set_soc_pstate` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

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

### 7.5.63 amdsmi\_get\_xgmi\_plpd

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

Input parameters:

- `processor` handle PF of a GPU device

Output: Dictionary with fields

Field	Description
<code>cur</code>	current policy index
<code>policies</code>	List of policies

Each policies list entry is a dictionary with following fields:

Field	Description
<code>policy_id</code>	policy id
<code>policy_description</code>	policy description

Exceptions that can be thrown by `amdsmi_get_xgmi_plpd` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

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

### 7.5.64 `amdsmi_set_xgmi_plpd`

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

Input parameters:

- `processor` handle PF of a GPU device
- `policy_id` policy id represents one of the values we get from the policies list from `amdsmi_get_soc_pstate`.

Output:

- None

Exceptions that can be thrown by `amdsmi_set_xgmi_plpd` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

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

### 7.5.65 `AmdSmiEventReader` class

Description: Providing methods for event monitoring

Methods:

**constructor**

Description: Allocates a new event reader notifier to monitor different types of issues with the GPU

Input parameters:

- `processor handle list` list of GPU device handle objects(PFs od Vfs) for which to create event reader
- `event category list` list of the different event categories that the event reader will monitor in GPU

Event category is `AmdSmiEventCategory` enum object with values

Category	Description
NOT_USED	not used category
DRIVER	driver events(allocation, failures of APIs, debug errors)
RESET	events/notifications regarding RESET executed by the GPU
SCHED	scheduling events(world switch fail ...)
VBIOS	VBIOS events(security failures, vbios corruption...)
ECC	ecc events
PP	pp events(slave not present, dpm fail ...)
IOV	events regarding the configuration of VF resources
VF	vf events(no vbios, gpu reset fail...)
FW	events related with FW loading or FW operations
GPU	gpu fatal conditions
GUARD	guard events
GPUMON	gpumon events(fb issues ...)
MMSCH	mmsch events
XGMI	xgmi events
ALL	monitor all categories

- `severity` of events that can be monitored

Severity is `AmdSmiEventSeverity` enum object with values

Severity	Description
HIGH	critical error
MED	significant error
LOW	trivial error
WARN	warning
INFO	info
ALL	monitor all severity levels

`severity` parameter is optional. If nothing is set, events with `LOW` severity will be monitored by default.

Output:

- created object of `AmdSmiEventReader` class

Exceptions that can be thrown by `AmdSmiEventReader` constructor function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 event_reader = AmdSmiEventReader(processors, {AmdSmiEventCategory.RESET})
except SmiException as e:
 print(e)

```

## read

Description: Reads and return one event from event reader

Input parameters:

- `timestamp` number of microseconds to wait for an event to occur. If event does not happen monitoring is finished

Output: Dictionary with fields

Field	Description
<code>fcn_id</code>	VF handle
<code>dev_id</code>	GPU device id
<code>timestamp</code>	UTC time (in microseconds) when the error happened
<code>data</code>	data value associated with the specific event
<code>category</code>	event category
<code>subcode</code>	event subcategory
<code>level</code>	event severity
<code>date</code>	UTC date and time when the error happend
<code>message</code>	message describing the event
<code>processor_handle</code>	processor handle object where the event happened

event category is `AmdSmiEventCategory` enum object with values

Category	Description
<code>NOT_USED</code>	not used category
<code>DRIVER</code>	driver events(allocation, failures of APIs, debug errors)
<code>RESET</code>	events/notifications regarding RESET executed by the GPU
<code>SCHED</code>	scheduling events(world switch fail ...)
<code>VBIOS</code>	VBIOS events(security failures, vbios corruption...)
<code>ECC</code>	ecc events
<code>PP</code>	pp events(slave not present, dpm fail ...)
<code>IOV</code>	events regarding the configuration of VF resources
<code>VF</code>	vf events(no vbios, gpu reset fail...)
<code>FW</code>	events related with FW loading or FW operations
<code>GPU</code>	gpu fatal conditions
<code>GUARD</code>	guard events
<code>GPUMON</code>	gpumon events(fb issues ...)
<code>MMSCH</code>	mmsch events
<code>XGMI</code>	xgmi events
<code>ALL</code>	monitor all categories

every `AmdSmiEventCategory` has its corresponding enum subcategory, subcategories are:

Sub-Field category	
AmdS	GPU_DEVICE_LOSTGPU_NOT_SUPPORTEDGPU_RMAGPU_NOT_INITIALIZEDGPU_MMSCH_ABNORMAL_STAT
AmdS	DRIVER_SPIN_LOCK_BUSYDRIVER_ALLOC_SYSTEM_MEM_FAILDRIVER_CREATE_GFX_WORKQUEUE_FAILD
AmdS	FW_CMD_ALLOC_BUF_FAILFW_CMD_BUF_PREP_FAILFW_RING_INIT_FAILFW_FW_APPLY_SECURITY_POLIC
AmdS	RESET_GPURESET_GPU_FAILEDRESET_FLRRESET_FLR_FAILED
AmdS	IOV_NO_GPU_IOV_CAPIOV_ASIC_NO_SRIOV_SUPPORTIOV_ENABLE_SRIOV_FAILIOV_CMD_TIMEOUTIOV_CM
AmdS	SMI_EVENT_ECC_UCESMI_EVENT_ECC_CEECC_IN_PF_FBECC_IN_CRI_REGECC_IN_VF_CRIECC_REACH_THD
AmdS	PP_SET_DPM_POLICY_FAILPP_ACTIVATE_DPM_POLICY_FAILPP_I2C_SLAVE_NOT_PRESENTPP_THROTTLER_E
AmdS	SCHED_WORLD_SWITCH_FAILSCHED_DISABLE_AUTO_HW_SWITCH_FAILSCHED_ENABLE_AUTO_HW_SWITC
AmdS	VF_ATOMBIOS_INIT_FAILVF_NO_VBIOSVF_GPU_POST_ERRORVF_ATOMBIOS_GET_CLOCK_FAILVF_FENCE_IN
AmdS	VBIOS_INVALIDVBIOS_IMAGE_MISSINGVBIOS_CHECKSUM_ERRVBIOS_POST_FAILVBIOS_READ_FAILVBIOS_
AmdS	GUARD_RESET_FAILGUARD_EVENT_OVERFLOW
AmdS	GPUMON_INVALID_OPTIONGPUMON_INVALID_VF_INDEXGPUMON_INVALID_FB_SIZEGPUMON_NO_SUITAB
AmdS	MMSCH_IGNORED_JOBMMMSCH_UNSUPPORTED_VCN_FW
AmdS	XGMI_TOPOLOGY_UPDATE_FAILEDXGMI_TOPOLOGY_HW_INIT_UPDATEXGMI_TOPOLOGY_UPDATE_DONEX

Severity is `AmdSmiEventSeverity` enum object with values

Severity	Description
HIGH	critical error
MED	significant error
LOW	trivial error
WARN	warning
INFO	info
ALL	monitor all severity levels

Exceptions that can be thrown by read function:

- `AmdSmiParameterException`
- `AmdSmiTimeoutException`
- `AmdSmiLibraryException`

### 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:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
```

