Introduction

1. General Purpose

NVWMI provider allows WMI clients to query and to monitor parameters of NVIDIA hardware. It is implemented as a decoupled WMI provider in a system service (nvwmi.exe in 32-bit or nvwmi64.exe in 64-bit flavors of
Windows).

2. General Requirements

- NVIDIA hardware
- operational NVIDIA display driver
- installed NVWMI provider

3. Version-specific

For detailed information about version-specific changes, please refer to the implementation Version-specific Implementation Details.

4. OS-specific

Windows XP has limited support. Supported classes are:

- **System**
- **Gpu**
- **Board**
- **Cooler**
- **ThermalProbe**
- **CoolerEvent**
- **ThermalEvent**

NVWMI Performance counter provider does not support Windows XP.

5. General Implementation Details

NVWMI implements several WMI providers inside `\\.\root\cimv2\nv` namespace:

- instance provider
- method provider
- property provider
- event provider
- performance counters provider
All classes are statically defined in the MOF file, there are no dynamically defined classes (e.g. no class provider).

**Static vs. dynamic instances.**
In WMI, there are two separate types of object instances - singletons and dynamic instances. If the MOF class is declared as a singleton, it has just one instance. The path to such an instance is always known to the WBEM engine. Therefore the exact path to the instance is not required to address the instance. All other classes have dynamic instances. This means that the user must specify either a precise path or WQL operator which addressess one or more instances.

classes **System, DisplayManager** and **ProfileManager** are declared as MOF singletons, all other classes have dynamic instancing. Thus addressing instances of any non-singleton class, requires either **where** WQL clause or instance path.

**GPU handles**
GPU handles are not guaranteed to persist across driver upgrades or hardware changes. Enabling or disabling hardware, installing new NVIDIA hardware, upgrading driver or changing SLI configuration might change value of any GPU handle.

**Object reference**
"Object reference" is a UTF-16 string which contains the path to the WMI object. References could be used to build WQL queries. References (i.e. object paths) could be used to address instances.

There is a potential for signed/unsigned mismatch, as the WMI COM interface **IWBemClassObject** requires that even uint32 MOF type is set as a variant of VI_I4 type. Conversely, when retrieving properties of WMI objects via **IWBemClassObject::Get**, supply **CIMTYPE** * pvtType to retrieve the correct type. Do not rely on **VARIANT** * pVal.

For more information about **IWBemClassObject::Get**, see [MSDN article about IWBemClassObject::Get](https://docs.microsoft.com/en-us/previous-versions/windows/windows-server-2003/ee184699(v=ws.10))

Example of the object reference (relative to NV namespace):
references instance of the **DisplayMode** class with id=452288000. **DisplayMode** instance with this id corresponds to the 1600x1200x32 Hz display mode (for versions 2.2, 2.3). Note that for classes with multiple keys, the reference string must contain all of them. Here is an example of **Gpu** instance reference:

Gpu.id=3137339648, uname="Quadro 5000"

When specifying references at command line, be sure to use quotation marks as appropriate (remember that reference is a string) - example of setting display mode on remote system **wks-1**

```
wmic /node:"wks-1" /namespace:nv path displaygrid where id=0 call setModeRef "DisplayMode.id=452288000"
```

6. WMI General Information

Please refer to the MSDN documentation on WMI providers for information about general functionality and use.

Links to relevant MSDN pages:

- [WMI Reference](#)
- [Creating WMI clients](#)

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Legal Notice

Copyright (c) 2010-2016 NVIDIA Corporation. All rights reserved.

Notice
This software may not, in whole or in part, be copied through any means, mechanical, electromechanical, or otherwise, without the express permission of NVIDIA Corporation.

Information furnished is believed to be accurate and reliable. However, NVIDIA assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties, which may result from its use. No License is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in the software are subject to change without notice.

NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks
NVIDIA and the NVIDIA logo are registered trademarks or trademarks of NVIDIA Corporation in the United States and/or other countries.
Microsoft, Windows, and the Windows logo are registered trademarks of Microsoft Corporation.

Other company and product names may be trademarks or registered trademarks of the respective companies with which they are associated.

---

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Version-specific Implementation Details

Version 2.31

- Version 2.31 implements new properties of the `VideoCodec` class with data about video encoding statistics:
  - `VideoCodec::encoderSessionsCount`
  - `VideoCodec::averageFps`
  - `VideoCodec::averageLatency`
- Version 2.31 implements several performance counters with video encoding statistics:
  - Video Encoder Sessions
  - Video Encoder Average FPS
  - Video Encoder Average Latency (ms)

Version 2.30

- Version 2.30 implements support of user-defined GPU utilization events. User-defined GPU utilization events are not persistent across service restarts. By default no GPU utilization events are defined.
- Version 2.30 implements `Gpu::createUtilizationEvent` method. This
method creates new utilization event in a given domain.

- Version 2.30 implements `Gpu::deleteUtilizationEvents` method. This method deletes previously created events in a given domain.
- Version 2.30 implements `Gpu::getAllUtilizationEvents` method. This method returns a string with all utilization events in a given domain.
- Version 2.30 implements support of user-defined filters of display modes. Custom display mode filters are not persistent across service restarts. By default display modes are not filtered.
- Version 2.30 implements `DisplayManager::createModeFilter` method. This method creates new custom display mode filter.
- Version 2.30 implements `DisplayManager::deleteModeFilter` method. This method deletes previously created custom display mode filter.
- Version 2.30 implements `DisplayManager::enumModeFilters` method. This method enumerates existing display mode filters.

**Version 2.29**

- Version 2.29 implements dithering control in the `Display` class. Following new properties are supported:
  - `Display::ditherState`
  - `Display::ditherBits`
  - `Display::ditherMode`
- Version 2.29 implements `Display::setDither` method. This method changes dithering parameters.
- Version 2.29 implements `DisplayManager::saveCustomTimings` method for saving custom timings. Custom timings are saved in XML file, compatible with NVIDIA Control Panel.
- Version 2.29 implements `DisplayManager::loadCustomTimings` method for loading custom timings.

**Version 2.28**

- Version 2.28 implements new class `VideoCodec` which contains information about VideoEngine encoder and decoder. Instances of `VideoCodec` class are accessible only as properties of the `Gpu` class.
- Version 2.28 implements new property of the Gpu class with data about VideoEngine:
  - Gpu::videoCodec

- Version 2.28 implements several performance counters for VideoEngine:
  - % Video Encoder usage
  - % Video Decoder usage

  New features of the version 2.28 were ported to the 2.27 and released as 2.27.2.

