

# **NVML API REFERENCE MANUAL** May 16, 2012 Version 3.295.54

# **Contents**

1	NVI	ML API Reference	1
	1.1	Feature Matrix	3
2	Cha	ange log of NVML library	7
	2.1	Changes between NVML v1.0 and v2.285	8
	2.2	Changes between NVML v2.285 and v3.295	8
3	Dep	precated List	11
4	Mod	dule Index	13
	4.1	Modules	13
5	Data	ta Structure Index	15
	5.1	Data Structures	15
6	Mod	dule Documentation	17
	6.1	Device Structs	17
		6.1.1 Define Documentation	17
		6.1.1.1 NVML_VALUE_NOT_AVAILABLE	17
	6.2	Device Enums	18
		6.2.1 Enumeration Type Documentation	19
		6.2.1.1 nvmlClockType_t	19
		6.2.1.2 nvmlComputeMode_t	20
		6.2.1.3 nvmlDriverModel_t	20
		6.2.1.4 nvmlEccBitType_t	20
		6.2.1.5 nvmlEccCounterType_t	20
		6.2.1.6 nvmlEnableState_t	21
		6.2.1.7 nvmlInforomObject_t	21
		6.2.1.8 nvmlPstates_t	21
		6.2.1.9 nymlReturn t	22

ii CONTENTS

		6.2.1.10	nvmlTemperatureSensors_t	22
6.3	Unit S	tructs		23
	6.3.1	Enumerat	tion Type Documentation	23
		6.3.1.1	nvmlFanState_t	23
		6.3.1.2	nvmlLedColor_t	23
6.4	Event '	Types		24
	6.4.1	Detailed 1	Description	24
	6.4.2	Define De	ocumentation	24
		6.4.2.1	nvmlEventTypePState	24
6.5	Initiali	zation and	Cleanup	25
	6.5.1	Detailed 1	Description	25
	6.5.2	Function	Documentation	25
		6.5.2.1	nvmlInit	25
		6.5.2.2	nvmlShutdown	25
6.6	Error r	reporting		26
	6.6.1	Detailed 1	Description	26
	6.6.2	Function	Documentation	26
		6.6.2.1	nvmlErrorString	26
6.7	Consta	ants		27
	6.7.1	Define De	ocumentation	27
		6.7.1.1	NVML_DEVICE_INFOROM_VERSION_BUFFER_SIZE	27
		6.7.1.2	NVML_DEVICE_NAME_BUFFER_SIZE	27
		6.7.1.3	NVML_DEVICE_SERIAL_BUFFER_SIZE	27
		6.7.1.4	NVML_DEVICE_UUID_BUFFER_SIZE	27
		6.7.1.5	NVML_DEVICE_VBIOS_VERSION_BUFFER_SIZE	27
		6.7.1.6	NVML_SYSTEM_DRIVER_VERSION_BUFFER_SIZE	27
		6.7.1.7	NVML_SYSTEM_NVML_VERSION_BUFFER_SIZE	27
6.8	System	n Queries		28
	6.8.1	Detailed l	Description	28
	6.8.2	Function	Documentation	28
		6.8.2.1	nvmlSystemGetDriverVersion	28
		6.8.2.2	nvmlSystemGetNVMLVersion	28
		6.8.2.3	nvmlSystemGetProcessName	29
6.9	Unit Q	Queries		30
	6.9.1	Detailed l	Description	30
	6.9.2	Function	Documentation	30
		6.9.2.1	nvmlSystemGetHicVersion	30

CONTENTS

6.9.2.2	nvmlUnitGetCount	30
6.9.2.3	nvmlUnitGetDevices	31
6.9.2.4	nvmlUnitGetFanSpeedInfo	31
6.9.2.5	nvmlUnitGetHandleByIndex	32
6.9.2.6	nvmlUnitGetLedState	32
6.9.2.7	nvmlUnitGetPsuInfo	32
6.9.2.8	nvmlUnitGetTemperature	33
6.9.2.9	nvmlUnitGetUnitInfo	33
6.10 Device Queries	8	34
6.10.1 Detaile	ed Description	35
6.10.2 Function	on Documentation	35
6.10.2.	1 nvmlDeviceGetClockInfo	35
6.10.2.	2 nvmlDeviceGetComputeMode	35
6.10.2.	3 nvmlDeviceGetComputeRunningProcesses	36
6.10.2.	4 nvmlDeviceGetCount	36
6.10.2	5 nvmlDeviceGetCurrPcieLinkGeneration	37
6.10.2.	6 nvmlDeviceGetCurrPcieLinkWidth	37
6.10.2.	7 nvmlDeviceGetDetailedEccErrors	38
6.10.2.	8 nvmlDeviceGetDisplayMode	38
6.10.2.	9 nvmlDeviceGetDriverModel	39
6.10.2.	10 nvmlDeviceGetEccMode	39
6.10.2.	11 nvmlDeviceGetFanSpeed	40
6.10.2.	12 nvmlDeviceGetHandleByIndex	40
6.10.2.	13 nvmlDeviceGetHandleByPciBusId	41
6.10.2.	14 nvmlDeviceGetHandleBySerial	41
6.10.2.	15 nvmlDeviceGetHandleByUUID	42
6.10.2.	16 nvmlDeviceGetInforomVersion	42
6.10.2.	17 nvmlDeviceGetMaxClockInfo	43
6.10.2.	18 nvmlDeviceGetMaxPcieLinkGeneration	43
6.10.2.	19 nvmlDeviceGetMaxPcieLinkWidth	44
6.10.2.	20 nvmlDeviceGetMemoryInfo	44
6.10.2.	21 nvmlDeviceGetName	45
6.10.2.	22 nvmlDeviceGetPciInfo	45
6.10.2.	23 nvmlDeviceGetPerformanceState	45
6.10.2.	24 nvmlDeviceGetPersistenceMode	46
6.10.2.	25 nvmlDeviceGetPowerManagementLimit	46
6.10.2.	26 nvmlDeviceGetPowerManagementMode	47

iv CONTENTS

		6.10.2.27 nvmlDeviceGetPowerState
		6.10.2.28 nvmlDeviceGetPowerUsage
		6.10.2.29 nvmlDeviceGetSerial
		6.10.2.30 nvmlDeviceGetTemperature
		6.10.2.31 nvmlDeviceGetTotalEccErrors
		6.10.2.32 nvmlDeviceGetUtilizationRates
		6.10.2.33 nvmlDeviceGetUUID
		6.10.2.34 nvmlDeviceGetVbiosVersion
		6.10.2.35 nvmlDeviceOnSameBoard
6.11	Unit C	ommands
	6.11.1	Detailed Description
	6.11.2	Function Documentation
		6.11.2.1 nvmlUnitSetLedState
6.12	Device	Commands
	6.12.1	Detailed Description
	6.12.2	Function Documentation
		6.12.2.1 nvmlDeviceClearEccErrorCounts
		6.12.2.2 nvmlDeviceSetComputeMode
		6.12.2.3 nvmlDeviceSetDriverModel
		6.12.2.4 nvmlDeviceSetEccMode
		6.12.2.5 nvmlDeviceSetPersistenceMode
6.13	Event	Handling Methods
		Detailed Description
	6.13.2	Typedef Documentation
		6.13.2.1 nvmlEventSet_t
	6.13.3	Function Documentation
		6.13.3.1 nvmlDeviceGetSupportedEventTypes
		6.13.3.2 nvmlDeviceRegisterEvents
		6.13.3.3 nvmlEventSetCreate
		6.13.3.4 nvmlEventSetFree
		6.13.3.5 nvmlEventSetWait
Data	a Struct	ure Documentation 65
7.1	nvmlE	ccErrorCounts_t Struct Reference
	7.1.1	Detailed Description
7.2	nvmlE	ventData_t Struct Reference
	7.2.1	Detailed Description

7

CONTENTS

7.3	nvmlHwbcEntry_t Struct Reference	65
	7.3.1 Detailed Description	65
7.4	nvmlLedState_t Struct Reference	66
	7.4.1 Detailed Description	66
7.5	nvmlMemory_t Struct Reference	67
	7.5.1 Detailed Description	67
7.6	nvmlPciInfo_t Struct Reference	68
	7.6.1 Detailed Description	68
7.7	nvmlProcessInfo_t Struct Reference	69
	7.7.1 Detailed Description	69
7.8	nvmlPSUInfo_t Struct Reference	70
	7.8.1 Detailed Description	70
7.9	nvmlUnitFanInfo_t Struct Reference	71
	7.9.1 Detailed Description	71
7.10	nvmlUnitFanSpeeds_t Struct Reference	72
	7.10.1 Detailed Description	72
7.11	nvmlUnitInfo_t Struct Reference	73
	7.11.1 Detailed Description	73
7.12	nvmlUtilization_t Struct Reference	74
	7.12.1 Detailed Description	74

# Chapter 1

# **NVML API Reference**

The NVIDIA Management Library (NVML) is a C-based programmatic interface for monitoring and managing various states within NVIDIA Tesla <sup>TM</sup>GPUs.

It is intended to be a platform for building 3rd party applications, and is also the underlying library for the NVIDIA-supported nvidia-smi tool.

NVML is thread-safe so it is safe to make simultaneous NVML calls from multiple threads.

# **API Documentation**

Supported OS platforms:

• Windows: Windows Server 2008 R2 64bit, Windows 7 64bit

• Linux: 32-bit and 64-bit

# Supported products:

- Full Support
  - NVIDIA Tesla ™Line: S1070, S2050, C1060, C2050/70/75, M2050/70/75/90, X2070/90
  - NVIDIA Quadro ®Line: 4000, 5000, 6000, 7000, M2070-Q
  - NVIDIA GeForce ®Line: None
- · Limited Support
  - NVIDIA Tesla TMLine: None
  - NVIDIA Quadro ®Line: All other current and previous generation Quadro-branded parts
  - NVIDIA GeForce ®Line: All current and previous generation GeForce-branded parts

The NVML library can be found at %ProgramW6432%\"NVIDIA Corporation"\NVSMI\ on Windows, but will not be added to the path. To dynamically link to NVML, add this path to the PATH environmental variable. To dynamically load NVML, call LoadLibrary with this path.

On Linux the NVML library will be found on the standard library path. For 64 bit Linux, both the 32 bit and 64 bit NVML libraries will be installed.

The NVML API is divided into five categories:

2 NVML API Reference

- Support Methods:
  - Initialization and Cleanup
- Query Methods:
  - System Queries
  - Device Queries
  - Unit Queries
- Control Methods:
  - Unit Commands
  - Device Commands
- Event Handling Methods:
  - Event Handling Methods
- Error reporting Methods
  - Error reporting

List of changes can be found in the Changelog

1.1 Feature Matrix 3

# 1.1 Feature Matrix

	S1070	S2050				
	✓	<b>✓</b>				
	✓	✓				
	Firmware Version					
	Attached GPUs					
LED State	Color	✓	✓			
LED State	Cause	✓	✓			
	Intake	✓	✓			
Temperature	Exhaust	×	×			
	Board	×	×			
	PSU State	✓	✓			
PSU	Voltage	✓	✓			
	Current	✓	✓			
Fanc	Fan Speed	✓	✓			
Fans	Fan State	✓	✓			

Commands	S1070	S2050
Toggle LED State	<b>✓</b>	<b>✓</b>

Figure 1.1: This chart shows which unit-level features are available for each S-class product. All GPUs within each S-class product also provide the information listed in the Device chart below.

NVML API Reference

Queries			C2050	C2070	C2075	M2050	M2070	M2075	M2090	S2050	X2070	X2090	Gemini
Е	loard Serial Nu	ımber	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	GPU UUIC		✓	✓	<b>√</b>	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
	VBios Versio	on	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>√</b>
		OEM Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
		ECC Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
		Power Object	×	×	✓	×	×	✓	✓	×	×	✓	×
PCI Info			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
Compute Mode			✓	✓	<b>√</b>	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
Display Mode			✓	✓	✓	×	×	×	×	×	×	×	×
Persis	tence Mode (L	inux-Only)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
		Current	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ECC Mode		Pending	✓	✓	<b>✓</b>	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
Driver Model		Current	✓	✓	<b>✓</b>	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
		Pending	✓	✓	<b>√</b>	<b>√</b>	✓	✓	✓	✓	✓	✓	<b>✓</b>
	Fan Speed		✓	✓	<b>√</b>	×	×	×	×	×	×	×	×
	GPU Tempera	ture	<b>√</b>	<b>√</b>	<b>√</b>	×	×	×	×	✓	×	×	<b>✓</b>
		Total	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
Memory Usage		Used	✓	✓	<b>√</b>	<b>✓</b>	✓	✓	✓	<b>√</b>	✓	✓	<b>✓</b>
		Free		<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>						
	Power N	Power Management Enabled		×	<b>√</b>	×	×	<b>√</b>	✓	×	×	<b>√</b>	×
	Current Power Draw		×	×	<b>✓</b>	×	×	<b>√</b>	<b>√</b>	×	×	<b>√</b>	<b>✓</b>
	Pc	wer Draw Limit	×	×	<b>✓</b>	×	×	<b>√</b>	<b>√</b>	×	×	<b>√</b>	<b>✓</b>
	Graphics		<b>√</b>										
	Current Max	SM	<b>√</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>						
		Memory	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>✓</b>
		Graphics	<b>√</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>						
		SM	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>
		Memory	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>✓</b>
	GPU Compute		<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>
		PCIE Memory	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>
		Location-Based	✓	<b>√</b>	<b>✓</b>								
	Volatile	Total	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>	✓	<b>✓</b>	<b>✓</b>	<b>✓</b>	<b>✓</b>
ECC Errors		Location-Based	×	<b>√</b>	<b>√</b>	×	<b>√</b>	<b>√</b>	<b>√</b>	×	<b>√</b>	<b>√</b>	<b>✓</b>
	Aggregate	Total	✓	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>√</b>	✓	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>✓</b>
	Performance S	State	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>√</b>	✓	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>
		Process Id	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>✓</b>
		Process Name	<b>√</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>√</b>	<b>✓</b>	<b>✓</b>
		ted Device Memory	<b>√</b>	<b>√</b>	<b>✓</b>	<b>✓</b>	<b>√</b>	<b>√</b>	✓	<b>√</b>	<b>√</b>	<b>√</b>	<b>√</b>
C	Comma	nds	C2050	C2070	C2075	M2050	M2070	M2075	M2090	S2050	X2070	X2090	Gemini
	Set Compute N	Лode	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>√</b>
Set Pers	istence Mode	(Linux-Only)	✓	✓	<b>✓</b>	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>
Set Di	splay Model (V	Vin7-Only)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>√</b>
	Set ECC Mo	de	✓	✓	<b>√</b>	✓	✓	✓	✓	✓	✓	✓	<b>√</b>
	Clear ECC Err	ors	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>√</b>
Re	eset GPU (Linux	k-Only)	✓	✓	<b>√</b>	<b>√</b>	✓	<b>√</b>	✓	✓	✓	✓	<b>√</b>

Figure 1.2: This chart shows which features are available for each Fermi and Kepler architecture GPU product.