(continues on next page)

(continued from previous page)

```

else:
 event_reader = AmdSmiEventReader(processors, {AmdSmiEventCategory.RESET}, ↵
↵AmdSmiEventSeverity.ALL)
 while True:
 event = event_reader.read(10*1000*1000)
 gpu_bdf = amdsmi_get_gpu_device_bdf(event['dev_id'])
 vf_bdf = amdsmi_get_gpu_device_bdf(event['fcn_id'])
 print("===== Event =====")
 print(" Time {}".format(event['timestamp']))
 print(" Category {}".format(event['category'].name))
 print(" Subcategory {}".format(event['subcode'].name))
 print(" Level {}".format(event['level'].name))
 print(" Data {}".format(event['data']))
 print(" VF BDF {}".format(vf_bdf))
 print(" GPU BDF {}".format(gpu_bdf))
 print(" Date {}".format(event['date']))
 print(" Message {}".format(event['message']))
 print("=====")
except AmdSmiTimeoutException:
 print("No more events")
finally:
 event_reader.stop()

```

Example with automatic cleanup using with statement:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 with AmdSmiEventReader(processors, {AmdSmiEventCategory.RESET}, ↵
↵AmdSmiEventSeverity.ALL) as event_reader:
 while True:
 event = event_reader.read(10*1000*1000)
 gpu_bdf = amdsmi_get_gpu_device_bdf(event['dev_id'])
 vf_bdf = amdsmi_get_gpu_device_bdf(event['fcn_id'])
 print("===== Event =====")
 print(" Time {}".format(event['timestamp']))
 print(" Category {}".format(event['category'].name))
 print(" Subcategory {}".format(event['subcode'].name))
 print(" Level {}".format(event['level'].name))
 print(" Data {}".format(event['data']))
 print(" VF BDF {}".format(vf_bdf))
 print(" GPU BDF {}".format(gpu_bdf))
 print(" Date {}".format(event['date']))
 print(" Message {}".format(event['message']))
 print("=====")
except AmdSmiTimeoutException:
 print("No more events")

```

### 7.5.66 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:
 devices = amdsmi_get_processor_handles()
 if len(devices) == 0:
 print("No GPUs on machine")
 else:
 for device in devices:
 version = amdsmi_get_lib_version()
 print(version)
except AmdSmiException as e:
 print(e)
```

### 7.5.67 amdsmi\_get\_gpu\_virtualization\_mode

Description: Retrieve the current GPU virtualization mode.

Input parameters:

- `processor handle` The handle to the GPU device for which the virtualization mode is being queried.

Output: Enum object representing the current virtualization mode.