**Version 2.27**

- Version 2.27 implements new class PcieLink which contains information about PCI-Express link to GPU. Instances of PcieLink class are accessible only as properties of the Gpu class.
- Version 2.27 implements new properties of the Gpu class with data about PCI-Express link to GPU:
  - Gpu::pcieDownstreamWidth
  - Gpu::pcieGpu

- Version 2.27 implements several performance counters for PCI-Express link to GPU:
  - % Bus Usage
  - % Video Usage
  - % FB Usage
  - PCI-E downstream width to GPU
  - PCI-E current width to GPU
  - PCI-E current speed to GPU Mbps

- Version 2.27 implements support of cloned displays in display profiles.

- Version 2.27 implements DisplayManager::createClone method. This method clones a screen of display into screens of other displays. Two modes supported:
  - basic clone - identical display mode is set for all displays
  - smart clone - source display with larger screen resolution is cloned into smaller resolution of a target display(s) with image panning
**Version 2.26**

- Version 2.26 improves access to display functionality. All `DisplayGrid` instances are searched for matching `Display` and corresponding method of `DisplayGrid` is called when match is found.
- Version 2.26 implements `Display::setDisplayMode` method as a shortcut to the `DisplayGrid::setDisplayMode`.
- Version 2.26 implements `Display::setDisplayModeById` method as a shortcut to the `DisplayGrid::setDisplayModeById`.
- Version 2.26 implements `Display::setDisplayModeByRef` method as a shortcut to the `DisplayGrid::setDisplayModeByRef`.

**Version 2.25**

- Version 2.25 implements support of licensable feature management in `System` and `Gpu` classes.
- Version 2.25 implements `Gpu::licensableFeatures` property. This property contains array of strings with names of all available GRID features to license.
- Version 2.25 implements `Gpu::licensableStatus` property. This property contains array of enumerated status codes of all available GRID features to license.
- Version 2.25 implements `Gpu::enableLicensedFeature` method. This method enables given features, based on available licenses on a server.
- Version 2.25 implements `Gpu::disableLicensedFeature` method. This method disables given features.
- Version 2.25 implements `System::licensingPort` property. This property contains networking port of a license server.
- Version 2.25 implements `System::licensingServer` property. This property contains network address of a license server.
- Version 2.25 implements `System::setLicensingServer` method. This method changes address and port of license server.
- Version 2.25 implements `System::unsetLicensingServer` method. This method resets address and port of license server and relinquish a license after reboot.
Version 2.24

- Version 2.24 changed type of the `Board::serialNumber` property. For user readability it is changed to a string with human-readable serial number of a GPU board.

Version 2.23

- Version 2.23 implements "Fan Speed (RPM)" performance counter.
- Version 2.23 changed type of several properties of the `Gpu` class from unsigned to signed. This change improves clarity of output when property is not supported in hardware and property value is set to -1. Following properties were modified:
  - `Gpu::powerSampleCount`
  - `Gpu::powerSamplingPeriod`
  - `Gpu::percentGpuUsage`
  - `Gpu::percentGpuMemoryUsage`

Version 2.22

- Version 2.22 implements support of Windows gamma ramp in display profiles (class `DisplayProfile`) and system profiles (class `Profile` of type "system"=5).
- Version 2.22 implements `Display::saveGammaRamp` method. This method saves gamma ramp data of individual display.
- Version 2.22 implements `Display::setGammaRamp` method. This method applies previously saved gamma ramp data of individual display.
- Version 2.22 implements `Display::setGammaRampBasic` method. This method generates monochromatic gamma ramp for individual display from three input parameters - gamma, contrast and brightness.
- Version 2.22 implements `DisplayGrid::saveGammaRamp` method. This method saves Color Space Conversion data of all displays in a display grid.
- Version 2.22 implements `DisplayGrid::setGammaRamp` method.
This method applies Color Space Conversion data to all displays in a display grid.

- Version 2.22 implements `DisplayGrid::setGammaRampBasic` method. This method generates monochromatic gamma ramp for all displays in display grid from three input parameters - gamma, contrast and brightness.
- Version 2.22 implements `DisplayGridInfo::gammaRampFilePath` property. It contains path to the binary file with display grid gamma ramp data.
- Version 2.22 implements singleton class `DesktopManager` and several methods for manipulating virtual nView desktops - `DesktopManager::getAllDesktops`, `DesktopManager::createDesktop`, `DesktopManager::editDesktop`, `DesktopManager::deleteDesktop`.

**Version 2.21**

- Version 2.21 implements support of Color Space Conversion (CSC) in display profiles (class `DisplayProfile`) and system profiles (class `Profile` of type "system"=5).
- Version 2.21 implements `Display::saveCSC` method. This method saves Color Space Conversion data of individual display.
- Version 2.21 implements `Display::setCSC` method. This method applies previously saved Color Space Conversion data of individual display.
- Version 2.21 implements `DisplayGrid::saveCSC` method. This method saves Color Space Conversion data of all displays in a display grid.
- Version 2.21 implements `DisplayGrid::setCSC` method. This method applies Color Space Conversion data to all displays in a display grid.
- Version 2.21 implements `DisplayGridInfo::cscFilePath` property. It contains path to the binary file with display grid CSC data.

**Version 2.20**

- Version 2.20 implements `Display::getCurrentTiming` method. This
method returns a string with current display timing parameters.

- Version 2.20 implements `Gpu::fakeEDIDOnPort` method. This method fakes EDID on specified GPU port.
- Version 2.20 improves `SyncTopology::id`, `SyncDelay::id`, `Sync::id`, `Gpu::id`, `Board::id` and `DisplayGrid::id` properties by using more user-friendly values.
- Version 2.20 implements `Display::displayConnectorType` property.
- Version 2.20 implements `Gpu::nvapiId` property.
- Version 2.20 implements `Gpu::archId` property. It contains numerical ID of the GPU architecture.
- Version 2.20 implements `Gpu::archName` property. It contains a string with the GPU architecture name (e.g. "Kepler", "Maxwell" etc.)
- Version 2.20 deprecates `handle` property in several classes. While `handle` is still available, `id` (WMI-only object identifier) is recommended for scripting and interactive commands and `nvapiId` for interfacing with NVAPI.

**Version 2.19**

- Version 2.19 implements `System::setLogState` method. This method changes logging verbosity and other logging settings.
- Version 2.19 implements additional functionality in the `System` class. Following new properties are supported:
  - `System::logTypes` property. This property is a bitmap - see `System::setLogState` for details.
  - `System::logFilter` property. This property is a bitmask.

