GRID SOFTWARE MANAGEMENT SDK

DU-08141-001 _v4.0 (GRID) | August 2016

User Guide

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. INTRODUCTION TO THE NVIDIA GRID SOFTWARE MANAGEMENT SDK

The NVIDIA GRID Software Management SDK enables third party applications to monitor and control NVIDIA physical GPUs and virtual GPUs that are running on virtualization hosts. The NVIDIA GRID Management SDK supports control and monitoring of GPUs from both the hypervisor host system and from within guest VMs.

NVIDIA GRID vGPU enables multiple virtual machines (VMs) to have simultaneous, direct access to a single physical GPU, using the same NVIDIA graphics drivers that are deployed on non-virtualized operating systems. For an introduction to NVIDIA GRID vGPU, see *GRID Virtual GPU User Guide*.

1.1. GRID management interfaces

The local management interfaces that are supported within a GRID server are shown in Figure 1.

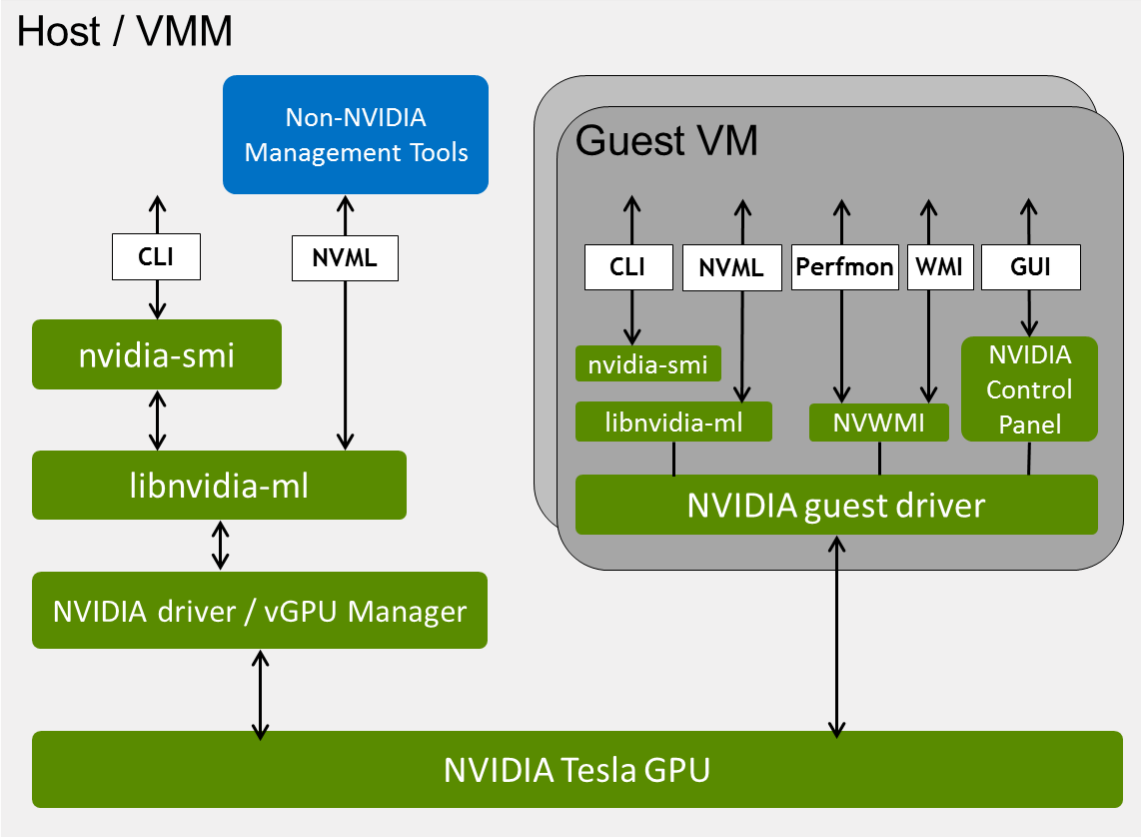

Figure 1 GRID server interfaces for GPU management

For a summary of the GRID server interfaces for GPU management, including the hypervisors and guest operating systems that support each interface, and notes about how each interface can be used, see Table 1.

