



VIRTUAL GPU SOFTWARE MANAGEMENT SDK

DU-08141-001 _v7.0 through 7.5 Revision 02 | May 2020

User Guide



TABLE OF CONTENTS

Chapter 1. Introduction to the NVIDIA vGPU Software Management SDK.....	1
1.1. NVIDIA vGPU Software Management Interfaces.....	1
1.2. Introduction to NVML.....	3
1.3. NVIDIA vGPU Software Management SDK contents.....	3
Chapter 2. Managing vGPUs from a hypervisor by using NVML.....	5
2.1. Determining whether a GPU supports hosting of vGPUs.....	5
2.2. Discovering the vGPU capabilities of a physical GPU.....	6
2.3. Getting the properties of a vGPU type.....	6
2.4. Getting the properties of a vGPU instance.....	7
2.5. Building an NVML-enabled application for a vGPU host.....	10
Chapter 3. Managing vGPUs from a guest VM.....	11
3.1. NVIDIA vGPU Software Server Interfaces for GPU Management from a Guest VM.....	11
3.2. How GPU engine usage is reported.....	11
3.3. Using NVML to manage vGPUs.....	12
3.3.1. Determining whether a GPU is a vGPU or pass-through GPU.....	12
3.3.2. Physical GPU properties that do not apply to a vGPU.....	12
3.3.2.1. GPU identification properties that do not apply to a vGPU.....	13
3.3.2.2. InfoROM properties that do not apply to a vGPU.....	13
3.3.2.3. GPU operation mode properties that do not apply to a vGPU.....	13
3.3.2.4. PCI Express properties that do not apply to a vGPU.....	14
3.3.2.5. Environmental properties that do not apply to a vGPU.....	14
3.3.2.6. Power consumption properties that do not apply to a vGPU.....	14
3.3.2.7. ECC properties that do not apply to a vGPU.....	15
3.3.2.8. Clocks properties that do not apply to a vGPU.....	15
3.3.3. Building an NVML-enabled application for a guest VM.....	15
3.4. Using Windows Performance Counters to monitor GPU performance.....	15
3.5. Using NVWMI to monitor GPU performance.....	16

LIST OF FIGURES

Figure 1 NVIDIA vGPU Software server interfaces for GPU management	2
--	---

LIST OF TABLES

Table 1 Summary of NVIDIA vGPU Software server interfaces for GPU management	2
--	---

Chapter 1.

INTRODUCTION TO THE NVIDIA vGPU SOFTWARE MANAGEMENT SDK

The NVIDIA vGPU 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 vGPU software Management SDK supports control and monitoring of GPUs from both the hypervisor host system and from within guest VMs.

NVIDIA vGPU software 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 vGPU software, see *Virtual GPU Software User Guide*.

1.1. NVIDIA vGPU Software Management Interfaces

The local management interfaces that are supported within an NVIDIA vGPU software server are shown in [Figure 1](#).