1.1 Feature Matrix 5

	Queries			5000	6000	Quadro Plex 7000	M2070-Q	600	2000	3000M	410	\$1070	C1060
Board Serial Number			✓	✓	✓	✓	✓	×	×	×	×	×	×
	GPU UUID		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VBios Version		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Inforom Version			✓	✓	✓	✓	✓	×	×	×	×	×	×
		ECC Object	×	✓	✓	✓	✓	×	×	×	×	×	×
	F	Power Object	<b>x</b> ✓	×	×	✓	×	×	×	×	×	×	×
	PCI Info			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Display Mod	le	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×
Persi	stence Mode (Li	nux-Only)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ECC Mode		Current	×	✓	✓	✓	✓	×	×	×	×	×	×
		Pending	×	✓	✓	✓	✓	×	×	×	×	×	×
		Current	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
(Win7-Only)		Pending	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Fan Speed		✓	✓	✓	✓	×	✓	✓	✓	✓	×	✓
	GPU Temperat	ture	✓	✓	✓	✓	×	✓	✓	✓	✓	<b>✓</b>	✓
		Total	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Used		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Free	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Power Management Enabled		×	×	×	✓	×	×	×	×	×	×	×
	ings Current Power Draw		×	×	×	✓	×	×	×	×	×	×	×
	Power Draw Limit		×	×	×	✓	×	×	×	×	×	×	×
		Graphics	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Current	SM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Clock Speeds		Memory	✓	✓	✓	✓	✓	✓	✓	✓	✓	<b>✓</b>	✓
		Graphics	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Max	SM	✓	✓	✓	✓	✓	$\checkmark$	<b>✓</b>		<b>✓</b>		
									•	<b>✓</b>	•	<b>✓</b>	<b>√</b>
Litilization Pates		Memory	✓	✓	✓	✓	✓	✓	<b>✓</b>	<b>✓</b>	<b>√</b>	✓	✓ ✓
Utilization Rates	0	Memory SPU Compute	✓ ✓	<b>✓</b>	<b>✓</b>	<b>✓</b>	✓ ✓	✓			✓ ✓	✓ ✓	✓ ✓
Utilization Rates									✓	<b>✓</b>	✓	<b>√</b>	✓
Utilization Rates	F	GPU Compute	<b>√</b>	✓	<b>√</b>	<b>√</b>	<b>✓</b>	✓	✓ ✓	✓ ✓ ✓	✓ ✓	✓ ✓	✓ ✓
		PU Compute	✓ ✓	<b>✓</b>	✓ ✓	✓ ✓	✓ ✓	✓ ✓	✓ ✓ ✓	✓ ✓	✓ ✓	✓ ✓	✓ ✓
Utilization Rates	Volatile	PCIE Memory  Location-Based	✓ ✓ ×	✓ ✓	✓ ✓	✓ ✓	✓ ✓ ✓	✓ ✓ ×	✓ ✓ ✓	✓ ✓ ✓	✓ ✓ ✓	✓ ✓ ✓ ✓ ×	✓ ✓ ✓ ×
	F	SPU Compute  **CIE Memory  **Location-Based  **Total	✓ ✓ × × × ×	✓ ✓ ✓ ✓ × ✓	✓ ✓ ✓ ✓ × ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓	× × ×		× × × ×	× × × ×	x x x x x	✓ ✓ ✓ × × × ×
	Volatile	PU Compute CIE Memory  Location-Based  Total  Location-Based  Total	× x x x x	✓ ✓ ✓ ✓ ✓ ×	✓ ✓ ✓ ✓ ×	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	× × × ×	✓ ✓ ✓ ✓ × × ×		× × ×	✓ ✓ ✓ × × ×	
	Volatile -	PU Compute CIE Memory  Location-Based  Total  Location-Based  Total	× × × × × ×		✓ ✓ ✓ ✓ ✓ × ✓ ✓ ✓ ✓	\( \sqrt{1} \)	\(      \)     \(      \)     \(      \)     \(      \)     \(      \)     \(      \)     \(      \)     \(      \)	✓ ✓ × × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓
	Volatile - Aggregate - Performance S	PU Compute  CIE Memory  Location-Based  Total  Location-Based  Total	√	√	✓ ✓ ✓ ✓ ✓ × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	\frac{1}{\sqrt{1}}	\(  \)	x x x x x x x x x x x x x x x x x x x	✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓	× × × × × · · · · · · · · · · · · · · ·	✓ ✓ ✓ ✓ × × × × × ✓ ✓ ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓ ✓	× × × × × × ×
ECC Errors	Volatile Aggregate Performance S	PU Compute  CIE Memory  Location-Based  Total  Location-Based  Total  tate  Process Id	× × × × × ×		✓ ✓ ✓ ✓ ✓ × ✓ ✓ ✓ ✓	\( \sqrt{1} \)	\(      \)     \(      \)     \(      \)     \(      \)     \(      \)     \(      \)     \(      \)     \(      \)	✓ ✓ × × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓	× × × × ×
ECC Errors  Process Info	Volatile Aggregate Performance S	CIE Memory  Location-Based  Total  Location-Based  Total  tate  Process Id  rocess Name  ed Device Memory	√	√	✓ ✓ ✓ ✓ ✓ × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	√	\(  \)	x x x x x x x x x x x x x x x x x x x	✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓	× × × × × · · · · · · · · · · · · · · ·	✓ ✓ ✓ ✓ × × × × × ✓ ✓ ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓ ✓	× × × × × · · · · · · · · · · · · · · ·
ECC Errors Process Info	Volatile Aggregate Performance S	CEPU Compute  CCIE Memory  Location-Based  Total  Location-Based  Total  tate  Process Id  Process Name ed Device Memory	✓ × × × × ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	√	\( \sqrt{1} \)	\( \sqrt{1} \)	x x x x	✓ ✓ ✓ × × × × × ✓ ✓ ✓ ✓ ✓	× × × × × × × × × × × × × × × × × × ×	x x x x	✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓	× × × × · ✓
ECC Errors Process Info	Volatile  Aggregate  Performance S  Allocat	CPU Compute  CCIE Memory  Location-Based  Total  Location-Based  Total  tate  Process Id  Process Name  ed Device Memory  Total  Location-Based  Total	✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	√	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓	✓ ✓ ✓ ✓ × × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
ECC Errors  Process Info	Volatile Aggregate Performance S Allocat Comman	PU Compute  CIE Memory  Location-Based  Total  Location-Based  Total  tate  Process Id  Process Name  ed Device Memory  Inds  Inde  (Linux-Only)	✓ × × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓	✓ ✓ ✓ ✓ × × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
ECC Errors  Process Info	Volatile  Aggregate  Performance S  Allocat  Comman  Set Compute M  sistence Mode	CEL Memory  Location-Based  Total  Location-Based  Total  tate  Process Id  Process Name ed Device Memory  Inds  Location-Based  Total  Location-Based  Total  Location-Based  Location-Based  Total  Location-Based  Location-Based  Total  Location-Based  L	✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	√	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	x x x x	✓ ✓ ✓ ✓ × × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
ECC Errors  Process Info	Volatile  Aggregate  Performance S  Allocat  Commail  Set Compute Model (Model	CEL Memory  Location-Based  Total  Location-Based  Total  tate  Process Id  rocess Name ed Device Memory  Location-Based  India	✓ × × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	√	√	✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓ × × × × ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

Figure 1.3: This chart shows which features are available for each Quadro and T10 GPU product.

6 NVML API Reference

# **Chapter 2**

# **Change log of NVML library**

This chapter list changes in API and bug fixes that were introduced to the library

# 2.1 Changes between NVML v1.0 and v2.285

- Added possibility to query separately current and pending driver model with nvmlDeviceGetDriverModel
- Added API nvmlDeviceGetVbiosVersion function to report VBIOS version.
- Added pciSubSystemId to nvmlPciInfo\_t struct
- Added API nvmlErrorString function to convert error code to string
- Updated docs to indicate we support M2075 and C2075
- Added API nvmlSystemGetHicVersion function to report HIC firmware version
- Added NVML versioning support
  - Functions that changed API and/or size of structs have appended versioning suffix (e.g. nvmlDeviceGetPciInfo\_v2). Appropriate C defines have been added that map old function names to the newer version of the function
- Added support for concurrent library usage by multiple libraries
- Added API nvmlDeviceGetMaxClockInfo function for reporting device's clock limits
- Added new error code NVML\_ERROR\_DRIVER\_NOT\_LOADED used by nvmlInit
- Extended nvmlPciInfo\_t struct with new field: sub system id
- Added NVML support on Windows guest account
- Changed format of pciBusId string (to XXXX:XX:XX.X) of nvmlPciInfo\_t
- Parsing of busId in nvmlDeviceGetHandleByPciBusId is less restrictive. You can pass 0:2:0.0 or 0000:02:00
  and other variations
- Added API for events waiting for GPU events (Linux only) see docs of Event Handling Methods
- Added API nvmlDeviceGetComputeRunningProcesses and nvmlSystemGetProcessName functions for looking up currently running compute applications
- Deprecated nvmlDeviceGetPowerState in favor of nvmlDeviceGetPerformanceState.

# 2.2 Changes between NVML v2.285 and v3.295

- deprecated nvmlDeviceGetHandleBySerial in favor of newly added nvmlDeviceGetHandleByUUID
- Marked the input parameters of nvmlDeviceGetHandleBySerial, nvmlDeviceGetHandleByUUID and nvmlDeviceGetHandleByPciBusId as const
- Added nvmlDeviceOnSameBoard
- Added Constants defines
- Added nvmlDeviceGetMaxPcieLinkGeneration, nvmlDeviceGetMaxPcieLinkWidth, nvmlDeviceGetCurrPcieLinkWidth

- nvmlDeviceGetDetailedEccErrors will report zero for unsupported ECC error counters when a subset of ECC error counters are supported

# Chapter 3

# **Deprecated List**

12 Deprecated List

**Global nvmlDeviceGetHandleBySerial** Since more than one GPU can exist on a single board this function is deprecated in favor of nvmlDeviceGetHandleByUUID. For dual GPU boards this function will return NVML\_ERROR\_INVALID\_ARGUMENT.

# **Chapter 4**

# **Module Index**

# 4.1 Modules

# Here is a list of all modules:

evice Structs	17
evice Enums	18
nit Structs	23
itialization and Cleanup	25
ror reporting	26
onstants	27
vstem Queries	28
nit Queries	30
evice Queries	34
nit Commands	53
evice Commands	54
vent Handling Methods	58
Event Types	24

14 Module Index

# **Chapter 5**

# **Data Structure Index**

# **5.1** Data Structures

Here are	the d	lata struct	ures with	brief	descrip	tions
----------	-------	-------------	-----------	-------	---------	-------

nvmlEccErrorCounts_t	63
nvmlEventData_t	64
nvmlHwbcEntry_t	65
nvmlLedState_t	66
nvmlMemory_t	67
nvmlPciInfo_t	68
nvmlProcessInfo_t	69
nvmlPSUInfo_t	70
nvmlUnitFanInfo_t	71
nvmlUnitFanSpeeds_t	
nvmlUnitInfo_t	73
nymlUtilization t	74

16 Data Structure Index

# Chapter 6

# **Module Documentation**

# **6.1** Device Structs

# **Data Structures**

- struct nvmlPciInfo\_t
- struct nvmlEccErrorCounts\_t
- struct nvmlUtilization\_t
- struct nvmlMemory\_t
- struct nvmlProcessInfo\_t

# **Defines**

• #define NVML\_VALUE\_NOT\_AVAILABLE (-1)

# **6.1.1** Define Documentation

# 6.1.1.1 #define NVML\_VALUE\_NOT\_AVAILABLE (-1)

Special constant that some fields take when they are not available. Used when only part of the struct is not available. Each structure explicitly states when to check for this value.

18 Module Documentation

# **6.2** Device Enums

# **Defines**

• #define nvmlFlagDefault 0x00

Generic flag used to specify the default behavior of some functions. See description of particular functions for details.

• #define nvmlFlagForce 0x01

Generic flag used to force some behavior. See description of particular functions for details.

# **Enumerations**

```
enum nvmlEnableState_t {
 NVML_FEATURE_DISABLED = 0,
 NVML FEATURE ENABLED = 1 }
• enum nvmlTemperatureSensors_t { NVML_TEMPERATURE_GPU = 0 }
enum nvmlComputeMode_t {
 NVML COMPUTEMODE DEFAULT = 0,
 NVML COMPUTEMODE EXCLUSIVE THREAD = 1,
 NVML_COMPUTEMODE_PROHIBITED = 2,
 NVML_COMPUTEMODE_EXCLUSIVE_PROCESS = 3 }
enum nvmlEccBitType_t {
 NVML SINGLE BIT ECC = 0,
 NVML DOUBLE BIT ECC = 1 }
enum nvmlEccCounterType_t {
 NVML VOLATILE ECC = 0,
 NVML AGGREGATE ECC = 1 }
enum nvmlClockType_t {
 NVML\_CLOCK\_GRAPHICS = 0,
 NVML CLOCK SM = 1,
 NVML CLOCK MEM = 2 }
enum nvmlDriverModel_t {
 NVML_DRIVER_WDDM = 0,
 NVML DRIVER WDM = 1 }
enum nvmlPstates_t {
 NVML_PSTATE_0 = 0,
 NVML_PSTATE_1 = 1,
 NVML_PSTATE_2 = 2,
 NVML_PSTATE_3 = 3,
 NVML_PSTATE_4 = 4,
 NVML PSTATE 5 = 5,
 NVML PSTATE 6 = 6,
 NVML_PSTATE_7 = 7,
```

6.2 Device Enums

```
NVML_PSTATE_8 = 8,
 NVML_PSTATE_9 = 9,
 NVML_PSTATE_{10} = 10,
 NVML_PSTATE_11 = 11,
 NVML_PSTATE_{12} = 12,
 NVML_PSTATE_{13} = 13,
 NVML_PSTATE_14 = 14,
 NVML_PSTATE_15 = 15,
 NVML_PSTATE_UNKNOWN = 32 }
enum nvmlInforomObject_t {
 NVML_INFOROM_OEM = 0,
 NVML_INFOROM_ECC = 1,
 NVML_INFOROM_POWER = 2 }
enum nvmlReturn_t {
 NVML_SUCCESS = 0,
 NVML_ERROR_UNINITIALIZED = 1,
 NVML\_ERROR\_INVALID\_ARGUMENT = 2,
 NVML\_ERROR\_NOT\_SUPPORTED = 3,
 NVML_ERROR_NO_PERMISSION = 4,
 NVML_ERROR_ALREADY_INITIALIZED = 5,
 NVML\_ERROR\_NOT\_FOUND = 6,
 NVML_ERROR_INSUFFICIENT_SIZE = 7,
 NVML_ERROR_INSUFFICIENT_POWER = 8,
 NVML_ERROR_DRIVER_NOT_LOADED = 9,
 NVML\_ERROR\_TIMEOUT = 10,
 NVML_ERROR_UNKNOWN = 999 }
```

# **6.2.1** Enumeration Type Documentation

# 6.2.1.1 enum nvmlClockType\_t

Clock types.