**Version 2.18**

- Version 2.18 implements additional functionality in the `Board` class. Following new properties are supported:
  - `Board::nvapiId` property. It contains NVAPI `Board` ID, which was previously reported as `Board::id`.
  - `Board::chipSKU` - SKU of the GPU chip(s) on this board
  - `Board::chipSKUMod` - SKU modifier the GPU chip(s) on this board
  - `Board::project` - project (board) number
  - `Board::projectSKU` - project (board) SKU number
• Version 2.18 implements additional functionality in the **Cooler** class. Following new properties are supported:
  - **Cooler::id** - unique object identifier.
  - **Cooler::minSpeed** - Minimal cooler speed. Units are revolutions per minute (RPM) for fans.
  - **Cooler::maxSpeed** - Maximal cooler speed. Units are revolutions per minute (RPM) for fans.
• Version 2.18 deprecates **Cooler::handle** property. While **Cooler::handle** is still available, **Cooler::id** should be used instead.
• Version 2.18 implements additional functionality in the **ThermalProbe** class. Following new properties are supported:
  - **ThermalProbe::id** - unique object identifier.
  - **ThermalProbe::defaultMinTemperature** - Minimal default temperature.
  - **ThermalProbe::defaultMaxTemperature** - Maximal default temperature.
• Version 2.18 deprecates **ThermalProbe::handle** property. While **ThermalProbe::handle** is still available, **ThermalProbe::id** should be used instead.
• Version 2.18 implements **ProfileManager::deleteDesktopProfile** method.
• Version 2.18 implements nView version as a property of **System** class - **System::vernViewDesktopManager**.
• Version 2.18 implements "% GPU Memory Usage" performance counter.
• Version 2.18 implements **Gpu::percentGpuMemoryUsage** property.
• Version 2.18 implements **Gpu::devicelInfo** property which reports the Device Id, Vendor Id, Sub-device Id, Revision Id.
• Version 2.18 implements **System::machineType** property which reports whether the system is Desktop or Mobile.
• Version 2.18 implements multiple **Display** grid support on single GPU.

**Version 2.17**

• Version 2.17 implements user-friendly Display IDs. User-friendly display IDs are created from the GPU ordinal and the display ordinal
according to this formula:
\[ id = (GPU \#) \times 1000 + (Display \#); \]
where (GPU #) is a GPU ordinal as enumerated by NVAPI and starts from 1, (Display#) is a display ordinal in per-GPU enumeration by NVAPI, and also starts with 1.

- Version 2.17 implements `Display::locus` property. It contains string, representing user-friendly ID as "(GPU #).(Display #)". For example, a display, attached to 1st output of 1st GPU will have `Display::locus`="1.1" and `Display::id`=1001.
- Version 2.17 implements `Display::nvapild` property. It contains NVAPI `Display` ID, which was previously reported as `Display::id`.
- Version 2.17 implements class `Ecc`. `Gpu::ecc` property is a reference to an instance of the `Ecc` class. This feature is not supported if there are multiple GPU topologies enabled.
- Version 2.17 implements `DisplayManager::tryCustomTiming` method. This method validates the new Custom timing before creating/ applying, for the given displays.
- Version 2.17 implements `DisplayManager::createCustomTiming` method. This method creates the new Custom timing for the given displays.
- Version 2.17 implements `DisplayManager::deleteCustomTiming` method. This method deletes the existing Custom timing for the given displays.
- Version 2.17 implements `DisplayManager::editCustomTiming` method. This method deletes the existing Custom timing and creates a new Custom timing, for the given displays.
- Version 2.17 implements `DisplayManager::enumCustomTimings` method. This method enumerates all the existing Custom modes, for the given displays.

**Version 2.16**

- Version 2.16 provides functionality related to System Profiles.
- Version 2.16 implements `ProfileManager::currentProfileDesktop` and `ProfileManager::defaultProfileDesktop` properties.
- Version 2.16 implements `ProfileManager::saveSystemProfile` method. This method saves the given System Profile.
- Version 2.16 implements `ProfileManager::applySystemProfile` method. This method applies the System Profiles.
- Version 2.16 implements `ProfileManager::currentSystemProfile` and `ProfileManager::defaultSystemProfile` properties.

**Version 2.15**

- Version 2.15 enhances `DisplayManager::createDisplayGrids` method with support for default and custom topologies.
- Version 2.15 enhances `DisplayManager::validateDisplayGrids` method with support for default and custom topologies.
- Version 2.15 implements `Gpu::fakeEDID` method. This method fakes EDID on all connectors of a given output type and could be used to create virtual displays in headless configurations. For example, if one physical display is attached to the DisplayPort connector of a single-GPU board with 2 DisplayPort connectors then invoking `Gpu::fakeEDID` for DisplayPort output will override EDID for attached physical display plus a new virtual display will appear.
- Version 2.15 implements `DisplayManager::fakeEDIDAll` method. This method fakes EDID on all connectors of a given output type on all GPUs and could be used to create virtual displays in headless configurations.
- NOTE: This version of NVWMI will not be compatible to older version of nView. Some of the nView setting in Profile class are removed, refer Profile settings available in NVWMI for details . To apply nView setting the user must be logged in.
- Version 2.15 deprecates properties System::verSDIOutputDriver and System::verSDICaptureDriver.
- Version 2.15 deprecates `DisplayManager::setEdidAll` method.

**Version 2.14**

- Version 2.14 implements `DisplayManager::createDisplayGrids` method. This method creates multiple display grids, with a single call.
- Version 2.14 implements `DisplayManager::validateDisplayGrids` method. This method validates multiple display grids, with a single call.
• Version 2.14 implements `DisplayManager::setGridPositions` method. This method repositions all display grids at once. Note that all active grids must be repositioned together.
• Version 2.14 implements support for passive stereo modes.
• Version 2.14 deprecates method `ProfileManager::getAllApplicationProfiles()`. This method has been superseded by `getAllProfiles()`.
• Version 2.14 implements method `ProfileManager::getAllProfiles()`. This method lists all profiles of a given type as a string in format "\<id1\> : \<profile name1\> ;\<id2\> : \<profile name2\>; ...". Profiles of type 3D Application Profile, 3D Global Profile, nView Profile, Display Profile, System Profile are supported

Version 2.13

• Version 2.13 implements profile settings reflection mechanism by providing SettingsTable MOF class:
  ◦ List all settings of a given profile type
  ◦ Convert between profile setting name and profile setting ID
  ◦ Obtain information about a profile setting by specifying either setting ID or setting name
  ◦ **Setting** IDs and setting names could be used to modify or query profile setting values

See also:
  PowerShell examples - Using profile settings reflection mechanism, table with profile settings and setting names - Profile settings available in NVWMI

• Version 2.13 implements `DisplayManager::validateDisplayGridById` method to aid user in display grid creation.
• Version 2.13 implements `DisplayGrid::setRotation` method. It rotates all displays in a display grid to a given angle. In this version only first element of an array with rotation angles is applied to all displays, in future versions per-display rotations will be supported.