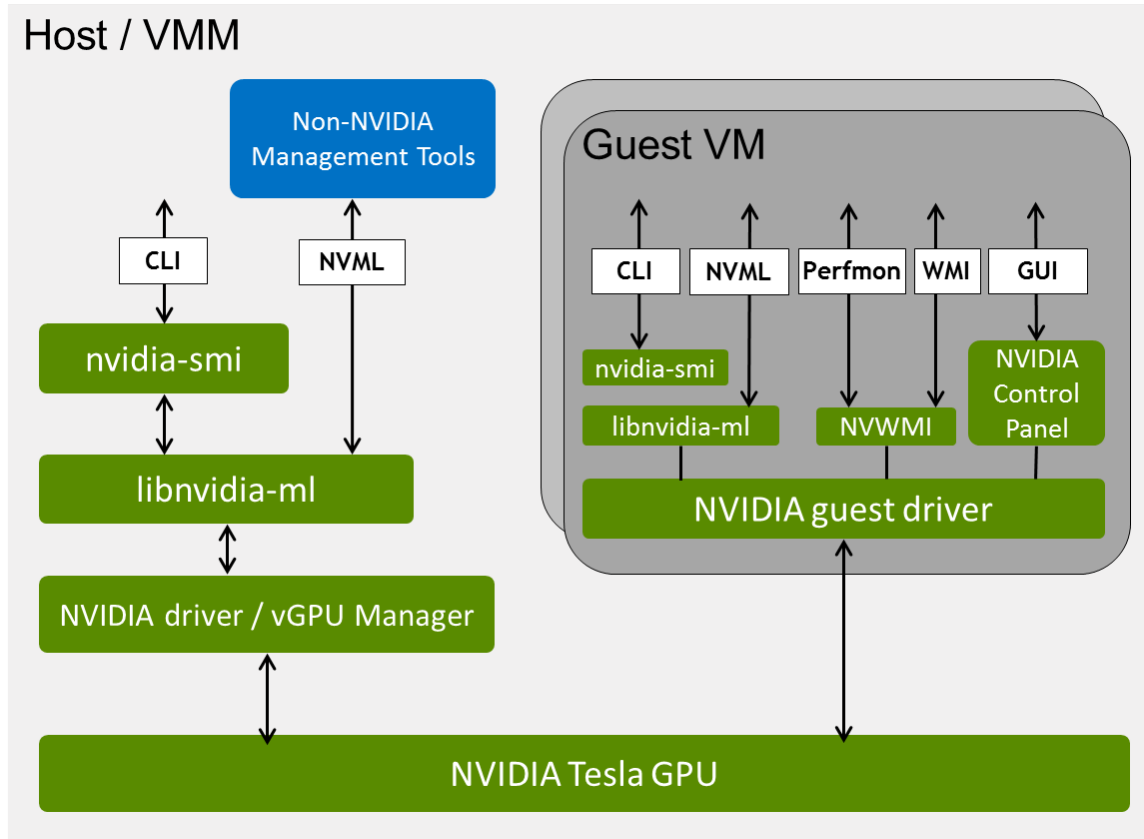


Figure 1 NVIDIA vGPU Software server interfaces for GPU management

For a summary of the NVIDIA vGPU software 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 NVIDIA vGPU Software server interfaces for GPU management

Interface	Hypervisor	Guest OS	Notes
<code>nvidia-smi</code> command	Any supported hypervisor	Windows, 64-bit Linux	Command line, interactive use
NVIDIA Management Library (NVML)	Any supported hypervisor	Windows, 64-bit Linux	Integration of NVIDIA GPU management with third-party applications
NVIDIA Control Panel	-	Windows	Detailed control of graphics settings, basic configuration reporting
Windows Performance Counters	-	Windows	Performance metrics provided by Windows Performance Counter interfaces
NVWMI	-	Windows	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 NVIDIA vGPU software Management SDK and as a runtime version:

- ▶ The NVIDIA vGPU software 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 vGPU software 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 the NVIDIA vGPU software drivers and NVML library.

For details about the NVML API, see:

- ▶ [NVML API Reference Manual](#)
- ▶ NVML man pages

1.3. NVIDIA vGPU 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_412.47.zip` ZIP archive
- ▶ Linux: `grid_nvml_sdk_410.141.tgz` GZIP-compressed tar archive

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

Content	Windows Folder	Linux Directory
<i>SDK Samples And Tools License Agreement</i>		
<i>Virtual GPU Software Management SDK User Guide</i> (this document)		
NVML API documentation, on Linux as man pages	<code>nvml_sdk/doc/</code>	<code>nvml_sdk/doc/</code>
Sample source code and platform-dependent build files: <ul style="list-style-type: none"> ▶ Windows: Visual C project ▶ Linux: Make file 	<code>nvml_sdk/example/</code>	<code>nvml_sdk/examples/</code>
NVML header file	<code>nvml_sdk/include/</code>	<code>nvml_sdk/include/</code>

Content	Windows Folder	Linux Directory
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

NVIDIA vGPU software 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 NVIDIA vGPU software virtualization hosts. These functions are defined in the `nvml_grid.h` header file.