All speeds are in Mhz.

# **Enumerator:**

```
NVML_CLOCK_GRAPHICS Graphics clock domain.

NVML_CLOCK_SM SM clock domain.

NVML_CLOCK_MEM Memory clock domain.
```

20 Module Documentation

# 6.2.1.2 enum nvmlComputeMode\_t

Compute mode.

NVML\_COMPUTEMODE\_EXCLUSIVE\_PROCESS was added in CUDA 4.0. Earlier CUDA versions supported a single exclusive mode, which is equivalent to NVML\_COMPUTEMODE\_EXCLUSIVE\_THREAD in CUDA 4.0 and beyond.

#### **Enumerator:**

NVML\_COMPUTEMODE\_DEFAULT Default compute mode – multiple contexts per device.

**NVML\_COMPUTEMODE\_EXCLUSIVE\_THREAD** Compute-exclusive-thread mode – only one context per device, usable from one thread at a time.

NVML\_COMPUTEMODE\_PROHIBITED Compute-prohibited mode – no contexts per device.

**NVML\_COMPUTEMODE\_EXCLUSIVE\_PROCESS** Compute-exclusive-process mode – only one context per device, usable from multiple threads at a time.

# 6.2.1.3 enum nvmlDriverModel t

Driver models.

Windows only.

#### **Enumerator:**

**NVML\_DRIVER\_WDDM** WDDM driver model – GPU treated as a display device.

NVML\_DRIVER\_WDM WDM (TCC) model (recommended) – GPU treated as a generic device.

# 6.2.1.4 enum nvmlEccBitType\_t

ECC bit types.

#### **Enumerator:**

**NVML\_SINGLE\_BIT\_ECC** Single bit ECC errors.

NVML DOUBLE BIT ECC Double bit ECC errors.

# 6.2.1.5 enum nvmlEccCounterType\_t

ECC counter types.

Note: Volatile counts are reset each time the driver loads. On Windows this is once per boot. On Linux this can be more frequent. On Linux the driver unloads when no active clients exist. If persistence mode is enabled or there is always a driver client active (e.g. X11), then Linux also sees per-boot behavior. If not, volatile counts are reset each time a compute app is run.

# **Enumerator:**

**NVML\_VOLATILE\_ECC** Volatile counts are reset each time the driver loads.

**NVML\_AGGREGATE\_ECC** Aggregate counts persist across reboots (i.e. for the lifetime of the device).

6.2 Device Enums 21

#### 6.2.1.6 enum nvmlEnableState t

Generic enable/disable enum.

#### **Enumerator:**

```
NVML_FEATURE_DISABLED Feature disabled.NVML_FEATURE_ENABLED Feature enabled.
```

# 6.2.1.7 enum nvmlInforomObject\_t

Available infoROM objects.

# **Enumerator:**

```
NVML_INFOROM_OEM An object defined by OEM.NVML_INFOROM_ECC The ECC object determining the level of ECC support.NVML_INFOROM_POWER The power management object.
```

# 6.2.1.8 enum nvmlPstates\_t

Allowed PStates.

#### **Enumerator:**

```
NVML PSTATE 0 Performance state 0 – Maximum Performance.
NVML PSTATE 1 Performance state 1.
NVML_PSTATE_2 Performance state 2.
NVML_PSTATE_3 Performance state 3.
NVML PSTATE 4 Performance state 4.
NVML_PSTATE_5 Performance state 5.
NVML_PSTATE_6 Performance state 6.
NVML_PSTATE_7 Performance state 7.
NVML_PSTATE_8 Performance state 8.
NVML_PSTATE_9 Performance state 9.
NVML_PSTATE_10 Performance state 10.
NVML_PSTATE_11 Performance state 11.
NVML_PSTATE_12 Performance state 12.
NVML_PSTATE_13 Performance state 13.
NVML_PSTATE_14 Performance state 14.
NVML_PSTATE_15 Performance state 15 – Minimum Performance.
NVML_PSTATE_UNKNOWN Unknown performance state.
```

22 Module Documentation

#### 6.2.1.9 enum nvmlReturn t

Return values for NVML API calls.

#### **Enumerator:**

**NVML\_SUCCESS** The operation was successful.

NVML\_ERROR\_UNINITIALIZED NVML was not first initialized with nvmlInit().

**NVML\_ERROR\_INVALID\_ARGUMENT** A supplied argument is invalid.

**NVML\_ERROR\_NOT\_SUPPORTED** The requested operation is not available on target device.

**NVML\_ERROR\_NO\_PERMISSION** The current user does not have permission for operation.

**NVML\_ERROR\_ALREADY\_INITIALIZED** Deprecated: Multiple initializations are now allowed through ref counting.

**NVML\_ERROR\_NOT\_FOUND** A query to find an object was unsuccessful.

**NVML\_ERROR\_INSUFFICIENT\_SIZE** An input argument is not large enough.

**NVML\_ERROR\_INSUFFICIENT\_POWER** A device's external power cables are not properly attached.

NVML ERROR DRIVER NOT LOADED NVIDIA driver is not loaded.

NVML\_ERROR\_TIMEOUT User provided timeout passed.

NVML\_ERROR\_UNKNOWN An internal driver error occurred.

# 6.2.1.10 enum nvmlTemperatureSensors\_t

Temperature sensors.

# **Enumerator:**

NVML\_TEMPERATURE\_GPU Temperature sensor for the GPU die.

6.3 Unit Structs 23

# **6.3** Unit Structs

# **Data Structures**

```
• struct nvmlHwbcEntry_t
```

- struct nvmlLedState\_t
- struct nvmlUnitInfo\_t
- struct nvmlPSUInfo\_t
- struct nvmlUnitFanInfo\_t
- struct nvmlUnitFanSpeeds\_t

# **Enumerations**

```
    enum nvmlFanState_t {
        NVML_FAN_NORMAL = 0,
        NVML_FAN_FAILED = 1 }
    enum nvmlLedColor_t {
        NVML_LED_COLOR_GREEN = 0,
        NVML_LED_COLOR_AMBER = 1 }
```

# **6.3.1** Enumeration Type Documentation

# 6.3.1.1 enum nvmlFanState\_t

Fan state enum.

# **Enumerator:**

```
NVML_FAN_NORMAL Fan is working properly. NVML_FAN_FAILED Fan has failed.
```

# 6.3.1.2 enum nvmlLedColor\_t

Led color enum.

#### **Enumerator:**

```
NVML_LED_COLOR_GREEN GREEN, indicates good health. NVML_LED_COLOR_AMBER AMBER, indicates problem.
```

24 Module Documentation

# **6.4** Event Types

# **Defines**

#define nvmlEventTypeSingleBitEccError 0x000000000000000001LL
 Event about single bit ECC errors.

• #define nvmlEventTypeDoubleBitEccError 0x000000000000000002LL

Event about double bit ECC errors.

• #define nvmlEventTypePState 0x00000000000000004LL

Event about PState changes.

• #define nvmlEventTypeXidCriticalError 0x00000000000000000LL

Event that Xid critical error occurred.

• #define nvmlEventTypeNone 0x000000000000000LL

Mask with no events.

• #define nvmlEventTypeAll

Mask of all events.

# 6.4.1 Detailed Description

Event Types which user can be notified about. See description of particular functions for details.

See nvmlDeviceRegisterEvents and nvmlDeviceGetSupportedEventTypes to check which devices support each event.

Types can be combined with bitwise or operator '|' when passed to nvmlDeviceRegisterEvents

# **6.4.2** Define Documentation

# 6.4.2.1 #define nvmlEventTypePState 0x00000000000000004LL

#### Note:

On Fermi architecture PState changes are also an indicator that GPU is throttling down due to no work being executed on the GPU, power capping or thermal capping. In a typical situation, Fermi-based GPU should stay in P0 for the duration of the execution of the compute process.

# 6.5 Initialization and Cleanup

#### **Functions**

- nvmlReturn\_t DECLDIR nvmlInit (void)
- nvmlReturn\_t DECLDIR nvmlShutdown (void)

# 6.5.1 Detailed Description

This chapter describes the methods that handle NVML initialization and cleanup. It is the user's responsibility to call <a href="nvmlInit">nvmlInit</a>() before calling any other methods, and <a href="nvmlShutdown">nvmlShutdown</a>() once NVML is no longer being used.

# **6.5.2** Function Documentation

# 6.5.2.1 nvmlReturn\_t DECLDIR nvmlInit (void)

Initialize NVML by discovering and attaching to all GPU devices in the system.

For all products.

This method should be called once before invoking any other methods in the library. A reference count of the number of initializations is maintained. Shutdown only occurs when the reference count reaches zero.

#### **Returns:**

- NVML\_SUCCESS if NVML has been properly initialized
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to talk to any device
- NVML\_ERROR\_DRIVER\_NOT\_LOADED if NVIDIA driver is not running
- NVML ERROR INSUFFICIENT POWER if any devices have improperly attached external power cables
- NVML\_ERROR\_UNKNOWN on any unexpected error

# 6.5.2.2 nvmlReturn\_t DECLDIR nvmlShutdown (void)

Shut down NVML by releasing all GPU resources previously allocated with nvmlInit().

For all products.

This method should be called after NVML work is done, once for each call to nvmlInit() A reference count of the number of initializations is maintained. Shutdown only occurs when the reference count reaches zero. For backwards compatibility, no error is reported if nvmlShutdown() is called more times than nvmlInit().

#### **Returns:**

- NVML\_SUCCESS if NVML has been properly shut down
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_UNKNOWN on any unexpected error

26 Module Documentation

# 6.6 Error reporting

# **Functions**

• const DECLDIR char \* nvmlErrorString (nvmlReturn\_t result)

# 6.6.1 Detailed Description

This chapter describes helper functions for error reporting routines.

# **6.6.2** Function Documentation

# 6.6.2.1 const DECLDIR char\* nvmlErrorString (nvmlReturn\_t result)

Helper method for converting NVML error codes into readable strings.

For all products

# **Parameters:**

result NVML error code to convert

# **Returns:**

String representation of the error.

6.7 Constants 27

# 6.7 Constants

# **Defines**

- #define NVML\_DEVICE\_INFOROM\_VERSION\_BUFFER\_SIZE 16
- #define NVML\_DEVICE\_UUID\_BUFFER\_SIZE 80
- #define NVML\_SYSTEM\_DRIVER\_VERSION\_BUFFER\_SIZE 80
- #define NVML\_SYSTEM\_NVML\_VERSION\_BUFFER\_SIZE 80
- #define NVML\_DEVICE\_NAME\_BUFFER\_SIZE 64
- #define NVML\_DEVICE\_SERIAL\_BUFFER\_SIZE 30
- #define NVML\_DEVICE\_VBIOS\_VERSION\_BUFFER\_SIZE 32

# **6.7.1** Define Documentation

# 6.7.1.1 #define NVML\_DEVICE\_INFOROM\_VERSION\_BUFFER\_SIZE 16

Buffer size guaranteed to be large enough for nvmlDeviceGetInforomVersion

# 6.7.1.2 #define NVML\_DEVICE\_NAME\_BUFFER\_SIZE 64

Buffer size guaranteed to be large enough for nvmlDeviceGetName

# 6.7.1.3 #define NVML DEVICE SERIAL BUFFER SIZE 30

Buffer size guaranteed to be large enough for nvmlDeviceGetSerial

# 6.7.1.4 #define NVML\_DEVICE\_UUID\_BUFFER\_SIZE 80

Buffer size guaranteed to be large enough for nvmlDeviceGetUUID

# 6.7.1.5 #define NVML\_DEVICE\_VBIOS\_VERSION\_BUFFER\_SIZE 32

Buffer size guaranteed to be large enough for nvmlDeviceGetVbiosVersion

# 6.7.1.6 #define NVML\_SYSTEM\_DRIVER\_VERSION\_BUFFER\_SIZE 80

Buffer size guaranteed to be large enough for nvmlSystemGetDriverVersion

# 6.7.1.7 #define NVML\_SYSTEM\_NVML\_VERSION\_BUFFER\_SIZE 80

Buffer size guaranteed to be large enough for nvmlSystemGetNVMLVersion

28 Module Documentation

# 6.8 System Queries

#### **Functions**

- nvmlReturn\_t DECLDIR nvmlSystemGetDriverVersion (char \*version, unsigned int length)
- nvmlReturn\_t DECLDIR nvmlSystemGetNVMLVersion (char \*version, unsigned int length)
- nvmlReturn\_t DECLDIR nvmlSystemGetProcessName (unsigned int pid, char \*name, unsigned int length)

# **6.8.1 Detailed Description**

This chapter describes the queries that NVML can perform against the local system. These queries are not device-specific.

# **6.8.2** Function Documentation

# 6.8.2.1 nvmlReturn\_t DECLDIR nvmlSystemGetDriverVersion (char \* version, unsigned int length)

Retrieves the version of the system's graphics driver.

For all products.

The version identifier is an alphanumeric string. It will not exceed 80 characters in length (including the NULL terminator). See <a href="https://nwww.nvmlconstants::NVML\_SYSTEM\_DRIVER\_VERSION\_BUFFER\_SIZE">nvmlconstants::NVML\_SYSTEM\_DRIVER\_VERSION\_BUFFER\_SIZE</a>.

#### **Parameters:**

version Reference in which to return the version identifier

length The maximum allowed length of the string returned in version

# **Returns:**

- NVML\_SUCCESS if version has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if version is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if length is too small

# 6.8.2.2 nvmlReturn\_t DECLDIR nvmlSystemGetNVMLVersion (char \* version, unsigned int length)

Retrieves the version of the NVML library.

For all products.

#### **Parameters:**

version Reference in which to return the version identifier

length The maximum allowed length of the string returned in version

# Returns:

• NVML\_SUCCESS if version has been set

6.8 System Queries 29

- NVML\_ERROR\_INVALID\_ARGUMENT if version is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if *length* is too small

## 6.8.2.3 nvmlReturn\_t DECLDIR nvmlSystemGetProcessName (unsigned int pid, char \* name, unsigned int length)

Gets name of the process with provided process id

For all products.