Version 2.12
• Version 2.12 implements classes **Sync** and **SyncTopology** with basic support of video signal synchronization on Sync-compatible hardware. Additional information about hardware compatibility see **NVIDIA Quadro Sync**.
• Version 2.12 implements **Board::serialNumber** property.

**Version 2.11**

• Version 2.11 implements support for measuring GPU power consumption. Following properties reported on high-end Kepler Quadro boards:
  ○ **Gpu::power** - in watts per sampling period
  ○ **Gpu::powerSampleCount** - number of power sensor samples per iteration
  ○ **Gpu::powerSamplingPeriod** - sampling period in milliseconds
On unsupported hardware these properties are set to 0.
• Version 2.11 improves hardware detection and differentiation.
• Version 2.11 improves **Gpu** class by reporting memory-related properties in megabytes rather than in kilobytes.

**Version 2.10**

• Version 2.10 implements **ApplicationProfile** class with basic support for 3D application profiles.
  ○ Querying instances of **ApplicationProfile** class without specifying **ApplicationProfile::id** will list all active application profiles at the moment of query. Such query could be used to monitor profile activity on a system and to verify that appropriate profile is associated with application of interest.
  ○ Querying instances of **ApplicationProfile** class with specific **ApplicationProfile::id** will return an instance with matching id only. Such query could be used to examine or modify a specific application profile.
• Version 2.10 implements method **ProfileManager::getAllApplicationProfiles()**. It lists all application profiles of a given type as a string in format "\<id1\> : \<profile name1\> ; \<id2\> : \<profile name2\> ; ...". Only profiles of type 0 (3D application profiles) are supported at present.
Version 2.10 implements display rotation support. Property `Display::rotation` reports current rotation. Following values are supported:

- **0** - No rotation
- **1** - rotated 90 degrees
- **2** - rotated 180 degrees
- **3** - rotated 270 degrees

Use method `Display::setRotation()` to change display rotation.

Version 2.10 implements additional properties in `DisplayProfile` class:

- "positionCols" (32-bit unsigned) ID=0x586748C2, default value is 0. Display positions in columns of display grid (in pixels).
- "positionRows" (32-bit unsigned) ID=0x5879DDC6, default value is 0. Display positions in rows of display grid (in pixels).
- "primaryId" (32-bit unsigned) ID=0x58597B87, no default value current value is always present. Display ID of the GDI primary display.
- "rotation" (string with rotation values as unsigned integers, separated by semicolon :) ID=0x58DECFA8, default value is "0".
- "scaling" (string with scaling values as unsigned integers, separated by semicolon :) ID=0x587B0428, default value is "0".

Version 2.9

- Version 2.9 improves performance of the profile framework
- Version 2.9 implements saving of an nView desktop profile in the method `ProfileManager::saveDesktopProfile`
- Version 2.9 supports saving a display EDID in the method `Display::saveEDID`

Version 2.8

- Version 2.8 has revised API. Many classes, their properties and methods are renamed or moved into other classes.
  - `update()` method is removed from all classes
  - class NV class is split into classes `System, DisplayManager`
and **ProfileManager**
- methods NV::setScaling, NV::setEdid are refactored into
  **DisplayManager::setScalingAll** and
  DisplayManager::setEDIDAll
- class Mode is renamed **DisplayMode**
- per-row and and per-column property names in class
  **DisplayGrid** now start with either **row** or **col** prefix

**Version 2.7**

- Version 2.7 implements additional functionality in **DisplayProfile**
  class. Following settings are supported:
  - "rows" (32-bit unsigned) **ID=0x5822918D**, default value is **1**
  - "cols" (32-bit unsigned) **ID=0x58E21BD4**, default value is **1**
  - "displayIds" (string with display IDs in hexadecimal notation
    separated by semicolon ;) **ID=0x58B21E43**, no default value
    current value is always present
  - "overlapCols" (string with overlap values as signed integers,
    separated by semicolon ;) **ID=0x58EB619D**, default value is "0"
  - "overlapRows" (string with overlap values as signed integers,
    separated by semicolon ;) **ID=0x58DD36C1**, default value is "0"

- Version 2.7 implements several methods for manipulating display
  profiles.
  - **ProfileManager::saveDisplayProfiles()**
  - **ProfileManager::applyDisplayProfiles()**

- Version 2.7 implements several methods for manipulating desktop
  profiles.
  - **ProfileManager::loadDesktopProfile()**
  - **ProfileManager::lockDesktopProfile()**

- Version 2.7 supports additional parameters for V-sync control.
  Method **ProfileManager::setVsync()** accepts following values:
  - 0 Passive
  - 1 Off
  - 2 On
  - 3 Adaptive (refresh rate)
  - 4 Adaptive (half refresh rate)

- Version 2.7 implements SDI capture driver version as a property of
NV class
- Version 2.7 allows to force EDID on inactive displays

Version 2.6
- Version 2.6 implements framework for generic profile support.
- Version 2.6 implements DisplayPath class and **DisplayProfile** class with limited functionality.
- Version 2.6 overhauls display-related classes.
  - Class Monitor renamed to **Display**
  - Class DisplaySource is deprecated.
  - Class DisplayTarget is deprecated.
  - Methods of DisplayTarget and DisplaySource classes are moved to **DisplayGrid** and **Display**.

Version 2.5
- Version 2.5 implements DisplayGrid::effectiveMode property. It reports bezel-corrected display dimensions while taking into account display grid layout, answering the question "how many unique pixels could you see?". Note that DisplayGrid::currentMode reports per-display video mode.