Table 1 Summary of GRID server interfaces for GPU management

Interface	Hypervisor	Guest OS	Notes
`nvidia-smi` command	Any supported hypervisor	Windows 64-bit, Linux 64-bit	Command line, interactive use
NVIDIA Management Library (NVML)	Any supported hypervisor	Windows 64-bit, Linux 64-bit	Integration of NVIDIA GPU management with third-party applications
NVIDIA Control Panel	-	Windows 64-bit, Windows 32-bit	Detailed control of graphics settings, basic configuration reporting
Windows Performance Counters	-	Windows 64-bit, Windows 32-bit	Performance metrics provided by Windows Performance Counter interfaces
NVWMI	-	Windows 64-bit, Windows 32-bit	Detailed configuration and performance metrics provided by Windows WMI interfaces

1.2. Introduction to NVML

NVIDIA Management Library (NVML) is a C-based API for monitoring and managing various states of NVIDIA GPU devices. NVML is delivered in the GRID Management SDK and as a runtime version:

- The GRID Management SDK is distributed as separate archives for Windows and Linux.

  The SDK provides the NVML headers and stub libraries that are required to build third-party NVML applications. It also includes a sample application.
- The runtime version of NVML is distributed with the NVIDIA GRID host driver.

Each new version of NVML is backwards compatible, so that applications written to a version of the NVML can expect to run unchanged on future releases of GRID drivers and NVML library.

For details about the NVML API, see:

- *NVML API Reference Manual*
- NVML man pages

1.3. GRID Software Management SDK contents

The SDK consists of the NVML developer package and is distributed as separate archives for Windows and Linux:

- Windows: `grid_nvml_sdk_369.17.zip` ZIP archive
- Linux: `grid_nvml_sdk_367.43.tgz` GZIP-compressed tar archive

The contents of these archives are summarized in the following table.

Content	Windows Folder	Linux Directory
SDK Samples And Tools License Agreement		
GRID Software Management SDK User Guide (this document)		
NVML API documentation, on Linux as man pages	`nvml_sdk/doc/`	`nvml_sdk/doc/`
Sample source code and platform-dependent build files: ▸ Windows: Visual C project ▸ Linux: Make file	`nvml_sdk/example/`	`nvml_sdk/examples/`
NVML header file	`nvml_sdk/include/`	`nvml_sdk/include/`
Stub library to allow compilation on platforms without an NVIDIA driver installed	`nvml_sdk/lib/`	`nvml_sdk/lib/`

Chapter 2. MANAGING VGPUS FROM A HYPERVISOR BY USING NVML

GRID supports monitoring and control of physical GPUs and virtual GPUs that are running on virtualization hosts. NVML includes functions that are specific to managing vGPUs on GRID virtualization hosts. These functions are defined in the `nvml_grid.h` header file.

GRID does not support the management of pass-through GPUs from a hypervisor. GRID supports the management of pass-through GPUs only from within the guest VM that is using them.

2.1. Determining whether a GPU supports hosting of vGPUs

If called on platforms or GPUs that do not support GRID vGPU, functions that are specific to managing vGPUs return one of the following errors:

- `NVML_ERROR_NOT_SUPPORTED`
- `NVML_ERROR_INVALID_ARGUMENT`

To determine whether a GPU supports hosting of vGPUs, call the `nvmlDeviceGetVirtualizationMode()` function.

A vGPU-capable device reports its virtualization mode as `NVML_GPU_VIRTUALIZATION_MODE_HOST_VGPU`.

2.2. Discovering the vGPU capabilities of a physical GPU

To discover the vGPU capabilities of a physical GPU, call the functions in the following table.

Function	Purpose
`nvmlDeviceGetVirtualizationMode()`	Determine the virtualization mode of a GPU. GPUs capable of hosting virtual GPUs report their virtualization mode as `NVML_GPU_VIRTUALIZATION_MODE_HOST_VGPU`.
`nvmlDeviceGetSupportedVgpus()`	Return a list of vGPU type IDs that are supported by a GPU.
`nvmlDeviceGetCreatableVgpus()`	Return a list of vGPU type IDs that can currently be created on a GPU. The result reflects the number and type of vGPUs that are already running on the GPU.
`nvmlDeviceGetActiveVgpus()`	Return a list of handles for vGPUs currently running on a GPU.

2.3. Getting the properties of a vGPU type

To get the properties of a vGPU type, call the functions in the following table.