NVIDIA vGPU software does not support the management of pass-through GPUs from a hypervisor. NVIDIA vGPU software 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 NVIDIA 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
<code>nvmlDeviceGetVirtualizationMode()</code>	Determine the virtualization mode of a GPU. GPUs capable of hosting virtual GPUs report their virtualization mode as <code>NVML_GPU_VIRTUALIZATION_MODE_HOST_VGPU</code> .
<code>nvmlDeviceGetSupportedVgpus()</code>	Return a list of vGPU type IDs that are supported by a GPU.
<code>nvmlDeviceGetCreatableVgpus()</code>	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.
<code>nvmlDeviceGetActiveVgpus()</code>	Return a list of handles for vGPUs currently running on a GPU.
<code>nvmlDeviceGetVgpuMetadata()</code>	Return a vGPU metadata structure for the physical 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
<code>nvmlVgpuTypeGetClass()</code>	Read the class of a vGPU type (for example, Quadro, or NVS)
<code>nvmlVgpuTypeGetName()</code>	Read the name of a vGPU type (for example, GRID M60-0Q)
<code>nvmlVgpuTypeGetDeviceID()</code>	Read PCI device ID of a vGPU type (vendor/device/subvendor/subsystem)
<code>nvmlVgpuTypeGetFramebufferSize()</code>	Read the frame buffer size of a vGPU type

Function	Purpose
<code>nvmlVgpuTypeGetNumDisplayHeads()</code>	Read the number of display heads supported by a vGPU type
<code>nvmlVgpuTypeGetResolution()</code>	Read the maximum resolution of a vGPU type's supported display head
<code>nvmlVgpuTypeGetLicense()</code>	Read license information required to operate a vGPU type
<code>nvmlVgpuTypeGetFrameRateLimit()</code>	Read the static frame limit for a vGPU type
<code>nvmlVgpuTypeGetMaxInstances()</code>	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
<code>nvmlVgpuInstanceGetVmID()</code>	Read the ID of the VM currently associated with a vGPU instance
<code>nvmlVgpuInstanceGetUUID()</code>	Read a vGPU instance's UUID
<code>nvmlVgpuInstanceGetVmDriverVersion()</code>	Read the guest driver version currently loaded on a vGPU instance
<code>nvmlVgpuInstanceGetFbUsage()</code>	Read a vGPU instance's current frame buffer usage
<code>nvmlVgpuInstanceGetFBCStats()</code>	Read the following frame buffer capture (FBC) statistics for a vGPU instance: <ul style="list-style-type: none"> ▶ Count of active FBC sessions ▶ Moving average of new frames captured per second by all active sessions ▶ Moving average of new frame capture latency in microseconds for all active sessions
<code>nvmlVgpuInstanceGetFBCSessions()</code>	For each active FBC session on a vGPU instance, read the following statistics: <ul style="list-style-type: none"> ▶ FBC session ID ▶ Owing PID ▶ vGPU instance identifier

Function	Purpose
	<ul style="list-style-type: none"> ▶ Display ordinal associated with the FBC session. ▶ FBC session type ▶ FBC session flags ▶ Maximum horizontal resolution supported by the session ▶ Maximum vertical resolution supported by the session ▶ Horizontal resolution requested by the caller in the capture call ▶ Vertical resolution requested by the caller in the capture call ▶ Moving average of new frames captured per second by the session ▶ Moving average new frame capture latency in microseconds for the session
<code>nvmlVgpuInstanceGetLicenseStatus ()</code>	Read a vGPU instance's current license status (licensed or unlicensed)
<code>nvmlVgpuInstanceGetType ()</code>	Read the vGPU type ID of a vGPU instance
<code>nvmlVgpuInstanceGetFrameRateLimit ()</code>	Read a vGPU instance's frame rate limit
<code>nvmlVgpuInstanceGetEncoderStats ()</code>	Read the following encoder statistics for a vGPU instance: <ul style="list-style-type: none"> ▶ Count of active encoder sessions ▶ One-second trailing average of encoded FPS of all active sessions ▶ One-second trailing average of encode latency in microseconds
<code>nvmlVgpuInstanceGetEncoderSessions ()</code>	For each active encoder session on a vGPU instance, read the following statistics: <ul style="list-style-type: none"> ▶ Encoder session ID ▶ Owning PID ▶ Codec type, for example, H.264 or H.265 ▶ Encode resolution ▶ One-second trailing averages for encoded FPS and encode latency