For example, for horizontal display grid 1x2 (1 row and 2 columns) with each display at 1600x1200 and with overlap -25 for the second display, DisplayGrid::currentMode (**DisplayGrid::displayModePhysical** in version 2.8 and later) will be reported as 1600x1200 whereas DisplayGrid::effectiveMode (**DisplayGrid::displayModeVirtual** in version 2.8 and later) will be 3175x1200. General formula is:

effective size = (per-display size)*(# of display for a given dimension)+(sum of all overlaps for a given dimension)

Version 2.4
- Version 2.4 implements several methods for defining display grids:
  - NV::createDisplayGridById() - renamed **DisplayManager::createDisplayGridById()** in version 2.8 and later
- NV::createDisplayGridByRef() - renamed 
  `DisplayManager::createDisplayGridByRef()` in version 2.8 and later
- NV::createDisplayGridByName() - renamed 
  `DisplayManager::createDisplayGridByName()` in version 2.8 and later

- Version 2.4 implements **OverlapLimits** class.
- Version 2.4 implements several methods of forcing EDID:
  - NV::setEdid() - renamed `DisplayManager::setEDIDAll()` in version 2.8 and later
  - Monitor::setEdid() - renamed `Display::setEDID()` in version 2.8 and later
- Version 2.4 implements method for turning Vertical Synchronization (V-sync) on and off - NV::setVsync().

NOTE: Creating display grids on QuadroPlex is partially supported. Creation of some legitimate display grid configurations straddling more than one GPU might fail.

**Version 2.3**

- Version 2.3 implements class **DisplayGrid** with basic support for multi-monitor and Mosaic configurations.
- Version 2.3 implements additional support for SELECT operator.
- Version 2.3 improves versioning by adding descriptions into binary version block and OS service records.
- Version 2.3 implements functionality differentiation between Quadro, NVS and GeForce. GeForce does not support advanced functionality. Note that for mixed configurations (GeForce and Quadro or NVS) functionality is equal to GeForce.
  - Basic classes (**Version**, **NamedObject**)
  - **System** class is supported
  - **Gpu** class is supported
  - **Board** class is supported
  - **Cooler** class is supported
  - **ThermalProbe** class is supported
  - Event classes are supported (**CoolerEvent**, **ThermalEvent**)
  - All other classes are supported only on Quadro (both desktop
and notebook) and NVS desktops

Version 2.2

- Version 2.2 implements several methods to change display parameters:
  - NV::setScaling() - renamed DisplayManager::setScalingAll() in version 2.8 and later
  - DisplaySource::setMode() - refactored to DisplayGrid::setMode() in version 2.8 and later
  - DisplayTarget::setMode() - refactored to DisplayGrid::setMode() in version 2.8 and later
  - DisplayTarget::setScaling() - deprecated and removed in version 2.8 and later
- Version 2.2 implements additional properties and dynamic instancing of DisplaySource and DisplayTarget classes.
- Version 2.2 implements a Mode class (renamed DisplayMode in version 2.8 and later) which contains basic information about display modes.
- Version 2.2 improves logging with OS event log support. See Logging and tracing NVWMI activity.

Version 2.1

- Version 2.1 implements methods for static and dynamic instances.
- Version 2.1 implements a Board class to simplify diagnostic of multi-GPU and multi-board configurations.
- Version 2.1 implements many display-related classes - Monitor, DisplaySource, DisplayTarget. Functionality is limited to read-only properties.

Version 2.0

- Version 2.0 implements basic WQL support. Complex WQL operators not supported.
- Version 2.0 implements individual versioning of each MOF class as well as version-reporting of many software and hardware components of a system.
• Version 2.0 derives most of classes from **NamedObject** class to allow consistent addressing of objects.

**Version 1.2**

• Version 1.2 implements access to physical GPUs' thermal environment and provides basic information such as GPU and board names.
• Clients could monitor events from GPU coolers and thermal probes. Hardware that is capable of reporting this information is required.
• Version 1.2 doesn't support WQL queries and doesn't expose methods to modify current configuration. All properties are read-only.

---

**NVIDIA**

*Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.*
NVWMI compatibility

General information about NVWMI compatibility

NVWMI versions have inter-dependencies with the NVIDIA Display Driver Package versions. It is strongly recommended that customers follow the table below to choose a supported combination of NVWMI & NVIDIA Display Driver to guarantee functionality of all the NVWMI features of a specific version.

Note: If NVWMI gets installed on an earlier NVIDIA Driver Package than the minimal recommended driver version there is a possibility that some of the NVWMI features/commands may not function and return a message indicating that the function is unsupported.

Please refer to the NVWMI release notes for details on which NVWMI/NVIDIA Display Driver versions add support for specific NVWMI features/commands.

Table of recommended version combinations
<table>
<thead>
<tr>
<th>NVWMI version</th>
<th>Driver Release</th>
<th>version of the Display Driver package</th>
<th>Minimal nView version</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.00</td>
<td>Release 275</td>
<td>275.36</td>
<td>NA</td>
</tr>
<tr>
<td>2.02</td>
<td>Release 285</td>
<td>285.58</td>
<td>NA</td>
</tr>
<tr>
<td>2.04</td>
<td>Release 295</td>
<td>295.73</td>
<td>NA</td>
</tr>
<tr>
<td>2.06</td>
<td>Release 304</td>
<td>305.93</td>
<td>NA</td>
</tr>
<tr>
<td>2.07</td>
<td>Release 304</td>
<td>305.93</td>
<td>NA</td>
</tr>
<tr>
<td>2.09</td>
<td>Release 310</td>
<td>310.90</td>
<td>NA</td>
</tr>
<tr>
<td>2.10</td>
<td>Release 313</td>
<td>314.07</td>
<td>NA</td>
</tr>
<tr>
<td>2.12</td>
<td>Release 319</td>
<td>320.86</td>
<td>140.75</td>
</tr>
<tr>
<td>2.14</td>
<td>Release 325</td>
<td>320.86</td>
<td>140.75</td>
</tr>
<tr>
<td>2.15</td>
<td>Release 331</td>
<td>331.65</td>
<td>140.84</td>
</tr>
<tr>
<td>2.16.0</td>
<td>Release 331</td>
<td>331.65</td>
<td>140.84</td>
</tr>
<tr>
<td>2.16.1</td>
<td>Release 334</td>
<td>334.67</td>
<td>141.24</td>
</tr>
<tr>
<td>2.17</td>
<td>Release 337</td>
<td>337.50</td>
<td>141.24</td>
</tr>
<tr>
<td>2.18</td>
<td>Release 340</td>
<td>340.52</td>
<td>141.24</td>
</tr>
<tr>
<td>Version</td>
<td>Release Number</td>
<td>Rate 1</td>
<td>Rate 2</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>2.19</td>
<td>343</td>
<td>340.52</td>
<td>141.24</td>
</tr>
<tr>
<td>2.20</td>
<td>346</td>
<td>347.88</td>
<td>141.36</td>
</tr>
<tr>
<td>2.21</td>
<td>349</td>
<td>348.27</td>
<td>146.33</td>
</tr>
<tr>
<td>2.22</td>
<td>352</td>
<td>353.82</td>
<td>146.78</td>
</tr>
<tr>
<td>2.23</td>
<td>352</td>
<td>353.82</td>
<td>146.78</td>
</tr>
<tr>
<td>2.24</td>
<td>352</td>
<td>353.82</td>
<td>146.78</td>
</tr>
<tr>
<td>2.25</td>
<td>361</td>
<td>361.75</td>
<td>146.78</td>
</tr>
<tr>
<td>2.26</td>
<td>364</td>
<td>364.72</td>
<td>146.78</td>
</tr>
<tr>
<td>2.27</td>
<td>367</td>
<td>368.39</td>
<td>147.00</td>
</tr>
<tr>
<td>2.28</td>
<td>370</td>
<td>372.95</td>
<td>147.00</td>
</tr>
<tr>
<td>2.29</td>
<td>375</td>
<td>375.63</td>
<td>147.00</td>
</tr>
<tr>
<td>2.30</td>
<td>378</td>
<td>378.49</td>
<td>147.00</td>
</tr>
<tr>
<td>2.31</td>
<td>384</td>
<td>387.39</td>
<td>147.00</td>
</tr>
</tbody>
</table>
**Cooler and Thermal Events**