Returned process name is cropped to provided length. name string is encoded in ANSI.

#### **Parameters:**

pid The identifier of the processname Reference in which to return the process namelength The maximum allowed length of the string returned in name

- NVML\_SUCCESS if name has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if name is NULL
- NVML\_ERROR\_NOT\_FOUND if process doesn't exists
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

### 6.9 Unit Queries

#### **Functions**

- nvmlReturn\_t DECLDIR nvmlUnitGetCount (unsigned int \*unitCount)
- nvmlReturn\_t DECLDIR nvmlUnitGetHandleByIndex (unsigned int index, nvmlUnit\_t \*unit)
- nvmlReturn\_t DECLDIR nvmlUnitGetUnitInfo (nvmlUnit\_t unit, nvmlUnitInfo\_t \*info)
- nvmlReturn\_t DECLDIR nvmlUnitGetLedState (nvmlUnit\_t unit, nvmlLedState\_t \*state)
- nvmlReturn\_t DECLDIR nvmlUnitGetPsuInfo (nvmlUnit\_t unit, nvmlPSUInfo\_t \*psu)
- nvmlReturn\_t DECLDIR nvmlUnitGetTemperature (nvmlUnit\_t unit, unsigned int type, unsigned int \*temp)
- nvmlReturn\_t DECLDIR nvmlUnitGetFanSpeedInfo (nvmlUnit\_t unit, nvmlUnitFanSpeeds\_t \*fanSpeeds)
- nvmlReturn\_t DECLDIR nvmlUnitGetDevices (nvmlUnit\_t unit, unsigned int \*deviceCount, nvmlDevice\_t \*devices)
- nvmlReturn\_t DECLDIR nvmlSystemGetHicVersion (unsigned int \*hwbcCount, nvmlHwbcEntry\_t \*hwbcEntries)

### 6.9.1 Detailed Description

This chapter describes that queries that NVML can perform against each unit. For S-class systems only. In each case the device is identified with an nvmlUnit\_t handle. This handle is obtained by calling nvmlUnitGetHandleByIndex().

#### **6.9.2** Function Documentation

## 6.9.2.1 nvmlReturn\_t DECLDIR nvmlSystemGetHicVersion (unsigned int \* hwbcCount, nvmlHwbcEntry\_t \* hwbcEntries)

Retrieves the IDs and firmware versions for any Host Interface Cards (HICs) in the system.

For S-class products.

The *hwbcCount* argument is expected to be set to the size of the input *hwbcEntries* array. The HIC must be connected to an S-class system for it to be reported by this function.

#### **Parameters:**

hwbcCount Size of hwbcEntries array

hwbcEntries Array holding information about hwbc

#### **Returns:**

- NVML\_SUCCESS if hwbcCount and hwbcEntries have been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if either hwbcCount or hwbcEntries is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if hwbcCount indicates that the hwbcEntries array is too small

### 6.9.2.2 nvmlReturn\_t DECLDIR nvmlUnitGetCount (unsigned int \* unitCount)

Retrieves the number of units in the system.

For S-class products.

6.9 Unit Queries 31

#### **Parameters:**

unitCount Reference in which to return the number of units

#### **Returns:**

- NVML SUCCESS if unitCount has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unitCount is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.9.2.3 nvmlReturn\_t DECLDIR nvmlUnitGetDevices (nvmlUnit\_t *unit*, unsigned int \* *deviceCount*, nvmlDevice\_t \* *devices*)

Retrieves the set of GPU devices that are attached to the specified unit.

For S-class products.

The *deviceCount* argument is expected to be set to the size of the input *devices* array.

#### **Parameters:**

unit The identifier of the target unit

deviceCount Reference in which to provide the devices array size, and to return the number of attached GPU devices

devices Reference in which to return the references to the attached GPU devices

#### **Returns:**

- NVML SUCCESS if deviceCount and devices have been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INSUFFICIENT\_SIZE if deviceCount indicates that the devices array is too small
- NVML ERROR INVALID ARGUMENT if unit is invalid, either of deviceCount or devices is NULL

## 6.9.2.4 nvmlReturn\_t DECLDIR nvmlUnitGetFanSpeedInfo (nvmlUnit\_t unit, nvmlUnitFanSpeeds\_t \* fanSpeeds)

Retrieves the fan speed readings for the unit.

For S-class products.

See nvmlUnitFanSpeeds\_t for details on available fan speed info.

#### **Parameters:**

unit The identifier of the target unit

fanSpeeds Reference in which to return the fan speed information

- NVML\_SUCCESS if fanSpeeds has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unit is invalid or fanSpeeds is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if this is not an S-class product
- NVML\_ERROR\_UNKNOWN on any unexpected error

### 6.9.2.5 nvmlReturn\_t DECLDIR nvmlUnitGetHandleByIndex (unsigned int index, nvmlUnit\_t \* unit)

Acquire the handle for a particular unit, based on its index.

For S-class products.

Valid indices are derived from the *unitCount* returned by nvmlUnitGetCount(). For example, if *unitCount* is 2 the valid indices are 0 and 1, corresponding to UNIT 0 and UNIT 1.

The order in which NVML enumerates units has no guarantees of consistency between reboots.

#### **Parameters:**

```
index The index of the target unit, >= 0 and < unitCount</li>unit Reference in which to return the unit handle
```

#### **Returns:**

- NVML SUCCESS if unit has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if index is invalid or unit is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### 6.9.2.6 nvmlReturn\_t DECLDIR nvmlUnitGetLedState (nvmlUnit\_t unit, nvmlLedState\_t \* state)

Retrieves the LED state associated with this unit.

For S-class products.

See <a href="nvmlLedState\_t">nvmlLedState\_t</a> for details on allowed states.

#### **Parameters:**

```
unit The identifier of the target unitstate Reference in which to return the current LED state
```

#### **Returns:**

- NVML\_SUCCESS if state has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unit is invalid or state is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if this is not an S-class product
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlUnitSetLedState()

#### 6.9.2.7 nvmlReturn\_t DECLDIR nvmlUnitGetPsuInfo (nvmlUnit\_t unit, nvmlPSUInfo\_t \* psu)

Retrieves the PSU stats for the unit.

For S-class products.

See <a href="nvmlPSUInfo\_t">nvmlPSUInfo\_t</a> for details on available PSU info.

6.9 Unit Queries 33

#### **Parameters:**

```
unit The identifier of the target unitpsu Reference in which to return the PSU information
```

#### **Returns:**

- NVML\_SUCCESS if *psu* has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unit is invalid or psu is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if this is not an S-class product
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.9.2.8 nvmlReturn\_t DECLDIR nvmlUnitGetTemperature (nvmlUnit\_t *unit*, unsigned int *type*, unsigned int \* *temp*)

Retrieves the temperature readings for the unit, in degrees C.

For S-class products.

Depending on the product, readings may be available for intake (type=0), exhaust (type=1) and board (type=2).

#### **Parameters:**

```
unit The identifier of the target unittype The type of reading to taketemp Reference in which to return the intake temperature
```

#### **Returns:**

- NVML\_SUCCESS if temp has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unit or type is invalid or temp is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if this is not an S-class product
- NVML\_ERROR\_UNKNOWN on any unexpected error

### 6.9.2.9 nvmlReturn\_t DECLDIR nvmlUnitGetUnitInfo (nvmlUnit\_t unit, nvmlUnitInfo\_t \* info)

Retrieves the static information associated with a unit.

For S-class products.

See <a href="mailto:nvmlUnitInfo\_t">nvmlUnitInfo\_t</a> for details on available unit info.

#### **Parameters:**

```
unit The identifier of the target unitinfo Reference in which to return the unit information
```

- NVML SUCCESS if info has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unit is invalid or info is NULL

### **6.10** Device Queries

#### **Functions**

- nvmlReturn\_t DECLDIR nvmlDeviceGetCount (unsigned int \*deviceCount)
- nvmlReturn\_t DECLDIR nvmlDeviceGetHandleByIndex (unsigned int index, nvmlDevice\_t \*device)
- nvmlReturn\_t DECLDIR nvmlDeviceGetHandleBySerial (const char \*serial, nvmlDevice\_t \*device)
- nvmlReturn t DECLDIR nvmlDeviceGetHandleByUUID (const char \*uuid, nvmlDevice t \*device)
- nvmlReturn\_t DECLDIR nvmlDeviceGetHandleByPciBusId (const char \*pciBusId, nvmlDevice\_t \*device)
- nvmlReturn\_t DECLDIR nvmlDeviceGetName (nvmlDevice\_t device, char \*name, unsigned int length)
- nvmlReturn\_t DECLDIR nvmlDeviceGetSerial (nvmlDevice\_t device, char \*serial, unsigned int length)
- nvmlReturn\_t DECLDIR nvmlDeviceGetUUID (nvmlDevice\_t device, char \*uuid, unsigned int length)
- nvmlReturn\_t DECLDIR nvmlDeviceGetInforomVersion (nvmlDevice\_t device, nvmlInforomObject\_t object, char \*version, unsigned int length)
- nvmlReturn\_t DECLDIR nvmlDeviceGetDisplayMode (nvmlDevice\_t device, nvmlEnableState\_t \*display)
- nvmlReturn\_t DECLDIR nvmlDeviceGetPersistenceMode (nvmlDevice\_t device, nvmlEnableState\_t \*mode)
- nvmlReturn\_t DECLDIR nvmlDeviceGetPciInfo (nvmlDevice\_t device, nvmlPciInfo\_t \*pci)
- nvmlReturn\_t DECLDIR nvmlDeviceGetMaxPcieLinkGeneration (nvmlDevice\_t device, unsigned int \*maxLinkGen)
- nvmlReturn\_t DECLDIR nvmlDeviceGetMaxPcieLinkWidth (nvmlDevice\_t device, unsigned int \*maxLinkWidth)
- nvmlReturn\_t DECLDIR nvmlDeviceGetCurrPcieLinkGeneration (nvmlDevice\_t device, unsigned int \*currLinkGen)
- nvmlReturn\_t DECLDIR nvmlDeviceGetCurrPcieLinkWidth (nvmlDevice\_t device, unsigned int \*currLinkWidth)
- nvmlReturn\_t DECLDIR nvmlDeviceGetClockInfo (nvmlDevice\_t device, nvmlClockType\_t type, unsigned int \*clock)
- nvmlReturn\_t DECLDIR nvmlDeviceGetMaxClockInfo (nvmlDevice\_t device, nvmlClockType\_t type, unsigned int \*clock)
- nvmlReturn\_t DECLDIR nvmlDeviceGetFanSpeed (nvmlDevice\_t device, unsigned int \*speed)
- nvmlReturn\_t DECLDIR nvmlDeviceGetTemperature (nvmlDevice\_t device, nvmlTemperatureSensors\_t sensorType, unsigned int \*temp)
- nvmlReturn t DECLDIR nvmlDeviceGetPerformanceState (nvmlDevice t device, nvmlPstates t \*pState)
- nvmlReturn t DECLDIR nvmlDeviceGetPowerState (nvmlDevice t device, nvmlPstates t \*pState)
- nvmlReturn\_t DECLDIR nvmlDeviceGetPowerManagementMode (nvmlDevice\_t device, nvmlEnableState\_t \*mode)
- nvmlReturn t DECLDIR nvmlDeviceGetPowerManagementLimit (nvmlDevice t device, unsigned int \*limit)
- nvmlReturn t DECLDIR nvmlDeviceGetPowerUsage (nvmlDevice t device, unsigned int \*power)
- nvmlReturn\_t DECLDIR nvmlDeviceGetMemoryInfo (nvmlDevice\_t device, nvmlMemory\_t \*memory)
- nvmlReturn\_t DECLDIR nvmlDeviceGetComputeMode (nvmlDevice\_t device, nvmlComputeMode\_t \*mode)
- nvmlReturn\_t DECLDIR nvmlDeviceGetEccMode (nvmlDevice\_t device, nvmlEnableState\_t \*current, nvmlEnableState\_t \*pending)
- nvmlReturn\_t DECLDIR nvmlDeviceGetTotalEccErrors (nvmlDevice\_t device, nvmlEccBitType\_t bitType, nvmlEccCounterType\_t counterType, unsigned long long \*eccCounts)
- nvmlReturn\_t DECLDIR nvmlDeviceGetDetailedEccErrors (nvmlDevice\_t device, nvmlEccBitType\_t bitType, nvmlEccCounterType\_t counterType, nvmlEccErrorCounts\_t \*eccCounts)
- nvmlReturn\_t DECLDIR nvmlDeviceGetUtilizationRates (nvmlDevice\_t device, nvmlUtilization\_t \*utilization)
- nvmlReturn\_t DECLDIR nvmlDeviceGetDriverModel (nvmlDevice\_t device, nvmlDriverModel\_t \*current, nvmlDriverModel\_t \*pending)
- nvmlReturn\_t DECLDIR nvmlDeviceGetVbiosVersion (nvmlDevice\_t device, char \*version, unsigned int length)

• nvmlReturn\_t DECLDIR nvmlDeviceGetComputeRunningProcesses (nvmlDevice\_t device, unsigned int \*infoCount, nvmlProcessInfo\_t \*infos)

nvmlReturn\_t DECLDIR nvmlDeviceOnSameBoard (nvmlDevice\_t device1, nvmlDevice\_t device2, int \*onSameBoard)

### 6.10.1 Detailed Description

This chapter describes that queries that NVML can perform against each device. In each case the device is identified with an nvmlDevice\_t handle. This handle is obtained by calling one of nvmlDeviceGetHandleByIndex(), nvmlDeviceGetHandleBySerial() or nvmlDeviceGetHandleByPciBusId().

### **6.10.2** Function Documentation

## 6.10.2.1 nvmlReturn\_t DECLDIR nvmlDeviceGetClockInfo (nvmlDevice\_t device, nvmlClockType\_t type, unsigned int \* clock)

Retrieves the current clock speeds for the device.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

See nvmlClockType\_t for details on available clock information.

#### **Parameters:**

```
device The identifier of the target devicetype Identify which clock domain to queryclock Reference in which to return the clock speed in MHz
```

#### **Returns:**

- NVML SUCCESS if *clock* has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if device is invalid or clock is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device cannot report the specified clock
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.2 nvmlReturn\_t DECLDIR nvmlDeviceGetComputeMode (nvmlDevice\_t device, nvmlComputeMode t \* mode)

Retrieves the current compute mode for the device.

For all CUDA-capable products.

See <a href="nvmlComputeMode\_t">nvmlComputeMode\_t</a> for details on allowed compute modes.