Function	Purpose
`nvmlVgpuTypeGetClass()`	Read the class of a vGPU type (for example, Quadro, or NVS)
`nvmlVgpuTypeGetName()`	Read the name of a vGPU type (for example, GRID M60-0Q)
`nvmlVgpuTypeGetDeviceID()`	Read PCI device ID of a vGPU type (vendor/device/subvendor/subsystem)
`nvmlVgpuTypeGetFramebufferSize()`	Read the frame buffer size of a vGPU type
`nvmlVgpuTypeGetNumDisplayHeads()`	Read the number of display heads supported by a vGPU type

Function	Purpose
`nvmlVgpuTypeGetResolution()`	Read the maximum resolution of a vGPU type's supported display head
`nvmlVgpuTypeGetLicense()`	Read license information required to operate a vGPU type
`nvmlVgpuTypeGetFrameRateLimit()`	Read the static frame limit for a vGPU type
`nvmlVgpuTypeGetMaxInstances()`	Read the maximum number of vGPU instances that can be created on a GPU

2.4. Getting the properties of a vGPU instance

To get the properties of a vGPU instance, call the functions in the following table.

Function	Purpose
`nvmlVgpuInstanceGetVmID()`	Read the ID of the VM currently associated with a vGPU instance
`nvmlVgpuInstanceGetUUID()`	Read a vGPU instance's UUID
`nvmlVgpuInstanceGetVmDriverVersion()`	Read the guest driver version currently loaded on a vGPU instance
`nvmlVgpuInstanceGetFbUsage()`	Read a vGPU instance's current frame buffer usage
`nvmlVgpuInstanceGetLicenseStatus()`	Read a vGPU instance's current license status (licensed or unlicensed)
`nvmlVgpuInstanceGetType()`	Read the vGPU type ID of a vGPU instance
`nvmlVgpuInstanceGetFrameRateLimit()`	Read a vGPU instance's frame rate limit
`nvmlDeviceGetVgpuUtilization()`	Read a vGPU instance's usage of the following resources as a percentage of the physical GPU's capacity: ▸ 3D/Compute ▸ Frame buffer bandwidth ▸ Video encoder ▸ Video decoder

2.5. Building an NVML-enabled application for a vGPU host

Fuctions that are specific to vGPUs are defined in the header file `nvml_grid.h`.

To build an NVML-enabled application for a vGPU host, ensure that you include `nvml_grid.h` in addition to `nvml.h`:

```
#include <nvml.h>
#include <nvml_grid.h>
```

For more information, refer to the sample code that is included in the SDK.

Chapter 3.
MANAGING VGPUS FROM A GUEST VM

GRID supports monitoring and control within a guest VM of vGPUs or pass-through GPUs that are assigned to the VM. The scope of management interfaces and tools used within a guest VM is limited to the guest VM within which they are used. They cannot monitor any other GPUs in the virtualization platform.

For monitoring from a guest VM, certain properties do not apply to vGPUs. The values that the GRID management interfaces report for these properties indicate that the properties do not apply to a vGPU.

3.1. GRID server interfaces for GPU management from a guest VM

The GRID server interfaces that are available for GPU management from a guest VM depend on the guest operating system that is running in the VM.

Interface	Guest OS	Notes
`nvidia-smi` command	Windows 64-bit, Linux 64-bit	Command line, interactive use
NVIDIA Management Library (NVML)	Windows 64-bit, Linux 64-bit	Integration of NVIDIA GPU management with third-party applications
NVIDIA Control Panel	Windows 64-bit, Windows 32-bit	Detailed control of graphics settings, basic configuration reporting
Windows Performance Counters	Windows 64-bit, Windows 32-bit	Performance metrics provided by Windows Performance Counter interfaces
NVWMI	Windows 64-bit, Windows 32-bit	Detailed configuration and performance metrics provided by Windows WMI interfaces

3.2. How GPU engine usage is reported

Usage of GPU engines is reported for vGPUs as a percentage of the vGPU's maximum possible capacity on each engine. The GPU engines are as follows:

- Graphics/SM
- Memory controller
- Video encoder
- Video decoder

GRID vGPUs are permitted to occupy the full capacity of each physical engine if no other vGPUs are contending for the same engine. Therefore, if a vGPU occupies 20% of the entire graphics engine in a particular sampling period, its graphics usage as reported inside the VM is 20%.

3.3. Using NVML to manage vGPUs

GRID supports monitoring and control within a guest VM by using NVML.

3.3.1. Determining whether a GPU is a vGPU or pass-through GPU

GRID vGPUs are presented in guest VM management interfaces in the same fashion as pass-through GPUs.

To determine whether a GPU device in a guest VM is a vGPU or a pass-through GPU, call the NVML function `nvmlDeviceGetVirtualizationMode()`.