**Cooler Events and Diagnostics**

**What is Fan Percent Speed?** It is the current speed of the fan, normalized between 0-100%, where 0% means the fan is not spinning at all and 100% means the fan is spinning at its maximum possible speed.

The fan speed varies dynamically, without user or system intervention, within the minimum and maximum allowed speeds depending on GPU temperature. Normally, the fan is not required to spin at 100% as the fan is designed to maintain operational temperature at a lower speed. Therefore, the maximum allowed fan percent speed may be less than 100%. This also increases fan life and is acoustically better.

The fan is never supposed to be at 0% for desktop systems unless the system is shut down, in Standby mode (S3), or in Hibernate mode (S4).

Every NVIDIA GPU board has its own maximum and minimum fan speed percentage values.

Users can manually override the fan percent speed via NVAPI, however, GPU functionality may be affected if the user changes the speed beyond
the minimum or maximum allowed speeds.

**What is Fan Mechanical Speed/RPM?** The fan mechanical speed is the physical measurement of the number of fan rotations per minute (RPM). The board has a dedicated tachometer which measures this value. The relationship between fan speed percent and mechanical speed is roughly linear, but the actual mapping is dependent on the physical cooler such that different boards and coolers will produce different fan mechanical speeds for the same percent speeds.

**How is Expected Fan Mechanical Speed calculated?** The expected fan mechanical speed is the desired number of fan revolutions per minute (RPM) with respect to the current fan speed percent. It’s calculated by linearly interpolating the current fan speed with minimum & maximum values of percent speed and RPM.

Below is the formula used for calculating expected RPM:

\[
ERPM = \text{MinRPM} + ((\text{CurPctSpeed} - \text{MinPctSpeed}) \times ((\text{MaxRPM} - \text{MinRPM}) / (\text{MaxPctSpeed} - \text{MinPctSpeed})))
\]

Where:
- **ERPM** > Expected fan mechanical speed (RPM)
- **CurPctSpeed** -> Current fan percent speed
- **MinPctSpeed** > Minimum fan percent speed
- **MaxPctSpeed** > Maximum fan percent speed
- **MinRPM** > Minimum RPMs with respect to Minimum fan percent speed
- **MaxRPM** > Maximum RPMs with respect to Maximum fan percent speed

**What is Allowed Error Margin?** It is the percent margin indicating the variation between electrical inputs and corresponding fan mechanical speed. Ideally mechanical speed should change linearly with electrical inputs, but based on statistical analysis, its slightly curved and is found to be in the range of 15%.

**Description of Fan events & diagnostics:**
1) **Normal**  Fan is spinning as expected and designed, within the allowed error margin.
   *Diagnostics*   Fan is in working condition, no action is required.
2) **Warning**  Fan is spinning either faster or slower than expected beyond the allowed error margin but less than 100% difference.  
*Diagnostics*  Check for any obstruction or dust and clean the fan. If fan continues to be in warning state, then fan mechanical components may be failing out.

3) **Critical**  Fan is spinning either faster or slower than expected with greater than 100% difference from expected value. The fan may be stopped.  
*Diagnostics*  Check for any obstruction or dust and clean the fan. If fan continues to be in the Critical state, mechanical components have likely failed or will fail soon and operator needs to seek replacement.

## Thermal Events and Diagnostics

**What is Thermal Slowdown?**  Thermal slowdown is a mechanism which automatically reduces clocks when temperature reaches a critical point/threshold usually around 100°C. This mechanism helps prevent thermal runaway and permanent damage to the chip/board.

**Description of Thermal events & diagnostics:**

1) **Normal**  Temperature is within normal operating ranges i.e. more than 10°C below the slowdown threshold temperature.  
*Diagnostics*  GPU is functioning normally

2) **Warning**  Temperature is within 10°C of thermal slowdown threshold temperature.  
*Diagnostics*  Check if the fan is spinning properly (verify the fan event in addition) and that automatic variable fan control is enabled. Ensure that the system has proper ventilation and system fan is running properly. In addition, any running application should be closed to see if the temperature is returning to normal level after some time. If the warning is cleared by doing this (and normal event is generated), report this behavior to administrator.

3) **Critical**  Temperature has reached thermal slowdown threshold temperature.
Diagnostics  Check if the temperature is reduced after closing the application(s) running on the system, if the temperature is reduced (and normal event is generated) then report this matter to administrator and stop using that application until its fixed. If the problem persists even without any application running on that system then immediately stop using that GPU and contact administrator for further help. Note that GPU will engage in built-in slow down mechanism, which may reduce the temperature leading to generation of warning or normal event If this happens, please report this behavior to administrator. But if the temperature keeps on rising, then build-in shutdown mechanism will trigger to prevent damage.
Using NVWMI

General Information about NVWMI Clients

NVWMI implements WMI provider interfaces and general guidelines for WMI provider usage applies. See MSDN article Using WMI.

Recommendations for selecting a language for NVWMI client implementation:

- **WinScript**
  Both VisualBasic Script and JavaScript could be used. See MSDN article Scripting API for WMI for details. Implementation complexity is low for simple scripts, but quickly rise with development. Not recommended for any complex clients.

- **PowerShell**
  Recommended way to create NVWMI clients of low to moderate complexity. See MS TechNet articles Get-WmiObject and Invoke-WmiMethod. There is a useful utility for exploring WMI objects - WMI Explorer.
  Moderate implementation complexity.