#### **Parameters:**

```
device The identifier of the target devicemode Reference in which to return the current compute mode
```

### **Returns:**

• NVML\_SUCCESS if mode has been set

- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or mode is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceSetComputeMode()

## 6.10.2.3 nvmlReturn\_t DECLDIR nvmlDeviceGetComputeRunningProcesses (nvmlDevice\_t device, unsigned int \* infoCount, nvmlProcessInfo\_t \* infos)

Get information about processes with a compute context on a device

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families.

This function returns information only about compute running processes (e.g. CUDA application which have active context). Any graphics applications (e.g. using OpenGL, DirectX) won't be listed by this function.

To query the current number of running compute processes, call this function with \*infoCount = 0. The return code will be NVML\_ERROR\_INSUFFICIENT\_SIZE, or NVML\_SUCCESS if none are running. For this call *infos* is allowed to be NULL.

Keep in mind that information returned by this call is dynamic and the number of elements might change in time. Allocate more space for *infos* table in case new compute processes are spawned.

#### **Parameters:**

device The identifier of the target device

*infoCount* Reference in which to provide the *infos* array size, and to return the number of returned elements *infos* Reference in which to return the process information

### **Returns:**

- NVML\_SUCCESS if infoCount and infos have been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INSUFFICIENT\_SIZE if *infoCount* indicates that the *infos* array is too small *infoCount* will contain minimal amount of space necessary for the call to complete
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid, either of infoCount or infos is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlSystemGetProcessName

### 6.10.2.4 nvmlReturn\_t DECLDIR nvmlDeviceGetCount (unsigned int \* deviceCount)

Retrieves the number of compute devices in the system. A compute device is a single GPU.

For all products.

On some platforms not all devices may be accessible due to permission restrictions. In these cases the device count will reflect only the GPUs that NVML can access.

#### **Parameters:**

deviceCount Reference in which to return the number of accessible devices

#### **Returns:**

- NVML\_SUCCESS if deviceCount has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if deviceCount is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.5 nvmlReturn\_t DECLDIR nvmlDeviceGetCurrPcieLinkGeneration (nvmlDevice\_t device, unsigned int \* currLinkGen)

Retrieves the current PCIe link generation

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

#### **Parameters:**

device The identifier of the target devicecurrLinkGen Reference in which to return the max PCIe link generation

#### Returns:

- NVML\_SUCCESS if currLinkGen has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or currLinkGen is null
- NVML\_ERROR\_NOT\_SUPPORTED if PCIe link information is not available
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.6 nvmlReturn\_t DECLDIR nvmlDeviceGetCurrPcieLinkWidth (nvmlDevice\_t device, unsigned int \* currLinkWidth)

Retrieves the current PCIe link width

For Tesla  $^{\text{TM}} and \ Quadro \ @products$  from the Fermi and Kepler families.

#### **Parameters:**

device The identifier of the target devicecurrLinkWidth Reference in which to return the max PCIe link generation

- NVML\_SUCCESS if *currLinkWidth* has been populated
- NVML ERROR UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or currLinkWidth is null
- NVML\_ERROR\_NOT\_SUPPORTED if PCIe link information is not available
- NVML\_ERROR\_UNKNOWN on any unexpected error

# 6.10.2.7 nvmlReturn\_t DECLDIR nvmlDeviceGetDetailedEccErrors (nvmlDevice\_t device, nvmlEccBitType\_t bitType, nvmlEccCounterType\_t counterType, nvmlEccErrorCounts\_t \* eccCounts)

Retrieves the detailed ECC error counts for the device.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families. Requires *NVML\_INFOROM\_ECC* version 2.0 or higher to report aggregate location-based ECC counts. Requires *NVML\_INFOROM\_ECC* version 1.0 or higher to report all other ECC counts. Requires ECC Mode to be enabled.

Detailed errors provide separate ECC counts for specific parts of the memory system.

Reports zero for unsupported ECC error counters when a subset of ECC error counters are supported.

See nvmlEccBitType\_t for a description of available bit types.

See <a href="nvmlEccCounterType\_t">nvmlEccCounterType\_t</a> for a description of available counter types.

See <a href="https://nvmlEccErrorCounts\_t">nvmlEccErrorCounts\_t</a> for a description of provided detailed ECC counts.

#### **Parameters:**

device The identifier of the target devicebitType Flag that specifies the bit-type of the errors.counterType Flag that specifies the counter-type of the errors.eccCounts Reference in which to return the specified ECC errors

#### **Returns:**

- NVML SUCCESS if eccCounts has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device, bitType or counterType is invalid, or eccCounts is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceClearEccErrorCounts()

## 6.10.2.8 nvmlReturn\_t DECLDIR nvmlDeviceGetDisplayMode (nvmlDevice\_t device, nvmlEnableState\_t \* display)

Retrieves the display mode for the device.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families.

This method indicates whether a physical display is currently connected to the device.

See nvmlEnableState\_t for details on allowed modes.

#### **Parameters:**

device The identifier of the target devicedisplay Reference in which to return the display mode

#### **Returns:**

- NVML\_SUCCESS if display has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or display is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature

## 6.10.2.9 nvmlReturn\_t DECLDIR nvmlDeviceGetDriverModel (nvmlDevice\_t device, nvmlDriverModel\_t \* current, nvmlDriverModel\_t \* pending)

Retrieves the current and pending driver model for the device.

For Tesla TM and Quadro ®products from the Fermi and Kepler families. For windows only.

On Windows platforms the device driver can run in either WDDM or WDM (TCC) mode. If a display is attached to the device it must run in WDDM mode. TCC mode is preferred if a display is not attached.

See nvmlDriverModel t for details on available driver models.

#### **Parameters:**

```
device The identifier of the target devicecurrent Reference in which to return the current driver modelpending Reference in which to return the pending driver model
```

#### **Returns:**

- NVML SUCCESS if current and pending have been populated
- NVML ERROR UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or both current and pending are NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the platform is not windows
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceSetDriverModel()

## 6.10.2.10 nvmlReturn\_t DECLDIR nvmlDeviceGetEccMode (nvmlDevice\_t device, nvmlEnableState\_t \* current, nvmlEnableState\_t \* pending)

Retrieves the current and pending ECC modes for the device.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families. Requires *NVML\_INFOROM\_ECC* version 1.0 or higher.

Changing ECC modes requires a reboot. The "pending" ECC mode refers to the target mode following the next reboot.

See nvmlEnableState\_t for details on allowed modes.

#### **Parameters:**

device The identifier of the target devicecurrent Reference in which to return the current ECC mode

**pending** Reference in which to return the pending ECC mode

#### **Returns:**

- NVML\_SUCCESS if current and pending have been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or either current or pending is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceSetEccMode()

#### 6.10.2.11 nvmlReturn\_t DECLDIR nvmlDeviceGetFanSpeed (nvmlDevice\_t device, unsigned int \* speed)

Retrieves the current operating speed of the device's fan.

For all discrete products with dedicated fans.

The fan speed is expressed as a percent of the maximum, i.e. full speed is 100%.

#### **Parameters:**

device The identifier of the target devicespeed Reference in which to return the fan speed percentage

### **Returns:**

- NVML\_SUCCESS if *speed* has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or speed is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not have a fan
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.12 nvmlReturn\_t DECLDIR nvmlDeviceGetHandleByIndex (unsigned int *index*, nvmlDevice\_t \* *device*)

Acquire the handle for a particular device, based on its index.

For all products.

Valid indices are derived from the *accessibleDevices* count returned by nvmlDeviceGetCount(). For example, if *accessibleDevices* is 2 the valid indices are 0 and 1, corresponding to GPU 0 and GPU 1.

The order in which NVML enumerates devices has no guarantees of consistency between reboots. For that reason it is recommended that devices be looked up by their PCI ids or board serial numbers. See <a href="https://nvmlDeviceGetHandleBySerial">nvmlDeviceGetHandleBySerial</a>() and <a href="https://nvmlDeviceGetHandleByPciBusId">nvmlDeviceGetHandleByPciBusId</a>().

#### **Parameters:**

*index* The index of the target GPU, >= 0 and < accessible Devices

device Reference in which to return the device handle

#### **Returns:**

- NVML\_SUCCESS if device has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if index is invalid or device is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.13 nvmlReturn\_t DECLDIR nvmlDeviceGetHandleByPciBusId (const char \* pciBusId, nvmlDevice\_t \* device)

Acquire the handle for a particular device, based on its PCI bus id.

For all products.

This value corresponds to the nvmlPciInfo\_t::busId returned by nvmlDeviceGetPciInfo().

#### **Parameters:**

pciBusId The PCI bus id of the target GPUdevice Reference in which to return the device handle

#### **Returns:**

- NVML\_SUCCESS if device has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if pciBusId is invalid or device is NULL
- NVML ERROR NOT FOUND if pciBusId does not match a valid device on the system
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.14 nvmlReturn\_t DECLDIR nvmlDeviceGetHandleBySerial (const char \* serial, nvmlDevice\_t \* device)

Acquire the handle for a particular device, based on its board serial number.

For all products.

This number corresponds to the value printed directly on the board, and to the value returned by nvmlDeviceGetSerial().

### **Deprecated**

Since more than one GPU can exist on a single board this function is deprecated in favor of nvmlDeviceGetH-andleByUUID. For dual GPU boards this function will return NVML\_ERROR\_INVALID\_ARGUMENT.

#### **Parameters:**

serial The board serial number of the target GPUdevice Reference in which to return the device handle

- NVML SUCCESS if device has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized

• NVML\_ERROR\_INVALID\_ARGUMENT if *serial* is invalid, *device* is NULL or more than one device has the same serial (dual GPU boards)

- NVML ERROR NOT FOUND if serial does not match a valid device on the system
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceGetSerial nvmlDeviceGetHandleByUUID

## 6.10.2.15 nvmlReturn\_t DECLDIR nvmlDeviceGetHandleByUUID (const char \* uuid, nvmlDevice\_t \* device)

Acquire the handle for a particular device, based on its globally unique immutable UUID associated with each device. For all products.

#### **Parameters:**

uuid The UUID of the target GPUdevice Reference in which to return the device handle

#### **Returns:**

- NVML\_SUCCESS if device has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if uuid is invalid or device is null
- NVML\_ERROR\_NOT\_FOUND if *uuid* does not match a valid device on the system
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceGetUUID

## 6.10.2.16 nvmlReturn\_t DECLDIR nvmlDeviceGetInforomVersion (nvmlDevice\_t device, nvmlInforomObject\_t object, char \* version, unsigned int length)

Retrieves the version information for the device's infoROM.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

Fermi and higher parts have non-volatile on-board memory for persisting device info, such as aggregate ECC counts. The version of the data structures in this memory may change from time to time. It will not exceed 16 characters in length (including the NULL terminator). See <a href="https://nwww.nvml.com/nvm

#### **Parameters:**

device The identifier of the target deviceobject The target infoROM objectversion Reference in which to return the infoROM version

length The maximum allowed length of the string returned in version

#### **Returns:**

- NVML\_SUCCESS if version has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if version is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if *length* is too small
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not have an infoROM
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.17 nvmlReturn\_t DECLDIR nvmlDeviceGetMaxClockInfo (nvmlDevice\_t device, nvmlClockType\_t type, unsigned int \* clock)

Retrieves the maximum clock speeds for the device.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

See <a href="nvmlClockType\_t">nvmlClockType\_t</a> for details on available clock information.

#### **Parameters:**

```
device The identifier of the target devicetype Identify which clock domain to queryclock Reference in which to return the clock speed in MHz
```

#### **Returns:**

- NVML\_SUCCESS if clock has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or clock is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device cannot report the specified clock
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.18 nvmlReturn\_t DECLDIR nvmlDeviceGetMaxPcieLinkGeneration (nvmlDevice\_t device, unsigned int \* maxLinkGen)

Retrieves the maximum PCIe link generation possible with this device and system

I.E. for a generation 2 PCIe device attached to a generation 1 PCIe bus the max link generation this function will report is generation 1.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

#### **Parameters:**

```
device The identifier of the target devicemaxLinkGen Reference in which to return the max PCIe link generation
```

### **Returns:**

• NVML\_SUCCESS if maxLinkGen has been populated

- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or maxLinkGen is null
- NVML\_ERROR\_NOT\_SUPPORTED if PCIe link information is not available
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.19 nvmlReturn\_t DECLDIR nvmlDeviceGetMaxPcieLinkWidth (nvmlDevice\_t device, unsigned int \* maxLinkWidth)

Retrieves the maximum PCIe link width possible with this device and system

I.E. for a device with a 16x PCIe bus width attached to a 8x PCIe system bus this function will report a max link width of 8.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families.

#### **Parameters:**

device The identifier of the target devicemaxLinkWidth Reference in which to return the max PCIe link generation

#### **Returns:**

- NVML\_SUCCESS if maxLinkWidth has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if device is invalid or maxLinkWidth is null
- NVML\_ERROR\_NOT\_SUPPORTED if PCIe link information is not available
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.20 nvmlReturn\_t DECLDIR nvmlDeviceGetMemoryInfo (nvmlDevice\_t device, nvmlMemory\_t \* memory)

Retrieves the amount of used, free and total memory available on the device, in bytes.

For all products.

Enabling ECC reduces the amount of total available memory, due to the extra required parity bits. Under WDDM most device memory is allocated and managed on startup by Windows.

Under Linux and Windows TCC, the reported amount of used memory is equal to the sum of memory allocated by all active channels on the device.

See <a href="nvmlMemory\_t">nvmlMemory\_t</a> for details on available memory info.

#### **Parameters:**

device The identifier of the target devicememory Reference in which to return the memory information

- NVML\_SUCCESS if *memory* has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or memory is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.21 nvmlReturn\_t DECLDIR nvmlDeviceGetName (nvmlDevice\_t device, char \* name, unsigned int length)

Retrieves the name of this device.

For all products.

The name is an alphanumeric string that denotes a particular product, e.g. Tesla ™C2070. It will not exceed 64 characters in length (including the NULL terminator). See nvmlConstants::NVML\_DEVICE\_NAME\_BUFFER\_-SIZE.

#### **Parameters:**

device The identifier of the target device

name Reference in which to return the product name

length The maximum allowed length of the string returned in name

#### **Returns:**

- NVML\_SUCCESS if *name* has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid, or name is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if *length* is too small

### 6.10.2.22 nvmlReturn\_t DECLDIR nvmlDeviceGetPciInfo (nvmlDevice\_t device, nvmlPciInfo\_t \* pci)

Retrieves the PCI attributes of this device.

For all products.

See <a href="nvmlPciInfo\_t">nvmlPciInfo\_t</a> for details on the available PCI info.