`AmdSmiVirtualizationMode` enum:

Field	Description
UNKNOWN	unknown virtualization mode
NONE	none virtualization mode
HOST	host virtualization mode
GUEST	guest virtualization mode
PASSTHROUGH	passthrough virtualization mode

Exceptions that can be thrown by `amdsmi_get_gpu_virtualization_mode` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
```

(continues on next page)

(continued from previous page)

```

for processor in processors:
 virtualization_mode = amdsmi_get_gpu_virtualization_mode(processor)
 print("Virtualization mode: {}".format(virtualization_mode))

except AmdSmiException as e:
 print(e)

```

### 7.5.68 amdsmi\_get\_gpu\_accelerator\_partition\_profile\_config

Description: Returns gpu accelerator partition caps as currently configured in the system

Input parameters:

- `processor` handle PF of a GPU device

Output: Dictionary with fields

Field	Description
<code>resource_partition</code> (dictionary)	Subfield Description <code>profile_index</code> index of a profile <code>resource_type</code> type of a resource from <code>AmdSmiAcceleratorPartitionResource</code> enum <code>partition_resource</code> the resources a partition can use, which may be shared <code>num_partitions_share_resource</code> number of partitions that share resource of that type
<code>default_profile</code>	index of the default profile
<code>profiles</code> (dictionary)	Subfield Description <code>profile_type</code> profile type from <code>AmdSmiAcceleratorPartitionSetting</code> enum <code>num_partitions</code> number of partitions in the profile <code>memory_caps</code> memory capabilities of the profile <code>profile_index</code> index of the profile <code>resources</code> resources in the profile

`AmdSmiAcceleratorPartitionResource` enum:

Field	Description
XCC	xcc resource capabilities
ENCODER	encoder resource capabilities
DECODER	decoder resource capabilities
DMA	dma resource capabilities
JPEG	jpeg resource capabilities

`AmdSmiAcceleratorPartitionSetting` enum:

Field	Description
INVALID	invalid compute partition
SPX	compute partition with all xccs in group (8/1)
DPX	compute partition with four xccs in group (8/2)
TPX	compute partition with two xccs in group (6/3)
QPX	compute partition with two xccs in group (8/4)
CPX	compute partition with one xcc in group (8/8)

Exceptions that can be thrown by `amdsmi_get_gpu_accelerator_partition_profile_config` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 accelerator_partition_config = amdsmi_get_gpu_accelerator_partition_profile_
 ↪config(processor)
 print(accelerator_partition_config)
except AmdSmiException as e:
 print(e)
```

### 7.5.69 amdsmi\_get\_gpu\_accelerator\_partition\_profile\_config\_global

Description: Returns all GPU accelerator partition capabilities which can be configured on the system

Input parameters:

- processor\_handle PF of a GPU device

Output: Dictionary with fields

Field	Description
profi:	List of dictionaries, each describing a supported accelerator partition profile. Each dictionary contains: Subfield Descriptionprofile_typeProfile type from AmdSmiAcceleratorPartitionSetting enumnum_partitionsNumber of partitions in the profilememory_capsMemory capabilities of the profileprofile_indexIndex of the profilevf_modeList of supported VF counts for this profile (e.g., [1, 2, 4, 8])resourcesList of resource types (from AmdSmiAcceleratorPartitionResource enum) available in this profile
defau:	Index of the default profile used if no custom configuration is set

AmdSmiAcceleratorPartitionSetting enum:

Field	Description
INVALID	Invalid compute partition
SPX	Compute partition with all xccs in group (8/1)
DPX	Compute partition with four xccs in group (8/2)
TPX	Compute partition with two xccs in group (6/3)
QPX	Compute partition with two xccs in group (8/4)
CPX	Compute partition with one xcc in group (8/8)

AmdSmiAcceleratorPartitionResource enum:

Field	Description
XCC	xcc resource capabilities
ENCODER	encoder resource capabilities
DECODER	decoder resource capabilities
DMA	dma resource capabilities
JPEG	jpeg resource capabilities

Exceptions that can be thrown by `amdsmi_get_gpu_accelerator_partition_profile_config_global` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 config = amdsmi_get_gpu_accelerator_partition_profile_config_
↪global(processor)
 print(config)
except AmdSmiException as e:
 print(e)
```

### 7.5.70 amdsmi\_get\_gpu\_accelerator\_partition\_profile

Description: Returns current gpu accelerator partition cap

Input parameters:

- `processor` handle PF of a GPU device

Output: Dictionary with fields

Field	Description
<code>profile</code>	Subfield Description <code>profile_type</code> current profile type from <code>AmdSmiAcceleratorPartitionSetting</code>
<code>(diction</code>	<code>enum num_partitions</code> number of partitions in the profile <code>memory_caps</code> memory capabilities of the
<code>profile</code>	<code>profile_index</code> index of the profile <code>resources</code> resources in the profile
<code>partitio</code>	<code>array of ids</code> for current accelerator profile

`AmdSmiComputePartitionResource` enum:

Field	Description
XCC	xcc resource capabilities
ENCODER	encoder resource capabilities
DECODER	decoder resource capabilities
DMA	dma resource capabilities
JPEG	jpeg resource capabilities

Exceptions that can be thrown by `amdsmi_get_gpu_accelerator_partition_profile` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 accelerator_partition_profile = amdsmi_get_gpu_accelerator_partition_
↪profile(processor)
 print(accelerator_partition_profile)
except AmdSmiException as e:
 print(e)

```

### 7.5.71 amdsmi\_get\_gpu\_memory\_partition\_config

Description: Returns current gpu memory partition config and mode capabilities

Input parameters:

- processor handle PF of a GPU device

Output: Dictionary with fields

Field	Description
partition_caps	memory partition capabilities
mp_mode	memory partition mode from AmdSmiMemoryPartitionSetting enum
numa_range (dictionary)	Subfield Descriptionmemory_type memory type from AmdSmiVramType enum start start of numa range end end of numa range

'AmdSmiMemoryPartitionSetting' enum:

Field	Description
UNKNOWN	unknown memory partition
NPS1	memory partition with 1 number per socket
NPS2	memory partition with 2 numbers per socket
NPS4	memory partition with 4 numbers per socket
NPS8	memory partition with 8 numbers per socket

Exceptions that can be thrown by amdsmi\_get\_gpu\_memory\_partition\_config function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```

try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 memory_partition_config = amdsmi_get_gpu_memory_partition_config(processor)
 print(memory_partition_config)

```

(continues on next page)

(continued from previous page)

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

### 7.5.72 amdsmi\_set\_gpu\_accelerator\_partition\_profile

Description: Sets accelerator partition setting based on profile\_index from amdsmi\_get\_gpu\_accelerator\_partition\_profile\_config

Input parameters:

- processor handle PF of a GPU device
- profile\_index Represents index of a partition user wants to set

Output:

- None

Exceptions that can be thrown by amdsmi\_set\_gpu\_accelerator\_partition\_profile function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

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

### 7.5.73 amdsmi\_set\_gpu\_memory\_partition\_mode

Description: Sets memory partition mode

Input parameters:

- processor handle PF of a GPU device
- setting Enum from AmdSmiMemoryPartitionSetting representing memory partitioning mode to set

AmdSmiMemoryPartitionSetting enum:

Field	Description
UNKNOWN	unknown memory partition
NPS1	memory partition with 1 number per socket
NPS2	memory partition with 2 numbers per socket
NPS4	memory partition with 4 numbers per socket
NPS8	memory partition with 8 numbers per socket

Output:

- None

Exceptions that can be thrown by `amdsmi_set_gpu_memory_partition_mode` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 amdsmi_set_gpu_memory_partition_mode(processor, AmdSmiMemoryPartitionSetting.
↪NPS1)
except AmdSmiException as e:
 print(e)
```

### 7.5.74 amdsmi\_get\_gpu\_cper\_entries

Description: Dump CPER entries for a given GPU in a file using from CPER header file from RAS tool.

Input parameters:

- `processor_handle` device which to query
- `severity_mask` Represents different severity masks from 'AmdSmiCperErrorSeverity' enum on which filtering of cpers is based.
- `buffer_size` pointer to a variable that specifies the size of the `cper_data`
- `cursor` pointer to a variable that will contain the cursor for the next call

`AmdSmiCperErrorSeverity`

Field	Description
<code>NON_FATAL_UNCORRECTED</code>	filters non-fatal-uncorrected cpers
<code>FATAL</code>	filters fatal cpers
<code>NON_FATAL_CORRECTED</code>	filters non_fatal_corrected cpers
<code>NUM</code>	shows all cper types

Output: Dictionary with fields

Field	Description
error_severity	The severity of the CPER error ex: non_fatal_uncorrected, fatal, non_fatal_corrected.
notify_type	The notification type associated with the CPER entry.
timestamp	The time when the CPER entry was recorded, formatted as YYYY/MM/DD HH:MM:SS.
signature	A 4-byte signature identifying the entry, typically CPER.
revision	The revision number of the CPER record format.
signature_end	A marker value (typically 0xFFFFFFFF) confirming the integrity of the signature.
sec_cnt	The count of sections included in the CPER entry.
record_length	The total length in bytes of the CPER entry.
platform_id	A character array identifying the GPU or platform.
creator_id	A character array indicating the creator of the CPER entry.
record_id	A unique identifier for the CPER entry.
flags	Reserved flags related to the CPER entry.
persistence_info	Reserved information related to persistence.

Exceptions that can be thrown by `amdsmi_get_gpu_cper_entries` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
for device in devices:
 entries, new_cursor, cper_data = amdsmi_get_gpu_cper_entries(device, severity_
 ↪mask, buffer_size, initial_cursor)
 print("CPER entries for device", device)
 for key, entry in entries.items():
 print("Entry", key)
 print(" Error Severity:", entry.get("error_severity", "Unknown"))
 print(" Notify Type:", entry.get("notify_type", "Unknown"))
 print(" Timestamp:", entry.get("timestamp", ""))
 print()
 print("New Cursor Position:", new_cursor)
except AmdSmiException as e:
 print(e)
```

### 7.5.75 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	<code>AmdSmiLinkType</code>
cap	Subfield Description <code>is_iolink_coherent1 == True; 0 == False; Uint_max = Undefined</code> <code>is_iolink_atomics_32bitSupports 32bit atomics</code> <code>is_iolink_atomics_64bitSupports 64bit atomics</code> <code>is_iolink_dmaSupports DMA</code> <code>is_iolink_bi_directional</code> Is the IOLink Bidirectional

Exceptions that can be thrown by `amdsmi_get_p2p_status` function:

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

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 processor_handle_src = processors[0]
 processor_handle_dest = processors[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)
```

### 7.5.76 `amdsmi_get_afids_from_cper`

Description: Get AFID for cper error

Input parameters: buffer which contains one cper

- `cper_buffer`

Output:

- List of AFID

Exceptions that can be thrown by `amdsmi_get_afids_from_cper` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 entries, new_cursor, cper_data = amdsmi_get_gpu_cper_entries(processor, severity_
 ↪mask, buffer_size, initial_cursor)
 afid = amdsmi_get_afids_from_cper(cper_data[0]['bytes'])
 print(afid)
except AmdSmiException as e:
 print(e)
```

### 7.5.77 amdsmi\_reset\_gpu

Description: Triggers a chain that resets all GPUs.

Input parameters: GPU device handle

- processor\_handle

Output:

- None

Exceptions that can be thrown by amdsmi\_reset\_gpu function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

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

### 7.5.78 amdsmi\_get\_cpu\_affinity\_with\_scope

Description: Returns list of bitmask information for the given GPU.

Input parameters:

- processor\_handle device which to query
- scope enum value for numa or socket affinity

Output: bitmask of CPU cores that this processor affinities with

Exceptions that can be thrown by amdsmi\_get\_cpu\_affinity\_with\_scope 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:
 bitmask = amdsmi_get_cpu_affinity_with_scope(device, AmdSmiAffinityScope.
↪ NUMA_SCOPE)
 print(bitmask)
```

(continues on next page)

(continued from previous page)

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

### 7.5.79 amdsmi\_topo\_get\_numa\_node\_number

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_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:
 numa_node = amdsmi_topo_get_numa_node_number(device)
 print(numa_node)
except AmdSmiException as e:
 print(e)
```

### 7.5.80 amdsmi\_get\_nic\_driver\_info

Description: Retrieves information about the NIC driver

Input parameters:

- `processor_handle` NIC for which to query

Output: Dictionary with fields

Field	Description
<code>name</code>	driver name
<code>version</code>	driver version

Exceptions that can be thrown by `amdsmi_get_nic_driver_info` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```

try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 print(amdsmi_get_nic_driver_info(nic))
except AmdSmiException as e:
 print(e)

```

### 7.5.81 amdsmi\_get\_nic\_asic\_info

Description: Retrieves ASIC information for the NIC

Input parameters:

- processor\_handle NIC for which to query

Output: Dictionary with fields

Field	Description
vendor_id	vendor id
subvendor_id	subsystem vendor id
device_id	device id
subsystem_id	subsystem device id
revision	revision id
permanent_address	permanent mac address
product_name	product name
part_number	part number
serial_number	serial number
vendor_name	vendor name

Exceptions that can be thrown by amdsmi\_get\_nic\_asic\_info function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```

try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 print(amdsmi_get_nic_asic_info(nic))
except AmdSmiException as e:
 print(e)

```

### 7.5.82 amdsmi\_get\_nic\_bus\_info

Description: Retrieves BUS information for the NIC

Input parameters:

- `processor_handle` NIC for which to query

Output: Dictionary with fields

Field	Description
<code>bdf</code>	bus
<code>max_pcie_width</code>	maximum supported PCIe link width
<code>max_pcie_speed</code>	maximum supported PCIe link speed
<code>pcie_interface_version</code>	PCIe interface version ( <i>Not supported yet, currently hardcoded to 0</i> )
<code>slot_type</code>	physical slot type ( <i>Not supported yet, currently hardcoded to 0</i> )

Exceptions that can be thrown by `amdsmi_get_nic_bus_info` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 print(amdsmi_get_nic_bus_info(nic))
except AmdSmiException as e:
 print(e)
```

### 7.5.83 amdsmi\_get\_nic\_numa\_info

Description: Retrieves NUMA information for the NIC

Input parameters:

- `processor_handle` NIC for which to query

Output: Dictionary with fields

Field	Content
<code>node</code>	NUMA (Non-Uniform Memory Access) node identifier associated with the device.
<code>affinity</code>	CPU affinity mask for the device

Exceptions that can be thrown by `amdsmi_get_nic_numa_info` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```