A GPU reports its virtualization mode as follows:

- A GPU operating in pass-through mode reports its virtualization mode as `NVML_GPU_VIRTUALIZATION_MODE_PASSTHROUGH`.
- A vGPU reports its virtualization mode as `NVML_GPU_VIRTUALIZATION_MODE_VGPU`.

3.3.2. Physical GPU properties that do not apply to a vGPU

Properties and metrics other than GPU engine usage are reported for a vGPU in a similar way to how the same properties and metrics are reported for a physical GPU. However, some properties do not apply to vGPUs. The NVML device query functions for getting these properties return a value that indicates that the properties do not apply to a vGPU. For details of NVML device query functions, see Device Queries in *NVML API Reference Manual*.

3.3.2.1. GPU identification properties that do not apply to a vGPU

GPU Property	NVML Device Query Function	NVML return code on vGPU
Serial Number	`nvmlDeviceGetSerial()` vGPUs are not assigned serial numbers.	`NOT_SUPPORTED`

GPU Property	NVML Device Query Function	NVML return code on vGPU
GPU UUID	`nvmlDeviceGetUUID()` vGPUs are allocated random UUIDs.	`SUCCESS`
VBIOS Version	`nvmlDevicenvmlDeviceGetVbiosVersion()` vGPU VBIOS version is hard-wired to zero.	`SUCCESS`
GPU Part Number	`nvmlDeviceGetBoardPartNumber()`	`NOT_SUPPORTED`

3.3.2.2. `InfoROM` properties that do not apply to a vGPU

The `InfoROM` object is not exposed on vGPUs. All the functions in the following table return `NOT_SUPPORTED`.

GPU Property	NVML Device Query Function
Image Version	`nvmlDeviceGetInforomImageVersion()`
OEM Object	`nvmlDeviceGetInforomVersion()`
ECC Object	`nvmlDeviceGetInforomVersion()`
Power Management Object	`nvmlDeviceGetInforomVersion()`

3.3.2.3. GPU operation mode properties that do not apply to a vGPU

GPU Property	NVML Device Query Function	NVML return code on vGPU
GPU Operation Mode (Current)	`nvmlDeviceGetGpuOperationMode()` Tesla GPU operating modes are not supported on vGPUs.	`NOT_SUPPORTED`
GPU Operation Mode (Pending)	`nvmlDeviceGetGpuOperationMode()` Tesla GPU operating modes are not supported on vGPUs.	`NOT_SUPPORTED`
Compute Mode	`nvmlDeviceGetComputeMode()` A vGPU always returns `NVML_COMPUTEMODE_PROHIBITED`.	`SUCCESS`
Driver Model	`nvmlDeviceGetDriverModel()` A vGPU supports WDDM mode only in Windows VMs.	`SUCCESS` (Windows)

3.3.2.4. PCI Express properties that do not apply to a vGPU

PCI Express characteristics are not exposed on vGPUs. All the functions in the following table return `NOT_SUPPORTED`.

GPU Property	NVML Device Query Function
Generation Max	`nvmlDeviceGetMaxPcieLinkGeneration()`
Generation Current	`nvmlDeviceGetCurrPcieLinkGeneration()`

GPU Property	NVML Device Query Function
Link Width Max	nvmlDeviceGetMaxPcieLinkWidth()
Link Width Current	nvmlDeviceGetCurrPcieLinkWidth()
Bridge Chip Type	nvmlDeviceGetBridgeChipInfo()
Bridge Chip Firmware	nvmlDeviceGetBridgeChipInfo()
Replays	nvmlDeviceGetPcieReplayCounter()
TX Throughput	nvmlDeviceGetPcieThroughput()
RX Throughput	nvmlDeviceGetPcieThroughput()

3.3.2.5. Environmental properties that do not apply to a vGPU

All the functions in the following table return NOT_SUPPORTED.

GPU Property	NVML Device Query Function
Fan Speed	nvmlDeviceGetFanSpeed()
Clocks Throttle Reasons	nvmlDeviceGetSupportedClocksThrottleReasons() nvmlDeviceGetCurrentClocksThrottleReasons()
Current Temperature	nvmlDeviceGetTemperature() nvmlDeviceGetTemperatureThreshold()
Shutdown Temperature	nvmlDeviceGetTemperature() nvmlDeviceGetTemperatureThreshold()
Slowdown Temperature	nvmlDeviceGetTemperature() nvmlDeviceGetTemperatureThreshold()

3.3.2.6. Power consumption properties that do not apply to a vGPU

vGPUs do not expose physical power consumption of the underlying GPU. All the functions in the following table return NOT_SUPPORTED.