#### **Parameters:**

device The identifier of the target devicepci Reference in which to return the PCI info

### **Returns:**

- NVML\_SUCCESS if *pci* has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or pci is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.23 nvmlReturn\_t DECLDIR nvmlDeviceGetPerformanceState (nvmlDevice\_t device, nvmlPstates\_t \* pState)

Retrieves the current performance state for the device.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

See <a href="nvmlPstates\_t">nvmlPstates\_t</a> for details on allowed performance states.

#### **Parameters:**

device The identifier of the target devicepState Reference in which to return the performance state reading

#### **Returns:**

- NVML\_SUCCESS if pState has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or pState is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.24 nvmlReturn\_t DECLDIR nvmlDeviceGetPersistenceMode (nvmlDevice\_t device, nvmlEnableState\_t \* mode)

Retrieves the persistence mode associated with this device.

For all CUDA-capable products. For Linux only.

When driver persistence mode is enabled the driver software state is not torn down when the last client disconnects. By default this feature is disabled.

See nvmlEnableState\_t for details on allowed modes.

#### **Parameters:**

device The identifier of the target device

mode Reference in which to return the current driver persistence mode

#### **Returns:**

- NVML\_SUCCESS if mode has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or mode is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceSetPersistenceMode()

## 6.10.2.25 nvmlReturn\_t DECLDIR nvmlDeviceGetPowerManagementLimit (nvmlDevice\_t device, unsigned int \* limit)

Retrieves the power management limit associated with this device, in milliwatts.

For "GF11x" Tesla TM and Quadro ®products from the Fermi family.

• Requires NVML\_INFOROM\_POWER version 3.0 or higher.

For Tesla <sup>TM</sup>and Quadro ®products from the Kepler family.

• Does not require NVML\_INFOROM\_POWER object.

The power limit defines the upper boundary for the card's power draw. If the card's total power draw reaches this limit the power management algorithm kicks in.

This reading is only available if power management mode is supported. See nvmlDeviceGetPowerManagementMode.

#### **Parameters:**

device The identifier of the target devicelimit Reference in which to return the power management limit

#### **Returns:**

- NVML\_SUCCESS if *limit* has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or limit is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.26 nvmlReturn\_t DECLDIR nvmlDeviceGetPowerManagementMode (nvmlDevice\_t device, nvmlEnableState\_t \* mode)

Retrieves the power management mode associated with this device.

For "GF11x" Tesla TM and Quadro ®products from the Fermi family.

• Requires NVML\_INFOROM\_POWER version 3.0 or higher.

For Tesla TM and Quadro ®products from the Kepler family.

• Does not require NVML\_INFOROM\_POWER object.

This flag indicates whether any power management algorithm is currently active on the device. An enabled state does not necessarily mean the device is being actively throttled – only that that the driver will do so if the appropriate conditions are met.

See <a href="nvmlEnableState\_t">nvmlEnableState\_t</a> for details on allowed modes.

#### **Parameters:**

device The identifier of the target device

mode Reference in which to return the current power management mode

- NVML SUCCESS if mode has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or mode is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### 6.10.2.27 nvmlReturn\_t DECLDIR nvmlDeviceGetPowerState (nvmlDevice\_t device, nvmlPstates\_t \* pState)

Deprecated: Use nvmlDeviceGetPerformanceState. This function exposes an incorrect generalization.

Retrieve the current power state for the device.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

See <a href="nvmlPstates\_t">nvmlPstates\_t</a> for details on allowed power states.

#### **Parameters:**

```
device The identifier of the target devicepState Reference in which to return the power state reading
```

#### **Returns:**

- NVML\_SUCCESS if *pState* has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or pState is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### 6.10.2.28 nvmlReturn\_t DECLDIR nvmlDeviceGetPowerUsage (nvmlDevice\_t device, unsigned int \* power)

Retrieves the power usage reading for the device, in milliwatts. This is the power draw for the entire board, including GPU, memory, etc.

For "GF11x" Tesla TM and Quadro ®products from the Fermi family.

• Requires NVML\_INFOROM\_POWER version 3.0 or higher.

For Tesla TM and Quadro ®products from the Kepler family.

• Does not require NVML\_INFOROM\_POWER object.

The reading is accurate to within a range of +/- 5 watts. It is only available if power management mode is supported. See nvmlDeviceGetPowerManagementMode.

#### **Parameters:**

```
device The identifier of the target devicepower Reference in which to return the power usage information
```

- NVML\_SUCCESS if power has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or power is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support power readings
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.29 nvmlReturn\_t DECLDIR nvmlDeviceGetSerial (nvmlDevice\_t device, char \* serial, unsigned int length)

Retrieves the globally unique board serial number associated with this device's board.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

The serial number is an alphanumeric string that will not exceed 30 characters (including the NULL terminator). This number matches the serial number tag that is physically attached to the board. See <a href="https://nvmlconstants::NVML\_DEVICE\_-SERIAL\_BUFFER\_SIZE">nvmlconstants::NVML\_DEVICE\_-SERIAL\_BUFFER\_SIZE</a>.

#### Parameters:

device The identifier of the target deviceserial Reference in which to return the board/module serial numberlength The maximum allowed length of the string returned in serial

#### **Returns:**

- NVML\_SUCCESS if serial has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid, or serial is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if *length* is too small
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature

## 6.10.2.30 nvmlReturn\_t DECLDIR nvmlDeviceGetTemperature (nvmlDevice\_t device, nvmlTemperatureSensors\_t sensorType, unsigned int \* temp)

Retrieves the current temperature readings for the device, in degrees C.

For all discrete and S-class products.

See nvmlTemperatureSensors\_t for details on available temperature sensors.

#### **Parameters:**

device The identifier of the target devicesensorType Flag that indicates which sensor reading to retrievetemp Reference in which to return the temperature reading

- NVML\_SUCCESS if temp has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid, sensorType is invalid or temp is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not have the specified sensor
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.31 nvmlReturn\_t DECLDIR nvmlDeviceGetTotalEccErrors (nvmlDevice\_t device, nvmlEccBitType\_t bitType, nvmlEccCounterType\_t counterType, unsigned long long \* eccCounts)

Retrieves the total ECC error counts for the device.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families. Requires *NVML\_INFOROM\_ECC* version 1.0 or higher. Requires ECC Mode to be enabled.

The total error count is the sum of errors across each of the separate memory systems, i.e. the total set of errors across the entire device.

See <a href="nvmlEccCounterType\_t">nvmlEccCounterType\_t</a> for a description of available counter types.

#### **Parameters:**

device The identifier of the target devicebitType Flag that specifies the bit-type of the errors.counterType Flag that specifies the counter-type of the errors.eccCounts Reference in which to return the specified ECC errors

#### **Returns:**

- NVML\_SUCCESS if eccCounts has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device, bitType or counterType is invalid, or eccCounts is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceClearEccErrorCounts()

## 6.10.2.32 nvmlReturn\_t DECLDIR nvmlDeviceGetUtilizationRates (nvmlDevice\_t device, nvmlUtilization\_t \* utilization)

Retrieves the current utilization rates for the device's major subsystems.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

See nvmlUtilization\_t for details on available utilization rates.

### **Parameters:**

device The identifier of the target deviceutilization Reference in which to return the utilization information

- NVML\_SUCCESS if utilization has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if device is invalid or utilization is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.33 nvmlReturn\_t DECLDIR nvmlDeviceGetUUID (nvmlDevice\_t device, char \* uuid, unsigned int length)

Retrieves the globally unique immutable UUID associated with this device, as a 5 part hexadecimal string, that augments the immutable, board serial identifier.

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

The UUID is a globally unique identifier. It is the only available identifier for pre-Fermi-architecture products. It does NOT correspond to any identifier printed on the board. It will not exceed 80 characters in length (including the NULL terminator). See <a href="https://www.nvml.com/nvm

#### Parameters:

device The identifier of the target deviceuuid Reference in which to return the GPU UUIDlength The maximum allowed length of the string returned in uuid

#### **Returns:**

- NVML SUCCESS if uuid has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid, or uuid is NULL
- NVML ERROR INSUFFICIENT SIZE if length is too small
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.34 nvmlReturn\_t DECLDIR nvmlDeviceGetVbiosVersion (nvmlDevice\_t device, char \* version, unsigned int length)

Get VBIOS version of the device.

For all products.

#### **Parameters:**

device The identifier of the target deviceversion Reference to which to return the VBIOS versionlength The maximum allowed length of the string returned in version

- NVML\_SUCCESS if version has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid, or version is NULL
- NVML\_ERROR\_INSUFFICIENT\_SIZE if *length* is too small
- NVML\_ERROR\_UNKNOWN on any unexpected error

## 6.10.2.35 nvmlReturn\_t DECLDIR nvmlDeviceOnSameBoard (nvmlDevice\_t device1, nvmlDevice\_t device2, int \* onSameBoard)

Check if the GPU devices are on the same physical board.

#### **Parameters:**

device1 The first GPU devicedevice2 The second GPU device

onSameBoard Reference in which to return the status. Non-zero indicates that the GPUs are on the same board.

- NVML\_SUCCESS when onSameBoard has been populated
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if dev1, dev2 or onSameBoard are invalid
- NVML\_ERROR\_UNKNOWN on any unexpected error

6.11 Unit Commands 53

### **6.11 Unit Commands**

#### **Functions**

nvmlReturn\_t DECLDIR nvmlUnitSetLedState (nvmlUnit\_t unit, nvmlLedColor\_t color)

### **6.11.1 Detailed Description**

This chapter describes NVML operations that change the state of the unit. For S-class products. Each of these requires root/admin access. Non-admin users will see an NVML\_ERROR\_NO\_PERMISSION error code when invoking any of these methods.

### **6.11.2** Function Documentation

#### 6.11.2.1 nvmlReturn\_t DECLDIR nvmlUnitSetLedState (nvmlUnit\_t unit, nvmlLedColor\_t color)

Set the LED state for the unit. The LED can be either green (0) or amber (1).

For S-class products. Requires root/admin permissions.

This operation takes effect immediately.

Current S-Class products don't provide unique LEDs for each unit. As such, both front and back LEDs will be toggled in unison regardless of which unit is specified with this command.

See <a href="nvmlLedColor\_t">nvmlLedColor\_t</a> for available colors.

#### **Parameters:**

```
unit The identifier of the target unitcolor The target LED color
```

#### **Returns:**

- NVML\_SUCCESS if the LED color has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if unit or color is invalid
- NVML\_ERROR\_NOT\_SUPPORTED if this is not an S-class product
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlUnitGetLedState()

### **6.12** Device Commands

#### **Functions**

- nvmlReturn\_t DECLDIR nvmlDeviceSetPersistenceMode (nvmlDevice\_t device, nvmlEnableState\_t mode)
- nvmlReturn\_t DECLDIR nvmlDeviceSetComputeMode (nvmlDevice\_t device, nvmlComputeMode\_t mode)
- nvmlReturn\_t DECLDIR nvmlDeviceSetEccMode (nvmlDevice\_t device, nvmlEnableState\_t ecc)
- nvmlReturn\_t DECLDIR nvmlDeviceClearEccErrorCounts (nvmlDevice\_t device, nvmlEccCounterType\_t counterType)
- nvmlReturn\_t DECLDIR nvmlDeviceSetDriverModel (nvmlDevice\_t device, nvmlDriverModel\_t driverModel, unsigned int flags)

### **6.12.1** Detailed Description

This chapter describes NVML operations that change the state of the device. Each of these requires root/admin access. Non-admin users will see an NVML\_ERROR\_NO\_PERMISSION error code when invoking any of these methods.

#### **6.12.2** Function Documentation

## 6.12.2.1 nvmlReturn\_t DECLDIR nvmlDeviceClearEccErrorCounts (nvmlDevice\_t device, nvmlEccCounterType\_t counterType)

Clear the ECC error counts for the device.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families. Requires *NVML\_INFOROM\_ECC* version 2.0 or higher to clear aggregate location-based ECC counts. Requires *NVML\_INFOROM\_ECC* version 1.0 or higher to clear all other ECC counts. Requires root/admin permissions. Requires ECC Mode to be enabled.

Sets all of the specified ECC counters to 0, including both detailed and total counts.

This operation takes effect immediately.

See <a href="nvmlEccCounterType\_t">nvmlEccCounterType\_t</a> for details on available counter types.

### **Parameters:**

device The identifier of the target device

*counterType* Flag that indicates which type of errors should be cleared.

### **Returns:**

- NVML SUCCESS if the error counts were cleared
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or counterType is invalid
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

- nvmlDeviceGetDetailedEccErrors()
- nvmlDeviceGetTotalEccErrors()

6.12 Device Commands 55

## 6.12.2.2 nvmlReturn\_t DECLDIR nvmlDeviceSetComputeMode (nvmlDevice\_t device, nvmlComputeMode\_t mode)

Set the compute mode for the device.

For all CUDA-capable products. Requires root/admin permissions.

The compute mode determines whether a GPU can be used for compute operations and whether it can be shared across contexts.

This operation takes effect immediately. Under Linux it is not persistent across reboots and always resets to "Default". Under windows it is persistent.

Under windows compute mode may only be set to DEFAULT when running in WDDM

See <a href="nvmlComputeMode\_t">nvmlComputeMode\_t</a> for details on available compute modes.

#### **Parameters:**

device The identifier of the target devicemode The target compute mode

#### **Returns:**

- NVML\_SUCCESS if the compute mode was set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or mode is invalid
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML ERROR NO PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

### See also:

nvmlDeviceGetComputeMode()

## 6.12.2.3 nvmlReturn\_t DECLDIR nvmlDeviceSetDriverModel (nvmlDevice\_t device, nvmlDriverModel\_t driverModel, unsigned int flags)

Set the driver model for the device.

For Tesla <sup>TM</sup> and Quadro ®products from the Fermi and Kepler families. For windows only. Requires root/admin permissions.

On Windows platforms the device driver can run in either WDDM or WDM (TCC) mode. If a display is attached to the device it must run in WDDM mode.

It is possible to force the change to WDM (TCC) while the display is still attached with a force flag (nvmlFlagForce). This should only be done if the host is subsequently powered down and the display is detached from the device before the next reboot.

This operation takes effect after the next reboot.

Under windows driver model may only be set to WDDM when running in DEFAULT compute mode.

See nvmlDriverModel\_t for details on available driver models. See nvmlFlagDefault and nvmlFlagForce

### **Parameters:**

device The identifier of the target device

driverModel The target driver model

flags Flags that change the default behavior

#### **Returns:**

- NVML SUCCESS if the driver model has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or driverModel is invalid
- NVML\_ERROR\_NOT\_SUPPORTED if the platform is not windows or the device does not support this
  feature
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceGetDriverModel()

#### 6.12.2.4 nvmlReturn\_t DECLDIR nvmlDeviceSetEccMode (nvmlDevice\_t device, nvmlEnableState\_t ecc)

Set the ECC mode for the device.