try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 print(amdsmi_get_nic_numa_info(nic))
except AmdSmiException as e:
 print(e)

```

### 7.5.84 amdsmi\_get\_nic\_port\_info

Description: Retrieves PORT information for the NIC

Input parameters:

- `processor_handle` NIC for which to query

Output: Dictionary with fields

Field	Description
ports	List of port information dictionaries

Each port dictionary contains:

Field	Content
bdf	BDF of the port
port_num	Port number
type	Type of the port
flavour	Port flavour
netdev	Associated network device name
ifindex	Interface index of the port
mac_address	MAC address assigned to the port
carrier	Carrier state
mtu	Maximum Transmission Unit size
link_state	Current link state
link_speed	Link speed in Mbps
active_fec	Currently active Forward Error Correction mode
autoneg	Auto-negotiation status
pause_autoneg	Pause frame auto-negotiation status
pause_rx	Receive pause frame status
pause_tx	Transmit pause frame status

Active FEC Modes:

The `active_fec` field provides a bitmask representation of Active FEC (Active Forward Error Correction) modes. The bitmask values are derived from the `ethtool_fecparam` structure, specifically the `active_fec` field. Below are examples of the defined FEC modes:

- `ETHTOOL_FEC_NONE` (0x01)
- `ETHTOOL_FEC_AUTO` (0x02)
- `ETHTOOL_FEC_RS` (0x04)

- ETHTOOL\_FEC\_BASER (0x08)
- ETHTOOL\_FEC\_LLRS (0x10)
- ETHTOOL\_FEC\_OFF (0x20)

Note: These definitions are based on the latest available ethtool information. Users should verify if there are any updates or changes to these definitions in the relevant ethtool structure or field before implementing them in their code.

Exceptions that can be thrown by `amdsmi_get_nic_port_info` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 port_info = amdsmi_get_nic_port_info(nic)
 print(f"Number of ports: {len(port_info['ports'])}")
 for i, port in enumerate(port_info['ports']):
 print(f"Port {i}:")
 print(f" BDF: {port['bdf']}")
 print(f" Port Number: {port['port_num']}")
 print(f" Type: {port['type']}")
 print(f" Flavour: {port['flavour']}")
 print(f" Network Device: {port['netdev']}")
 print(f" Interface Index: {port['ifindex']}")
 print(f" MAC Address: {port['mac_address']}")
 print(f" Carrier: {port['carrier']}")
 print(f" MTU: {port['mtu']}")
 print(f" Link State: {port['link_state']}")
 print(f" Link Speed: {port['link_speed']}")
 print(f" Active FEC: {port['active_fec']}")
 print(f" Auto-negotiation: {port['autoneg']}")
 print(f" Pause Auto-negotiation: {port['pause_autoneg']}")
 print(f" Pause RX: {port['pause_rx']}")
 print(f" Pause TX: {port['pause_tx']}")
except AmdSmiException as e:
 print(e)
```

### 7.5.85 `amdsmi_get_nic_rdma_dev_info`

Description: Retrieves RDMA devices information for the NIC

Input parameters:

- `processor_handle` NIC for which to query

Output: Dictionary with fields

Field	Description
<code>rdma_dev_info</code>	List of RDMA device information dictionaries

Each RDMA device info dictionary contains:

Field	Description
rdma_dev	RDMA device name
node_guid	Node GUID
node_type	Node type
sys_image_guid	System image GUID
fw_ver	Firmware version
rdma_port_info	List of RDMA port information dictionaries

Each RDMA port info dictionary contains:

Field	Description
netdev	Associated network device name
state	Port state (DOWN, INIT, ARMED, ACTIVE, ACTIVE_DEFER)
rdma_port	RDMA port number
max_mtu	Maximum MTU size
active_mtu	Currently active MTU size

Exceptions that can be thrown by `amdsmi_get_nic_rdma_dev_info` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 rdma_info = amdsmi_get_nic_rdma_dev_info(nic)
 print(f"Number of RDMA devices: {len(rdma_info['rdma_dev_info'])}")
 for i, rdma_dev in enumerate(rdma_info['rdma_dev_info']):
 print(f"RDMA Device {i}:")
 print(f" Device: {rdma_dev['rdma_dev']}")
 print(f" Node GUID: {rdma_dev['node_guid']}")
 print(f" Node Type: {rdma_dev['node_type']}")
 print(f" System Image GUID: {rdma_dev['sys_image_guid']}")
 print(f" Firmware Version: {rdma_dev['fw_ver']}")
 print(f" Number of RDMA ports: {len(rdma_dev['rdma_port_info'])}")
 for j, port in enumerate(rdma_dev['rdma_port_info']):
 print(f" Port {j}:")
 print(f" Network Device: {port['netdev']}")
 print(f" State: {port['state']}")
 print(f" RDMA Port: {port['rdma_port']}")
 print(f" Max MTU: {port['max_mtu']}")
 print(f" Active MTU: {port['active_mtu']}")
except AmdSmiException as e:
 print(e)
```

### 7.5.86 amdsmi\_get\_nic\_port\_statistics

Description: Retrieve all available PORT statistics for the specified NIC port

Input parameters:

- `processor_handle` NIC for which to query
- `port_index` index of the NIC port to query

Output: Returns a list of dictionaries, each containing:

- `name`
- `value`

Exceptions that can be thrown by `amdsmi_get_nic_port_statistics` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```
try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 port_stats = amdsmi_get_nic_port_statistics(nic, 0)
 for stat in port_stats:
 print(f"{stat['name']}: {stat['value']}")
except AmdSmiException as e:
 print(e)
```

### 7.5.87 amdsmi\_get\_nic\_vendor\_statistics

Description: Retrieve vendor specific statistics for the NIC port

Input parameters:

- `processor_handle` NIC for which to query
- `port_index` index of the NIC port to query

Output: Returns a list of dictionaries, each containing:

- `name`
- `value`

This API provides access to vendor/driver specific statistics that may vary between different NIC vendors and driver/firmware versions. The statistic names are preserved as provided by the underlying driver implementation.

Note: The exact statistics available depend on the NIC vendor, hardware and driver version.