Function	Purpose
<code>nvmlDeviceGetVgpuUtilization()</code>	<p>Read a vGPU instance's usage of the following resources as a percentage of the physical GPU's capacity:</p> <ul style="list-style-type: none"> ▶ 3D/Compute ▶ Frame buffer bandwidth ▶ Video encoder ▶ Video decoder
<code>nvmlDeviceGetVgpuProcessUtilization()</code>	<p>For each process running on a vGPU instance, read the process ID and usage by the process of the following resources as a percentage of the physical GPU's capacity:</p> <ul style="list-style-type: none"> ▶ 3D/Compute ▶ Frame buffer bandwidth ▶ Video encoder ▶ Video decoder
<code>nvmlVgpuInstanceGetAccountingMode()</code>	<p>Read the accounting mode of the vGPU instance</p>
<code>nvmlVgpuInstanceGetAccountingPids()</code>	<p>Read the maximum number of processes that can be queried and the current list of process IDs</p>
<code>nvmlVgpuInstanceGetAccountingStats()</code>	<p>For each process ID returned by <code>nvmlVgpuInstanceGetAccountingPids()</code>, read the following statistics:</p> <ul style="list-style-type: none"> ▶ GPU utilization ▶ Memory utilization ▶ Maximum memory usage ▶ Amount of time the graphics or compute context was active ▶ Start time of the process ▶ Whether the process is still running
<code>nvmlGetVgpuCompatibility()</code>	<p>Return compatibility information about a vGPU and a physical GPU, such as:</p> <ul style="list-style-type: none"> ▶ The VM states from which the vGPU can be run on the physical GPU ▶ Any factors limiting compatibility between the vGPU and the physical GPU

Function	Purpose
nvmlVgpuInstanceGetMetadata()	Return a vGPU metadata structure for a vGPU and its associated VM

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

Functions 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

NVIDIA vGPU software 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 NVIDIA vGPU software management interfaces report for these properties indicate that the properties do not apply to a vGPU.

3.1. NVIDIA vGPU Software Server Interfaces for GPU Management from a Guest VM

The NVIDIA vGPU software 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
<code>nvidia-smi</code> command	Windows, 64-bit Linux	Command line, interactive use
NVIDIA Management Library (NVML)	Windows, 64-bit Linux	Integration of NVIDIA GPU management with third-party applications
NVIDIA Control Panel	Windows	Detailed control of graphics settings, basic configuration reporting
Windows Performance Counters	Windows	Performance metrics provided by Windows Performance Counter interfaces
NVWMI	Windows	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

The amount of a physical engine's capacity that a vGPU is permitted to occupy depends on the scheduler under which the GPU is operating:

- ▶ NVIDIA vGPUs operating under the **Best Effort Scheduler** and the **Equal Share Scheduler** 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%.
- ▶ NVIDIA vGPUs operating under the **Equal Share Scheduler** can occupy no more than their allocated share of the graphics engine. Therefore, if a vGPU has a fixed allocation of 25% of the graphics engine, and it occupies 25% of the engine in a particular sampling period, its graphics usage as reported inside the VM is 100%.

3.3. Using NVML to manage vGPUs

NVIDIA vGPU software 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

NVIDIA 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	<code>nvmlDeviceGetSerial()</code> vGPUs are not assigned serial numbers.	NOT_SUPPORTED
GPU UUID	<code>nvmlDeviceGetUUID()</code> vGPUs are allocated random UUIDs.	SUCCESS
VBIOS Version	<code>nvmlDeviceGetVbiosVersion()</code> vGPU VBIOS version is hard-wired to zero.	SUCCESS
GPU Part Number	<code>nvmlDeviceGetBoardPartNumber()</code>	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	<code>nvmlDeviceGetInforomImageVersion()</code>
OEM Object	<code>nvmlDeviceGetInforomVersion()</code>
ECC Object	<code>nvmlDeviceGetInforomVersion()</code>
Power Management Object	<code>nvmlDeviceGetInforomVersion()</code>