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families. Requires *NVML\_INFOROM\_ECC* version 1.0 or higher. Requires root/admin permissions.

The ECC mode determines whether the GPU enables its ECC support.

This operation takes effect after the next reboot.

See nvmlEnableState\_t for details on available modes.

#### **Parameters:**

device The identifier of the target deviceecc The target ECC mode

### **Returns:**

- NVML\_SUCCESS if the ECC mode was set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML ERROR INVALID ARGUMENT if device is invalid or ecc is invalid
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceGetEccMode()

6.12 Device Commands 57

## 6.12.2.5 nvmlReturn\_t DECLDIR nvmlDeviceSetPersistenceMode (nvmlDevice\_t device, nvmlEnableState\_t mode)

Set the persistence mode for the device.

For all CUDA-capable products. For Linux only. Requires root/admin permissions.

The persistence mode determines whether the GPU driver software is torn down after the last client exits.

This operation takes effect immediately. It is not persistent across reboots. After each reboot the persistence mode is reset to "Disabled".

See nvmlEnableState\_t for available modes.

#### **Parameters:**

device The identifier of the target devicemode The target persistence mode

#### **Returns:**

- NVML\_SUCCESS if the persistence mode was set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if device is invalid or mode is invalid
- NVML\_ERROR\_NOT\_SUPPORTED if the device does not support this feature
- NVML\_ERROR\_NO\_PERMISSION if the user doesn't have permission to perform this operation
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceGetPersistenceMode()

### **6.13** Event Handling Methods

#### **Data Structures**

struct nvmlEventData\_t

### **Modules**

• Event Types

### **Typedefs**

typedef struct nvmlEventSet\_st \* nvmlEventSet\_t

#### **Functions**

- nvmlReturn\_t DECLDIR nvmlEventSetCreate (nvmlEventSet\_t \*set)
- nvmlReturn\_t DECLDIR nvmlDeviceRegisterEvents (nvmlDevice\_t device, unsigned long long eventTypes, nvmlEventSet\_t set)
- nvmlReturn\_t DECLDIR nvmlDeviceGetSupportedEventTypes (nvmlDevice\_t device, unsigned long long \*eventTypes)
- nvmlReturn\_t DECLDIR nvmlEventSetWait (nvmlEventSet\_t set, nvmlEventData\_t \*data, unsigned int timeoutms)
- nvmlReturn\_t DECLDIR nvmlEventSetFree (nvmlEventSet\_t set)

### **6.13.1** Detailed Description

This chapter describes methods that NVML can perform against each device to register and wait for some event to occur.

### **6.13.2** Typedef Documentation

### 6.13.2.1 typedef struct nvmlEventSet\_st\* nvmlEventSet\_t

Handle to an event set

### **6.13.3** Function Documentation

## 6.13.3.1 nvmlReturn\_t DECLDIR nvmlDeviceGetSupportedEventTypes (nvmlDevice\_t device, unsigned long long \* eventTypes)

Returns information about events supported on device

For all products.

Events are not supported on Windows. So this function returns an empty mask in eventTypes on Windows.

### **Parameters:**

device The identifier of the target device

eventTypes Reference in which to return bitmask of supported events

#### **Returns:**

- NVML\_SUCCESS if the eventTypes has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if eventType is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

Event Types nvmlDeviceRegisterEvents

## 6.13.3.2 nvmlReturn\_t DECLDIR nvmlDeviceRegisterEvents (nvmlDevice\_t device, unsigned long long eventTypes, nvmlEventSet\_t set)

Starts recording of events on a specified devices and add the events to specified nvmlEventSet\_t

For Tesla <sup>TM</sup>and Quadro ®products from the Fermi and Kepler families. Ecc events are available only on ECC enabled devices (see <a href="nvmlDeviceGetTotalEccErrors">nvmlDeviceGetTotalEccErrors</a>) Power capping events are available only on Power Management enabled devices (see <a href="nvmlDeviceGetPowerManagementMode">nvmlDeviceGetPowerManagementMode</a>)

For linux only.

**IMPORTANT:** Operations on set are not thread safe

This call starts recording of events on specific device. All events that occurred before this call are not recorded. Checking if some event occurred can be done with nvmlEventSetWait

#### **Parameters:**

```
device The identifier of the target deviceeventTypes Bitmask of Event Types to recordset Set to which add new event types
```

#### **Returns:**

- NVML SUCCESS if the event has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if eventTypes is invalid or set is NULL
- NVML\_ERROR\_NOT\_SUPPORTED if the platform does not support this feature or some of requested event types
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

Event Types nvmlDeviceGetSupportedEventTypes nvmlEventSetWait nvmlEventSetFree

### 6.13.3.3 nvmlReturn\_t DECLDIR nvmlEventSetCreate (nvmlEventSet\_t \* set)

Create an empty set of events. Event set should be freed by nvmlEventSetFree

#### **Parameters:**

set Reference in which to return the event handle

#### **Returns:**

- NVML\_SUCCESS if the event has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if set is NULL
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlEventSetFree

### 6.13.3.4 nvmlReturn\_t DECLDIR nvmlEventSetFree (nvmlEventSet\_t set)

Releases events in the set

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

#### **Parameters:**

set Reference to events to be released

#### **Returns:**

- NVML\_SUCCESS if the event has been successfully released
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

nvmlDeviceRegisterEvents

## 6.13.3.5 nvmlReturn\_t DECLDIR nvmlEventSetWait (nvmlEventSet\_t set, nvmlEventData\_t \* data, unsigned int timeoutms)

Waits on events and delivers events

For Tesla TM and Quadro ®products from the Fermi and Kepler families.

If some events are ready to be delivered at the time of the call, function returns immediately. If there are no events ready to be delivered, function sleeps till event arrives but not longer than specified timeout. This function in certain conditions can return before specified timeout passes (e.g. when interrupt arrives)

#### **Parameters:**

set Reference to set of events to wait on

data Reference in which to return event data

timeoutms Maximum amount of wait time in ms for registered event

### **Returns:**

- NVML\_SUCCESS if the data has been set
- NVML\_ERROR\_UNINITIALIZED if the library has not been successfully initialized
- NVML\_ERROR\_INVALID\_ARGUMENT if data is NULL
- NVML\_ERROR\_TIMEOUT if no event arrived in specified timeout or interrupt arrived
- NVML\_ERROR\_UNKNOWN on any unexpected error

#### See also:

Event Types nvmlDeviceRegisterEvents

## **Chapter 7**

## **Data Structure Documentation**

### 7.1 nvmlEccErrorCounts\_t Struct Reference

#include <nvml.h>

### **Data Fields**

- unsigned long long l1Cache

  L1 cache errors.
- unsigned long long l2Cache

  L2 cache errors.
- unsigned long long deviceMemory Device memory errors.
- unsigned long long registerFile Register file errors.

### 7.1.1 Detailed Description

Detailed ECC error counts for a device.

### 7.2 nvmlEventData\_t Struct Reference

#include <nvml.h>

### **Data Fields**

- nvmlDevice\_t device

  Specific device where the event occurred.
- unsigned long long eventType

  Information about what specific event occurred.

### 7.2.1 Detailed Description

Information about occurred event

# 7.3 nvmlHwbcEntry\_t Struct Reference

#include <nvml.h>

## 7.3.1 Detailed Description

Description of HWBC entry

## 7.4 nvmlLedState\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

• char cause [256]

If amber, a text description of the cause.

• nvmlLedColor\_t color GREEN or AMBER.

## 7.4.1 Detailed Description

LED states for an S-class unit.

## 7.5 nvmlMemory\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

- unsigned long long total

  Total installed FB memory (in bytes).
- unsigned long long free

  Unallocated FB memory (in bytes).
- unsigned long long used

  Allocated FB memory (in bytes). Note that the driver/GPU always sets aside a small amount of memory for bookkeeping.

## 7.5.1 Detailed Description

Memory allocation information for a device.

## 7.6 nvmlPciInfo\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

• char busId [16]

The tuple domain:bus:device.function PCI identifier (& NULL terminator).

• unsigned int domain

The PCI domain on which the device's bus resides, 0 to 0xffff.

• unsigned int bus

The bus on which the device resides, 0 to 0xff.

• unsigned int device

The device's id on the bus, 0 to 31.

• unsigned int pciDeviceId

The combined 16-bit device id and 16-bit vendor id.

• unsigned int pciSubSystemId

The 32-bit Sub System Device ID.

#### 7.6.1 Detailed Description

PCI information about a GPU device.

## 7.7 nvmlProcessInfo\_t Struct Reference

#include <nvml.h>

## **Data Fields**

• unsigned int pid

Process ID.

• unsigned long long usedGpuMemory

Amount of used GPU memory in bytes. < Under WDDM, NVML\_VALUE\_NOT\_AVAILABLE is always reported < because Windows KMD manages all the memory and not the NVIDIA driver.

### 7.7.1 Detailed Description

Information about running compute processes on the GPU

## 7.8 nvmlPSUInfo\_t Struct Reference

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

#### **Data Fields**

• char state [256]

The power supply state.

unsigned int current

PSU current (A).

• unsigned int voltage

PSU voltage (V).

• unsigned int power

PSU power draw (W).

## 7.8.1 Detailed Description

Power usage information for an S-class unit. The power supply state is a human readable string that equals "Normal" or contains a combination of "Abnormal" plus one or more of the following:

- High voltage
- Fan failure
- · Heatsink temperature
- · Current limit
- Voltage below UV alarm threshold
- Low-voltage
- SI2C remote off command
- MOD\_DISABLE input
- · Short pin transition

## 7.9 nvmlUnitFanInfo\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

• unsigned int speed Fan speed (RPM).

• nvmlFanState\_t state

Flag that indicates whether fan is working properly.

## 7.9.1 Detailed Description

Fan speed reading for a single fan in an S-class unit.

# 7.10 nvmlUnitFanSpeeds\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

- nvmlUnitFanInfo\_t fans [24] Fan speed data for each fan.
- unsigned int count

  Number of fans in unit.

## 7.10.1 Detailed Description

Fan speed readings for an entire S-class unit.

## 7.11 nvmlUnitInfo\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

- char name [96]

  Product name.
- char id [96]

  Product identifier.
- char serial [96]

  Product serial number.
  - Trouver serial minioen

• char firmware Version [96] *Firmware version.* 

## 7.11.1 Detailed Description

Static S-class unit info.

## 7.12 nvmlUtilization\_t Struct Reference

#include <nvml.h>

#### **Data Fields**

• unsigned int gpu

Percent of time over the past second during which one or more kernels was executing on the GPU.

• unsigned int memory

Percent of time over the past second during which global (device) memory was being read or written.

## 7.12.1 Detailed Description

Utilization information for a device.

# **Index**

Constants, 27	nvmlDeviceEnumvs, 22
	NVML_ERROR_NOT_FOUND
Device Commands, 54	nvmlDeviceEnumvs, 22
Device Enums, 18	NVML_ERROR_NOT_SUPPORTED
Device Queries, 34	nvmlDeviceEnumvs, 22
Device Structs, 17	NVML_ERROR_TIMEOUT
	nvmlDeviceEnumvs, 22
Error reporting, 26	NVML_ERROR_UNINITIALIZED
Event Handling Methods, 58	nvmlDeviceEnumvs, 22
Event Types, 24	NVML ERROR UNKNOWN
	nvmlDeviceEnumvs, 22
Initialization and Cleanup, 25	NVML_FAN_FAILED
NVML_AGGREGATE_ECC	nvmlUnitStructs, 23
nvmlDeviceEnumvs, 20	NVML_FAN_NORMAL
NVML_CLOCK_GRAPHICS	nvmlUnitStructs, 23
nvmlDeviceEnumvs, 19	NVML_FEATURE_DISABLED
NVML_CLOCK_MEM	nvmlDeviceEnumvs, 21
nvmlDeviceEnumvs, 19	NVML_FEATURE_ENABLED
NVML_CLOCK_SM	nvmlDeviceEnumvs, 21
nvmlDeviceEnumvs, 19	NVML_INFOROM_ECC
NVML_COMPUTEMODE_DEFAULT	nvmlDeviceEnumvs, 21
nvmlDeviceEnumvs, 20	NVML_INFOROM_OEM
NVML_COMPUTEMODE_EXCLUSIVE_PROCESS	nvmlDeviceEnumvs, 21
	NVML_INFOROM_POWER
nvmlDeviceEnumvs, 20 NVML_COMPUTEMODE_EXCLUSIVE_THREAD	nvmlDeviceEnumvs, 21
nvmlDeviceEnumvs, 20	NVML_LED_COLOR_AMBER
	nvmlUnitStructs, 23
NVML_COMPUTEMODE_PROHIBITED	NVML_LED_COLOR_GREEN
nvmlDeviceEnumvs, 20	nvmlUnitStructs, 23
NVML_DOUBLE_BIT_ECC	NVML_PSTATE_0
nvmlDeviceEnumvs, 20	nvmlDeviceEnumvs, 21
NVML_DRIVER_WDDM	NVML_PSTATE_1
nvmlDeviceEnumvs, 20	nvmlDeviceEnumvs, 21
NVML_DRIVER_WDM	NVML PSTATE 10
nvmlDeviceEnumvs, 20	nvmlDeviceEnumvs, 21
NVML_ERROR_ALREADY_INITIALIZED	NVML_PSTATE_11
nvmlDeviceEnumvs, 22	nvmlDeviceEnumvs, 21
NVML_ERROR_DRIVER_NOT_LOADED	NVML_PSTATE_12
nvmlDeviceEnumvs, 22	nvmlDeviceEnumvs, 21
NVML_ERROR_INSUFFICIENT_POWER	NVML_PSTATE_13
nvmlDeviceEnumvs, 22	nvmlDeviceEnumvs, 21
NVML_ERROR_INSUFFICIENT_SIZE	NVML_PSTATE_14
nvmlDeviceEnumvs, 22	nvmlDeviceEnumvs, 21
NVML_ERROR_INVALID_ARGUMENT	NVML_PSTATE_15
nvml Eppor no permission	nvmlDeviceEnumvs, 21