Exceptions that can be thrown by `amdsmi_get_nic_vendor_statistics` function:

- `AmdSmiParameterException`
- `AmdSmiLibraryException`

Example:

```

try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 for nic in nic_handles:
 vendor_stats = amdsmi_get_nic_vendor_statistics(nic, 0)
 for stat in vendor_stats:
 print(f"{stat['name']}: {stat['value']}")
except AmdSmiException as e:
 print(e)

```

### 7.5.88 amdsmi\_get\_nic\_rdma\_port\_statistics

Description: Retrieve RDMA port statistics for the NIC

Input parameters:

- processor\_handle NIC for which to query
- rdma\_port\_index index of the NIC RDMA port to query

Output: Returns a list of dictionaries, each containing:

- name
- value

Exceptions that can be thrown by amdsmi\_get\_nic\_rdma\_port\_statistics function:

- AmdSmiParameterException
- AmdSmiLibraryException

Example:

```

try:
 nic_handles = amdsmi_get_processor_handles_by_type(AmdSmiProcessorType.AMD_NIC)
 if len(nic_handles) == 0:
 print("No NICs on machine")
 else:
 rdma_port_index = 0
 for nic in nic_handles:
 rdma_stats = amdsmi_get_nic_rdma_port_statistics(nic, rdma_port_index)
 for stat in rdma_stats:
 print(f"{stat['name']}: {stat['value']}")
except AmdSmiException as e:
 print(e)

```

### 7.5.89 amdsmi\_get\_node\_handle

Description: Get the node handle associated with processor handle. Note: This function retrieves the node handle of a processor handle. The processor\_handle must be provided for the processor. Currently, only AMD GPUs are supported.

Input parameters:

- processor\_handle PF of a GPU device

Output: Node handle

Exceptions that can be thrown by `amdsmi_get_node_handle` function:

- `AmdSmiLibraryException`

Example:

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

### 7.5.90 `amdsmi_get_npm_info`

Description: Returns node power management (NPM) status and limit for specific node.

Note: This function queries the NPM controller for the given node and returns whether NPM is enabled,

- along with the current node-level power limit in Watts. The NPM status and limit are set out-of-band
- and reported via this API.

Input parameters:

- `processor handle` PF of a GPU device

Output:

- `npm info` NPM `AmdSmiNpmStatus` structure

Field	Description
<code>status</code>	NPM status
<code>limit</code>	currently set limit

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 node = amdsmi_get_node_handle(processor)
 npm_info = amdsmi_get_npm_info(node)
 print(npm_info)
except AmdSmiException as e:
 print(e)
```

### 7.5.91 amdsmi\_get\_gpu\_ras\_policy\_info

Description: Retrieve the Reliability, Availability, and Serviceability (RAS) policy information for a specified GPU device.

Input parameters:

processor\_handle: The handle for the GPU device for which policy information is to be retrieved.

Output: A dictionary containing RAS policy information including version, major and minor versions, and thresholds for DRAM regions.

Exceptions that can be thrown by amdsmi\_get\_gpu\_ras\_policy\_info function:

- AmdSmiLibraryException
- AmdSmiRetryException
- AmdSmiParameterException

Example:

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

### 7.5.92 amdsmi\_get\_gpu\_ptl\_state

Description: Gets the PTL (Peak Tops Limiter) enable/disable state for the processor

Input parameters:

- processor handle processor handle

Output:

- True if PTL is enabled, False if PTL is disabled

Exceptions that can be thrown by amdsmi\_get\_gpu\_ptl\_state function:

- AmdSmiLibraryException
- AmdSmiParameterException

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 enabled = amdsmi_get_gpu_ptl_state(processor)
 print(f"PTL Enabled: {enabled}")
```

(continues on next page)

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

### 7.5.93 amdsmi\_set\_gpu\_ptl\_state

Description: Sets the PTL (Peak Tops Limiter) enable/disable state for the processor

Input parameters:

- `processor handle` processor handle
- `enable` (bool) True to enable PTL with default formats, False to disable PTL

Exceptions that can be thrown by `amdsmi_set_gpu_ptl_state` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 # Enable PTL
 amdsmi_set_gpu_ptl_state(processor, True)

 # Disable PTL
 # amdsmi_set_gpu_ptl_state(processor, False)

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

### 7.5.94 amdsmi\_get\_gpu\_ptl\_formats

Description: Gets the current PTL (Peak Tops Limiter) preferred data formats for the processor

Input parameters:

- `processor handle` processor handle

Output:

- `tuple(AmdSmiPtlDataFormat, AmdSmiPtlDataFormat)`: an ordered pair of enums representing the two preferred formats Example usage: `fmt1, fmt2 = amdsmi_get_gpu_ptl_formats(handle)`

Note:

- If `fmt1 == fmt2 == AmdSmiPtlDataFormat.I8` (value 0), this indicates that PTL has not been enabled yet on that system and the PTL state is disabled

Available data formats (`AmdSmiPtlDataFormat`):

- I8 - Integer 8-bit format
- F16 - Float 16-bit format
- BF16 - Brain Float 16-bit format

- F32 - Float 32-bit format
- F64 - Float 64-bit format

Exceptions that can be thrown by `amdsmi_get_gpu_ptl_formats` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 format1, format2 = amdsmi_get_gpu_ptl_formats(processor)
 print(f"PTL Format 1: {format1.name}")
 print(f"PTL Format 2: {format2.name}")

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

### 7.5.95 amdsmi\_set\_gpu\_ptl\_formats

Description: Sets the PTL (Peak Tops Limiter) with specified preferred data format pair. PTL must be enabled first before calling this function using `amdsmi_set_gpu_ptl_state`.