- **WMIC** - WMI Console
  WMIC is a standard utility for MS Windows since Windows XP. It has command-line interface and could operate in batch or interactive
mode. See Using NVWMI with the WMIC tool for details. Should not be used in applications. Recommended as a tool for quick testing only - see also WMIC Known limitations.

- .NET/C#/Managed C++
  Recommended for complex NVWM client applications. See MSDN article WMI .NET Overview.
  High implementation complexity.

- DCOM/C++
  Recommended for complex NVWMI client applications. See MSDN article COM API for WMI.
  Highest implementation complexity.

Profile data portability

Data of system profiles (see SystemProfile) and display profiles (see DisplayProfile) is stored as UTF-16 text files in "%ProgramData%\NVIDIA Corporation\Drs". Data of 3D profiles (see ApplicationProfile, Profile) is stored in binary format in the same location. Note that %ProgramData% is hidden directory.

NVWMI supports profile data transfer from one system to another with following limitations:

- system components are identical: MB, system BIOS, GPU boards, V-BIOS, CPU, RAM, any additional PCI-E device
- GPU boards are plugged into identical PCIe slots. Ordinal of PCIe slot and data width must match (e.g. if a system1 has Board1(Gpu1) in slot0 with 16 lanes then system2 must have Board1(Gpu1) in slot0 with 16 lanes)
- Monitors are identical and attached in same order via identical connectors
- Running identical OS and NVIDIA driver stack

Known issues
Windows 10-specific

Smart clone is not supported in Windows 10 as of version 2.27.

Windows 7-specific

Using method `DisplayManager::createDisplayGrids` may not produce desired results when layout key is specified. `Display` ordering might be off for 1xN grids, when N>3. This is OS-specific limitation. If you encounter this problem, please use following steps:

- configure desired display configuration using NVIDIA Control Panel
- save it as a set of display profiles using method `ProfileManager::saveDisplayProfiles`
- when you need to set or restore saved display configuration, use `ProfileManager::applyDisplayProfiles`

Diagnostic and troubleshooting

Controlling NVWMI Service

- In Services GUI application

  - browse running services
  - locate and select "NVIDIA WMI Provider"
  - click "Stop Service" button on the toolbar or "Stop" link in the left pane

You might pause and start the NVWMI service in a similar way.

- In command shell (CMD prompt)

To stop NVWMI service:

```plaintext
sc stop nvwmi
```

To start NVWMI service:

```plaintext
sc start nvwmi
```
To query whether NVWMI service is running:

```
sc query nvwmi
```

Logging off and logging on in interactive local session (not RDP), OS restart will restart NVWMI as well.

**Using NVWMI logging to diagnose issues**

See [Examples of Useful Logging configurations](#). Enabling errors and warnings in the OS event log is recommended to diagnose and triage hardware or driver issues against NVWMI.
NVIDIA Performance Counters

Introduction

NVWMI provides performance counters with information about the state of NVIDIA hardware in real-time. Available performance counters are described in the manifest file %System%\nvPerfProvider.man.

Performance counters are accessible using standard OS tools such as Microsoft Performance Monitor or programmatically. Microsoft supplies the Performance Monitor tool in all supported versions of the OS.

Read About Performance Counters for general outline.

For more technical details see Overview of Windows Performance Monitor.

Available NVIDIA GPU Counters

- Available Memory MB - Amount of free memory, available for GPU in megabytes
- Total Memory MB - Total amount of memory, available for GPU in megabytes
- Virtual Memory MB - Amount of virtual memory, available for GPU in megabytes
- Memory Clock MHz - On-board memory clock in megahertz
- Core Clock MHz - GPU core clock in megahertz
- % Cooler rate - GPU cooler rate in percents
- Temperature C - GPU temperature (Celsius)
- Power Consumption mW - Consumed power over sampling interval in milliwatts. Duration of a sampling interval (in ms) stored in `Gpu::powerSamplingPeriod` property
- % GPU Usage - The percentage of time where the GPU is considered busy in the last 1 second interval
- % GPU Memory Usage - The percentage of GPU Memory currently being utilized by all running applications
- Fan Speed - GPU fan speed rotations per minute
- % Bus Usage - PCI-Express Bus Usage in percents
- % Video Usage - Video Usage in percents
- % FB Usage - Frame Buffer Usage in percents
- PCI-E downstream width to GPU - Width of the PCI-Express downstream link to GPU (number of lanes)
- PCI-E current width to GPU - PCI-Express current width of link to GPU (number of lanes)
- PCI-E current speed to GPU Mbps - Current speed of the PCI-E link to GPU (Megabits per second)
- % Video Encoder usage - GPU Video Engine encoder usage in percents
- % Video Decoder usage - GPU Video Engine decoder usage in percents
- Video Encoder Sessions - Number of GPU Video Encoding Sessions
- Video Encoder Average FPS - Video Encoder frames per seconds as a trailing average
- Video Encoder Average Latency (ms) - Video Encoder average latency in milliseconds

**Working with Windows Performance Monitor**

1. Monitoring the counter graph
- Run perfmon.exe
- Browse to Performance->Monitoring Tools->Performance Monitor in the navigation bar
- Select item **Add counters** from context menu (right-click in graph area)
- Select target system (local or remote)
- Browse the list to find **NVIDIA GPU**
- Expand **NVIDIA GPU** item
- Select any counter from the list, multiple selections supported
- Select the instance of that counter from the instance list below
- Add the counter instance and press OK
- Choose appropriate graph scale and sampling rate

2. Creating a custom counter data set

- Run perfmon.exe
- Select **Data Collector Sets**
- Select **User Defined node** and bring up its context menu (by right click)
- Select **New, Data collector Set**
- Select **Create data logs, Performance counter** and click on **Next**
- Select **Add**
- Select desired counters (for example, **Processor Time**), **Add** and click on **OK**
- Select **Next**
- Browse to a folder, select **Next**
- Click on **Change** button to specify desired credentials under **Run as**
- Select **Save, Close** and click on **Finish**
- The user data collector is created
- Bring up context menu (right click on the created data set) and select **Start**
- Properties could be changed at any time via context menu (right click)
- To stop collecting data, select **Stop** in the context menu

Diagnostics and troubleshooting
When NVWMI system service is not properly installed, NVIDIA performance counters are also not available. See **Controlling NVWMI Service** for service troubleshooting. The manifest file `nvPerfProvider.man` must be properly registered and enabled.

**Manual installation**

- `C:\>lodctr /M:nvPerfProvider.man` - Registers counters by writing GUID to `[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\_V2Providers]`
- `C:\>lodctr /E:nvWmi64.exe` - Enables performance counters. For 32-bit OS substitute `nvWmi.exe`
- `sc start nvwmi` - Starts the service, counters should be available.