NVML_PSTATE_2	NVML_DEVICE_VBIOS_VERSION_BUFFER
nvmlDeviceEnumvs, 21	SIZE, 27
NVML_PSTATE_3	NVML_SYSTEM_DRIVER_VERSION
nvmlDeviceEnumvs, 21	BUFFER_SIZE, 27
NVML_PSTATE_4	NVML_SYSTEM_NVML_VERSION_BUFFER_
nvmlDeviceEnumvs, 21	SIZE, 27
NVML_PSTATE_5	nvmlDeviceClearEccErrorCounts
nvmlDeviceEnumvs, 21	nvmlDeviceCommands, 54
NVML_PSTATE_6	nvmlDeviceCommands
nvmlDeviceEnumvs, 21	nvmlDeviceClearEccErrorCounts, 54
NVML_PSTATE_7	nvmlDeviceSetComputeMode, 54
nvmlDeviceEnumvs, 21	nvmlDeviceSetDriverModel, 55
NVML_PSTATE_8	nvmlDeviceSetEccMode, 56
nvmlDeviceEnumvs, 21	nvmlDeviceSetPersistenceMode, 56
NVML_PSTATE_9	nvmlDeviceEnumvs
nvmlDeviceEnumvs, 21	NVML_AGGREGATE_ECC, 20
NVML_PSTATE_UNKNOWN	NVML CLOCK GRAPHICS, 19
nvmlDeviceEnumvs, 21	NVML_CLOCK_MEM, 19
NVML_SINGLE_BIT_ECC	NVML_CLOCK_SM, 19
nvmlDeviceEnumvs, 20	NVML_COMPUTEMODE_DEFAULT, 20
NVML SUCCESS	NVML_COMPUTEMODE_EXCLUSIVE
<del>-</del>	PROCESS, 20
nvmlDeviceEnumvs, 22 NVML TEMPERATURE GPU	
<del>-</del>	NVML_COMPUTEMODE_EXCLUSIVE
nvmlDeviceEnumvs, 22	THREAD, 20
NVML_VOLATILE_ECC	NVML_COMPUTEMODE_PROHIBITED, 20
nvmlDeviceEnumvs, 20	NVML_DOUBLE_BIT_ECC, 20
NVML_DEVICE_INFOROM_VERSION_BUFFER	NVML_DRIVER_WDDM, 20
SIZE	NVML_DRIVER_WDM, 20
nvmlConstants, 27	NVML_ERROR_ALREADY_INITIALIZED, 22
NVML_DEVICE_NAME_BUFFER_SIZE	NVML_ERROR_DRIVER_NOT_LOADED, 22
nvmlConstants, 27	NVML_ERROR_INSUFFICIENT_POWER, 22
NVML_DEVICE_SERIAL_BUFFER_SIZE	NVML_ERROR_INSUFFICIENT_SIZE, 22
nvmlConstants, 27	NVML_ERROR_INVALID_ARGUMENT, 22
NVML_DEVICE_UUID_BUFFER_SIZE	NVML_ERROR_NO_PERMISSION, 22
nvmlConstants, 27	NVML_ERROR_NOT_FOUND, 22
NVML_DEVICE_VBIOS_VERSION_BUFFER_SIZE	NVML_ERROR_NOT_SUPPORTED, 22
nvmlConstants, 27	NVML_ERROR_TIMEOUT, 22
NVML_SYSTEM_DRIVER_VERSION_BUFFER	NVML_ERROR_UNINITIALIZED, 22
SIZE	NVML_ERROR_UNKNOWN, 22
nvmlConstants, 27	NVML_FEATURE_DISABLED, 21
NVML_SYSTEM_NVML_VERSION_BUFFER_SIZE	NVML_FEATURE_ENABLED, 21
nvmlConstants, 27	NVML_INFOROM_ECC, 21
NVML_VALUE_NOT_AVAILABLE	NVML_INFOROM_OEM, 21
nvmlDeviceStructs, 17	NVML_INFOROM_POWER, 21
nvmlClockType_t	NVML_PSTATE_0, 21
nvmlDeviceEnumvs, 19	NVML_PSTATE_1, 21
nvmlComputeMode_t	NVML PSTATE 10, 21
nvmlDeviceEnumvs, 19	NVML_PSTATE_11, 21
nvmlConstants	NVML_PSTATE_12, 21
NVML_DEVICE_INFOROM_VERSION	NVML_PSTATE_13, 21
BUFFER_SIZE, 27	NVML_PSTATE_14, 21
NVML_DEVICE_NAME_BUFFER_SIZE, 27	NVML_PSTATE_15, 21
NVML_DEVICE_NAME_BOTTER_SIZE, 27 NVML_DEVICE_SERIAL_BUFFER_SIZE, 27	NVML_PSTATE_13, 21 NVML_PSTATE_2, 21
NVML_DEVICE_SERIAL_BUFFER_SIZE, 27  NVML_DEVICE_UUID_BUFFER_SIZE, 27	NVML_PSTATE_2, 21 NVML_PSTATE_3, 21
IN VIVIL_DE VICE_UUID_DUFFER_SIZE, 2/	1 V IVIL_F & 1741 L_3, 41

NVML_PSTATE_4, 21	nvmlDeviceQueries, 43
NVML_PSTATE_5, 21	nvmlDeviceGetMaxPcieLinkGeneration
NVML_PSTATE_6, 21	nvmlDeviceQueries, 43
NVML_PSTATE_7, 21	nvmlDeviceGetMaxPcieLinkWidth
NVML_PSTATE_8, 21	nvmlDeviceQueries, 44
NVML_PSTATE_9, 21	nvmlDeviceGetMemoryInfo
NVML_PSTATE_UNKNOWN, 21	nvmlDeviceQueries, 44
NVML_SINGLE_BIT_ECC, 20	nvmlDeviceGetName
NVML_SUCCESS, 22	nvmlDeviceQueries, 44
NVML_TEMPERATURE_GPU, 22	nvmlDeviceGetPciInfo
NVML_VOLATILE_ECC, 20	nvmlDeviceQueries, 45
nvmlClockType_t, 19	nvmlDeviceGetPerformanceState
nvmlComputeMode_t, 19	nvmlDeviceQueries, 45
nvmlDriverModel_t, 20	nvmlDeviceGetPersistenceMode
nvmlEccBitType_t, 20	nvmlDeviceQueries, 46
nvmlEccCounterType_t, 20	nvmlDeviceGetPowerManagementLimit
nvmlEnableState_t, 20	nvmlDeviceQueries, 46
nvmlInforomObject_t, 21	nvmlDeviceGetPowerManagementMode
nvmlPstates_t, 21	nvmlDeviceQueries, 47
nvmlReturn_t, 21	nvmlDeviceGetPowerState
nvmlTemperatureSensors_t, 22	nvmlDeviceQueries, 47
nvmlDeviceGetClockInfo	nvmlDeviceGetPowerUsage
nvmlDeviceQueries, 35	nvmlDeviceQueries, 48
nvmlDeviceGetComputeMode	nvmlDeviceGetSerial
nvmlDeviceQueries, 35	nvmlDeviceQueries, 48
nvmlDeviceGetComputeRunningProcesses	nvmlDeviceGetSupportedEventTypes
nvmlDeviceQueries, 36	nvmlEvents, 58
nvmlDeviceGetCount	nvmlDeviceGetTemperature
nvmlDeviceQueries, 36	nvmlDeviceQueries, 49
nvmlDeviceGetCurrPcieLinkGeneration	nvmlDeviceGetTotalEccErrors
nvmlDeviceQueries, 37	nvmlDeviceQueries, 49
nvmlDeviceGetCurrPcieLinkWidth	nvmlDeviceGetUtilizationRates
nvmlDeviceQueries, 37	nvmlDeviceQueries, 50
nvmlDeviceGetDetailedEccErrors	nvmlDeviceGetUUID
nvmlDeviceQueries, 37	nvmlDeviceQueries, 50
nvmlDeviceGetDisplayMode	nvmlDeviceGetVbiosVersion
nvmlDeviceQueries, 38	nvmlDeviceQueries, 51
nvmlDeviceGetDriverModel	nvmlDeviceOnSameBoard
nvmlDeviceQueries, 39	nvmlDeviceQueries, 51
nvmlDeviceGetEccMode	nymlDeviceQueries
nymlDeviceQueries, 39	nvmlDeviceGetClockInfo, 35
nvmlDeviceGetFanSpeed	nvmlDeviceGetCockinio, 35
nvmlDeviceQueries, 40	nvmlDeviceGetComputeRunningProcesses, 36
	nvmlDeviceGetComputeRummigF10cesses, 30
nvmlDeviceGetHandleByIndex	
nvmlDeviceQueries, 40	nvmlDeviceGetCurrPcieLinkGeneration, 37
nvmlDeviceGetHandleByPciBusId	nvmlDeviceGetCurrPcieLinkWidth, 37
nvmlDeviceQueries, 41	nvmlDeviceGetDetailedEccErrors, 37
nvmlDeviceGetHandleBySerial	nvmlDeviceGetDisplayMode, 38
nvmlDeviceQueries, 41	nvmlDeviceGetDriverModel, 39
nvmlDeviceGetHandleByUUID	nvmlDeviceGetEccMode, 39
nvmlDeviceQueries, 42	nvmlDeviceGetFanSpeed, 40
nvmlDeviceGetInforomVersion	nvmlDeviceGetHandleByIndex, 40
nvmlDeviceQueries, 42	nvmlDeviceGetHandleByPciBusId, 41
nvmlDeviceGetMaxClockInfo	nvmlDeviceGetHandleBySerial, 41

nvmlDeviceGetHandleByUUID, 42	nvmlEventSet_t
nvmlDeviceGetInforomVersion, 42	nvmlEvents, 58
nvmlDeviceGetMaxClockInfo, 43	nvmlEventSetCreate
nvmlDeviceGetMaxPcieLinkGeneration, 43	nvmlEvents, 59
nvmlDeviceGetMaxPcieLinkWidth, 44	nvmlEventSetFree
nvmlDeviceGetMemoryInfo, 44	nvmlEvents, 60
nvmlDeviceGetName, 44	nvmlEventSetWait
nvmlDeviceGetPciInfo, 45	nvmlEvents, 60
nvmlDeviceGetPerformanceState, 45	nvmlEventType
nvmlDeviceGetPersistenceMode, 46	nvmlEventTypePState, 24
nvmlDeviceGetPowerManagementLimit, 46	nvmlEventTypePState
nvmlDeviceGetPowerManagementMode, 47	nvmlEventType, 24
nvmlDeviceGetPowerState, 47	nvmlFanState_t
nvmlDeviceGetPowerUsage, 48	nvmlUnitStructs, 23
nvmlDeviceGetSerial, 48	nvmlHwbcEntry_t, 65
nvmlDeviceGetTemperature, 49	nvmlInforomObject_t
nvmlDeviceGetTotalEccErrors, 49	nvmlDeviceEnumvs, 21
nvmlDeviceGetUtilizationRates, 50	nymlInit
nvmlDeviceGetUUID, 50	nvmlInitializationAndCleanup, 25
nvmlDeviceGetVbiosVersion, 51	nvmlInitializationAndCleanup
nvmlDeviceOnSameBoard, 51	nymlInit, 25
nvmlDeviceRegisterEvents	nvmlShutdown, 25
nvmlEvents, 59	nvmlLedColor_t
nvmlDeviceSetComputeMode	nvmlUnitStructs, 23
nvmlDeviceCommands, 54	nvmlLedState_t, 66
nvmlDeviceSetDriverModel	nvmlMemory_t, 67
nvmlDeviceSciDiiverModel	nvmlPciInfo_t, 68
nvmlDeviceSetEccMode	
	nvmlProcessInfo_t, 69
nvmlDeviceCommands, 56	nvmlPstates_t
nvmlDeviceSetPersistenceMode	nvmlDeviceEnumvs, 21
nvmlDeviceCommands, 56	nvmlPSUInfo_t, 70
nvmlDeviceStructs	nvmlReturn_t
NVML_VALUE_NOT_AVAILABLE, 17	nvmlDeviceEnumvs, 21
nvmlDriverModel_t	nvmlShutdown
nvmlDeviceEnumvs, 20	nvmlInitializationAndCleanup, 25
nvmlEccBitType_t	nvmlSystemGetDriverVersion
nvmlDeviceEnumvs, 20	nvmlSystemQueries, 28
nvmlEccCounterType_t	nvmlSystemGetHicVersion
nvmlDeviceEnumvs, 20	nvmlUnitQueries, 30
nvmlEccErrorCounts_t, 63	nvmlSystemGetNVMLVersion
nvmlEnableState_t	nvmlSystemQueries, 28
nvmlDeviceEnumvs, 20	nvmlSystemGetProcessName
nvmlErrorReporting	nvmlSystemQueries, 29
nvmlErrorString, 26	nvmlSystemQueries
nvmlErrorString	nvmlSystemGetDriverVersion, 28
nvmlErrorReporting, 26	nvmlSystemGetNVMLVersion, 28
nvmlEventData_t, 64	nvmlSystemGetProcessName, 29
nvmlEvents	nvmlTemperatureSensors_t
nvmlDeviceGetSupportedEventTypes, 58	nvmlDeviceEnumvs, 22
nvmlDeviceRegisterEvents, 59	nvmlUnitCommands
nvmlEventSet_t, 58	nvmlUnitSetLedState, 53
nvmlEventSetCreate, 59	nvmlUnitFanInfo_t, 71
nvmlEventSetFree, 60	nvmlUnitFanSpeeds_t, 72
nvmlEventSetWait, 60	nvmlUnitGetCount

nvmlUnitQueries, 30
nvmlUnitGetDevices
nvmlUnitQueries, 31
nvmlUnitGetFanSpeedInfo
nvmlUnitQueries, 31
nvmlUnitGetHandleByIndex
nvmlUnitQueries, 31
nvmlUnitGetLedState
nvmlUnitQueries, 32
nvmlUnitGetPsuInfo
nvmlUnitQueries, 32
nvmlUnitGetTemperature
nvmlUnitQueries, 33
nvmlUnitGetUnitInfo
nvmlUnitQueries, 33
nvmlUnitInfo_t, 73
nvmlUnitQueries
nvmlSystemGetHicVersion, 30
nvmlUnitGetCount, 30
nvmlUnitGetDevices, 31
nvmlUnitGetFanSpeedInfo, 31
nvmlUnitGetHandleByIndex, 31
nvmlUnitGetLedState, 32
nvmlUnitGetPsuInfo, 32
nvmlUnitGetTemperature, 33
nvmlUnitGetUnitInfo, 33
nvmlUnitSetLedState
nvmlUnitCommands, 53
nvmlUnitStructs
NVML_FAN_FAILED, 23
NVML_FAN_NORMAL, 23
NVML_LED_COLOR_AMBER, 23
NVML_LED_COLOR_GREEN, 23
nvmlFanState_t, 23
nvmlLedColor_t, 23
nvmlUtilization_t, 74
System Queries, 28
Unit Commands, 53
Unit Queries, 30
Unit Structs, 23

#### 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.

#### Trademarks

NVIDIA, the NVIDIA logo, GeForce, Tesla, and Quadro are trademarks and/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

© 2007-2012 NVIDIA Corporation. All rights reserved.