Input parameters:

- `processor` handle processor handle
- `data_format1` (`AmdSmiPtlDataFormat`) First preferred data format
- `data_format2` (`AmdSmiPtlDataFormat`) Second preferred data format (must be different from `data_format1`)

The two specified formats will receive accurate performance monitoring and peak performance. F8 and XF32 formats always receive peak performance regardless of this setting.

Exceptions that can be thrown by `amdsmi_set_gpu_ptl_formats` function:

- `AmdSmiLibraryException`
- `AmdSmiParameterException`

Example:

```
try:
 from amdsmi import AmdSmiPtlDataFormat

 processors = amdsmi_get_processor_handles()
 if len(processors) == 0:
 print("No GPUs on machine")
 else:
 for processor in processors:
 # Set PTL formats
 amdsmi_set_gpu_ptl_formats(processor,
 AmdSmiPtlDataFormat.F16,
```

(continues on next page)

(continued from previous page)

```
AmdSmiPtlDataFormat.BF16)
```

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

## VERSIONING GUIDE FOR AMD SMI LIBRARY AND TOOL

### 8.1 Overview

This section explains the versioning conventions used in the AMD SMI (System Management Interface) library and AMD SMI tool. Understanding these versioning rules helps developers maintain backward compatibility and communicate changes effectively.

### 8.2 SMI Library Versioning

The SMI library uses a three-part version number format:

```
major.minor.release
```

Version information is stored in the VERSION file at the root of the smi-lib repository:

```
major=<int_major>
minor=<int_minor>
release=<int_release>
```

### 8.3 Version Number Components

#### 8.3.1 Major Version (major)

The major version is incremented when changes affect backward compatibility of the API:

- Modification or removal of existing structures that break backward compatibility
- Breaking changes to enum definitions (removing values or reordering)
- Modifications to API function signatures
- Deprecating or removing existing APIs
- Any other breaking change in the amdsmi.h header file

When the major version is incremented, both minor and release versions are reset to 0.

#### 8.3.2 Minor Version (minor)

The minor version is incremented when new functionality is added in a backward-compatible manner:

- Addition of new structures
- Addition of new enum values (without reordering existing ones)

- Addition of new API functions
- Non-breaking modifications to existing structures (e.g., adding new fields at the end)
- Any other non-breaking additions to the amdsmi.h header file

When the minor version is incremented, the release version is reset to 0.

### 8.3.3 Release Version (release)

The release version is incremented for implementation changes that don't affect the public API:

- Bug fixes
- Performance improvements
- Internal code refactoring
- Any changes outside the amdsmi.h header file (excluding cli/cpp folders)

## 8.4 SMI Tool Versioning

The SMI tool uses a similar three-part version number format:

```
major.minor.release
```

Version information is defined in the smi\_cli\_helpers.h file:

```
#define AMDSMI_TOOL_VERSION_MAJOR 27
#define AMDSMI_TOOL_VERSION_MINOR 6
#define AMDSMI_TOOL_VERSION_RELEASE 0
```

## 8.5 Version Number Components

### 8.5.1 Major Version

The major version is incremented when changes affect the tool's command interface:

- Deprecating or removing existing commands
- Modifying input/output format of commands

### 8.5.2 Minor Version

The minor version is incremented for command changes that maintain backward compatibility:

- Adding new commands
- Improvements to existing commands
- Adding new options to existing commands without changing the basic input/output format

### 8.5.3 Release Version

The release version is incremented for minor bug fixes and maintenance updates that don't add features or change command behavior.

## 8.6 Version Usage

### 8.6.1 In Code

Version numbers should be used throughout the codebase:

- For conditional compilation
- For feature detection
- For API compatibility checks

### 8.6.2 In Documentation

When referencing features or behavior, always include the version number where the feature was introduced or modified.

## 8.7 Version Compatibility

The SMI library and tool versions are not directly tied to each other, but certain tool versions may require minimum library versions to function correctly.

### 8.7.1 Updating Versions

1. Identify the type of change being made
2. Update the appropriate version number in the VERSION file or header file
3. Document the version change in the changelog

## 8.8 Examples

### 8.8.1 AMD SMI library:

#### Example 1: Bug Fix in Library Implementation

- Before: 31.1.2
- Change: Fix a bug in the temperature reporting function
- After: 31.1.3 (only release version incremented)

#### Example 2: New API Function

- Before: 31.1.3
- Change: Add new function `amdsmi_get_new_metric()`
- After: 31.2.0 (minor version incremented, release reset)

#### Example 3: Modify Existing API Signature

- Before: 31.2.0
- Change: Change parameter types in `amdsmi_existing_function()`
- After: 32.0.0 (major version incremented, minor and release reset)

## 8.8.2 AMD SMI tool:

### Example 1: Add New Tool Command

- Before: 27.6.0
- Change: Add new command `amdsmi new-command`
- After: 27.7.0 (minor version incremented)

**LICENSE**

Copyright (c) 2025 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.