GPU Property	NVML Device Query Function
Management Mode	nvmlDeviceGetPowerManagementMode()
Draw	nvmlDeviceGetPowerUsage()
Limit	nvmlDeviceGetPowerManagementLimit()
Default Limit	nvmlDeviceGetPowerManagementDefaultLimit()
Enforced Limit	nvmlDeviceGetEnforcedPowerLimit()
Min Limit	nvmlDeviceGetPowerManagementLimitConstraints()
Max Limit	nvmlDeviceGetPowerManagementLimitConstraints()

3.3.2.7. ECC properties that do not apply to a vGPU

Error-correcting code (ECC) is not supported on vGPUs. All the functions in the following table return `NOT_SUPPORTED`.

GPU Property	NVML Device Query Function
Mode	`nvmlDeviceGetEccMode()`
Error Counts	`nvmlDeviceGetMemoryErrorCounter()` `nvmlDeviceGetTotalEccErrors()`
Retired Pages	`nvmlDeviceGetRetiredPages()` `nvmlDeviceGetRetiredPagesPendingStatus()`

3.3.2.8. Clocks properties that do not apply to a vGPU

All the functions in the following table return `NOT_SUPPORTED`.

GPU Property	NVML Device Query Function
Application Clocks	`nvmlDeviceGetApplicationsClock()`
Default Application Clocks	`nvmlDeviceGetDefaultApplicationsClock()`
Max Clocks	`nvmlDeviceGetMaxClockInfo()`
Policy: Auto Boost	`nvmlDeviceGetAutoBoostedClocksEnabled()`
Policy: Auto Boost Default	`nvmlDeviceGetAutoBoostedClocksEnabled()`

3.3.3. Building an NVML-enabled application for a guest VM

To build an NVML-enabled application, refer to the sample code included in the SDK.

3.4. Using Windows Performance Counters to monitor GPU performance

In Windows VMs, GPU metrics are available as Windows Performance Counters through the `NVIDIA GPU` object.

For access to Windows Performance Counters through programming interfaces, refer to the performance counter sample code included with the NVIDIA Windows Management Instrumentation SDK.

On vGPUs, the following GPU performance counters read as 0 because they are not applicable to vGPUs:

- % Bus Usage

- % Cooler rate
- Core Clock MHz
- Fan Speed
- Memory Clock MHz
- PCI-E current speed to GPU Mbps
- PCI-E current width to GPU
- PCI-E downstream width to GPU
- Power Consumption mW
- Temperature C

3.5. Using NVWMI to monitor GPU performance

In Windows VMs, Windows Management Instrumentation (WMI) exposes GPU metrics in the `ROOT\CIMV2\NV` namespace through NVWMI. NVWMI is included with the NVIDIA driver package. After the driver is installed, NVWMI help information in Windows Help format is available as follows:

```
C:\Program Files\NVIDIA Corporation\NVIDIA WMI Provider>nvwmi.chm
```

For access to NVWMI through programming interfaces, use the NVWMI SDK. The NVWMI SDK, with white papers and sample programs, is included in the NVIDIA Windows Management Instrumentation SDK.

On vGPUs, some instance properties of the following classes do not apply to vGPUs:

- `Ecc`
- `Gpu`
- `PcieLink`

`Ecc` instance properties that do not apply to vGPUs

`Ecc` Instance Property	Value reported on vGPU
`isSupported`	False
`isWritable`	False
`isEnabled`	False
`isEnabledByDefault`	False
`aggregateDoubleBitErrors`	0
`aggregateSingleBitErrors`	0
`currentDoubleBitErrors`	0
`currentSingleBitErrors`	0

`Gpu` instance properties that do not apply to vGPUs

`Gpu` Instance Property	Value reported on vGPU
`gpuCoreClockCurrent`	-1
`memoryClockCurrent`	-1
`pciDownstreamWidth`	0
`pcieGpu.curGen`	0
`pcieGpu.curSpeed`	0
`pcieGpu.curWidth`	0
`pcieGpu.maxGen`	1
`pcieGpu.maxSpeed`	2500
`pcieGpu.maxWidth`	0
`power`	-1
`powerSampleCount`	-1
`powerSamplingPeriod`	-1
`verVBIOS.orderedValue`	0
`verVBIOS.strValue`	-
`verVBIOS.value`	0

`PcieLink` instance properties that do not apply to vGPUs

No instances of `PcieLink` are reported for vGPU.

Notice

HDMI

OpenCL

Trademarks

Copyright