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)	<code>nvmlDeviceGetGpuOperationMode()</code> Tesla GPU operating modes are not supported on vGPUs.	NOT_SUPPORTED
GPU Operation Mode (Pending)	<code>nvmlDeviceGetGpuOperationMode()</code> Tesla GPU operating modes are not supported on vGPUs.	NOT_SUPPORTED
Compute Mode	<code>nvmlDeviceGetComputeMode()</code> A vGPU always returns NVML_COMPUTEMODE_PROHIBITED.	SUCCESS
Driver Model	<code>nvmlDeviceGetDriverModel()</code> 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	<code>nvmlDeviceGetMaxPcieLinkGeneration()</code>
Generation Current	<code>nvmlDeviceGetCurrPcieLinkGeneration()</code>
Link Width Max	<code>nvmlDeviceGetMaxPcieLinkWidth()</code>
Link Width Current	<code>nvmlDeviceGetCurrPcieLinkWidth()</code>
Bridge Chip Type	<code>nvmlDeviceGetBridgeChipInfo()</code>
Bridge Chip Firmware	<code>nvmlDeviceGetBridgeChipInfo()</code>
Replays	<code>nvmlDeviceGetPcieReplayCounter()</code>
TX Throughput	<code>nvmlDeviceGetPcieThroughput()</code>
RX Throughput	<code>nvmlDeviceGetPcieThroughput()</code>

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	<code>nvmlDeviceGetFanSpeed()</code>
Clocks Throttle Reasons	<code>nvmlDeviceGetSupportedClocksThrottleReasons()</code> <code>nvmlDeviceGetCurrentClocksThrottleReasons()</code>
Current Temperature	<code>nvmlDeviceGetTemperature()</code> <code>nvmlDeviceGetTemperatureThreshold()</code>
Shutdown Temperature	<code>nvmlDeviceGetTemperature()</code> <code>nvmlDeviceGetTemperatureThreshold()</code>
Slowdown Temperature	<code>nvmlDeviceGetTemperature()</code> <code>nvmlDeviceGetTemperatureThreshold()</code>

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	<code>nvmlDeviceGetPowerManagementMode()</code>
Draw	<code>nvmlDeviceGetPowerUsage()</code>
Limit	<code>nvmlDeviceGetPowerManagementLimit()</code>

GPU Property	NVML Device Query Function
Default Limit	<code>nvmlDeviceGetPowerManagementDefaultLimit()</code>
Enforced Limit	<code>nvmlDeviceGetEnforcedPowerLimit()</code>
Min Limit	<code>nvmlDeviceGetPowerManagementLimitConstraints()</code>
Max Limit	<code>nvmlDeviceGetPowerManagementLimitConstraints()</code>

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	<code>nvmlDeviceGetEccMode()</code>
Error Counts	<code>nvmlDeviceGetMemoryErrorCounter()</code> <code>nvmlDeviceGetTotalEccErrors()</code>
Retired Pages	<code>nvmlDeviceGetRetiredPages()</code> <code>nvmlDeviceGetRetiredPagesPendingStatus()</code>

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	<code>nvmlDeviceGetApplicationsClock()</code>
Default Application Clocks	<code>nvmlDeviceGetDefaultApplicationsClock()</code>
Max Clocks	<code>nvmlDeviceGetMaxClockInfo()</code>
Policy: Auto Boost	<code>nvmlDeviceGetAutoBoostedClocksEnabled()</code>
Policy: Auto Boost Default	<code>nvmlDeviceGetAutoBoostedClocksEnabled()</code>

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

Ecc Instance Property	Value reported on vGPU
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

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA, the NVIDIA logo, NVIDIA GRID, vGPU, Pascal, Quadro, and Tesla are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2013-2020 NVIDIA Corporation. All rights reserved.