**Attention:**

NVWMI service is presumed to be stopped

**Manual uninstallation**

- `C:\>unlodctr /M:nvPerfProvider.man` - Removes counters GUID from `[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Perflib\_V2Providers]`
- `C:\>lodctr /D:nvWmi64.exe` - Disables performance counters. For 32-bit OS substitute `nvWmi.exe`
- `sc stop nvwmi` - Stops the service, counters will not be provided anymore.

**Attention:**

NVWMI service is presumed to be running

---

**NVIDIA**

[Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.]
Logging and tracing NVWMI activity

NVWMI equipped with robust logging and tracing capabilities. Clients can select logging types and control logging levels. Supported logging types:

- MS Windows Event log
- Debugger message stream
- File log

Note that for logging types or logging filter changes to be in effect, NVWMI service restart is required. See Controlling NVWMI Service stopping and starting NVWMI.

Logging Types

Logging types are controlled by the registry value `HKLM\SOFTWARE\NVIDIA Corporation\NVWMI\LogTypes`. It is a bitmask, several values might be combined to enable several logging types at once. Supported types are:

- Debug stream.
  Any application, capable of capturing Win32 OutputDebugString. For example - DebugView, WinDbg, Visual Studio debugger.
LogTypes=1

- File log.
  %ALLUSERSPROFILE%\nvwmi.log will be produced with records in format
  <date> <timestamp> : <message text>

LogTypes=2

- MS OS event log.
  Could be examined by Event Viewer - standard utility of MS
  Windows. Or any other application which is capable of attaching to
  the MS Windows event log. Event Viewer requires manual refresh
  (hotkey F5) to monitor events. NVWMI events are logged in
  Application log. Event source is NVWMI.

LogTypes=4

  Setting LogType to 0 will disable logging completely (silent mode).

**Configuring Logging Filter**

Logging filter is controlled by the registry value HKLM\SOFTWARE\NVIDIA
  Corporation\NVWMI\LogFilter.
  It is a bitmask, several values might be combined to filter out messages
  of several levels at once.
  LogFilter=1 - traces
  LogFilter=0x10 - errors
  LogFilter=0x100 - warnings
  LogFilter=0x1000 - function entries
  LogFilter=0x10000 - informational messages
  LogFilter=0x11111 will disable message filtering (all messages will be
  logged),
  LogFilter=0 will filter out all messages (silent).

**Examples of Useful Logging configurations**

This registry script will enable logging to OS event log. Only error
  messages will be logged, all other messages will be discarded:
Registry script to enable logging to the debug stream and OS event log. All messages will be logged (maximum verbosity):

[\HKEY_LOCAL_MACHINE\SOFTWARE\NVIDIA Corporation\NVWMI]
"LogFilter"=dword:00111111
"LogTypes"=dword:00000005

Disable all logging:

[\HKEY_LOCAL_MACHINE\SOFTWARE\NVIDIA Corporation\NVWMI]
"LogTypes"=dword:00000000
Using NVWMI with the WMIC tool

Examples of WMIC command-lines

You could set default namespace to `root\cimv2\nv` (otherwise path <object> won't work) or specify `/namespace:nv` at the command line:

```powershell
wmic:root\cli>/namespace:nv
```

List all **System** singleton properties:

```powershell
wmic:root\cli>path System get *
```

Default verb is `get`, it could be skipped, so command

```powershell
wmic:root\cli>path System
```

will produces same output.

List several properties for all GPUs (see **Gpu**) in a system - in this example there are 2 Quadro K5000 GPUs:

```powershell
wmic:root\cli>path Gpu get id,nvapiId,name,uname
```

Produces this output:

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>nvapiId</th>
<th>uname</th>
</tr>
</thead>
</table>
```
List several properties for a given GPU (see **Gpu**) in a system - in this example GPU is identified by id, power consumption and current GPU load is listed (with several 3D applications running):

```
wmic:root\cli>path Gpu where id=1 get power, percentGpuUsage
percentGpuUsage   power
12                44.797
```

**Invoking WMI methods in WMIC**

Be aware of a difference between singleton class and classes with dynamic instances as it affects the way WMI methods are called in general. See **static vs. dynamic instances** in the Introduction. Calling methods of the **System** class (singleton class, other singleton classes are **DisplayManager** and **ProfileManager**) without any parameters:

```
wmic /namespace:nv path System call info
```

Calling methods of the **ProfileManager** class (singleton) with input parameter (in this example - global profile name is "Autodesk Mudbox - compatible"): 

```
wmic /namespace:nv path ProfileManager call setCurrentProfile3D "Aut
```

Calling methods of any other class (non-singleton) - general format:

```
wmic /namespace:nv path <class name> where <instance selection claus
```

Calling methods of any other class (non-singleton) - example of calling **Gpu::info**:

```
wmic /namespace:nv path Gpu where ordinal=1 call info
```

Getting some properties

```
wmic /namespace:nv path <class name> where <instance selection claus
```

For example, listing just uname and id property of all **Gpu** instances (note that where clause is skipped as it is query for all instances)
wmic /namespace:nv path Gpu get uname,id

Note that `where` clause is a system-dependent. Example demonstrates calling `Gpu::info` on a system with Quadro 5000 GPU:

wmic /namespace:nv path gpu where name="Quadro 5000" call info

Calling a method of MOF class with dynamic instances and input parameters is performed in similar way. Example demonstrates changing scaling mode to produce image centered on a screen without any scaling (scaling mode value=3) for the instance of the `DisplayGrid` class with unique name "DisplayGrid (1 of 4)"

wmic /namespace:nv path DisplayGrid where uname="DisplayGrid (1 of 4)" call setScaling 3

The following is an example of calling a method with multiple input parameters using 'named parameters' convention.

wmic /namespace:nv path DisplayManager call createDisplayGridById rows=2 cols=1 displayIds=(1001,1002)

The following is an example of creating a desktop using `Profile` class. String expected for creating desktop is "DesktopName;PerMonitorBackground;WallpaperPath;PerMonitorStyle;" where PerMonitorBackground=1 means to have different desktop background per display (0 means to have same background across all displays), PerMonitorStyle means how background appears on desktop (0 = Center, 1 = Tile, 2 = Stretch)

For 2 display and different background per display the input string is "DesktopName;PerMonitorBackground;WallpaperPath for Display1,PerMonitorStyle for Display1;WallpaperPath for Display2,PerMonitorStyle for Display2" and so on for other displays

This example is for a single display and creates a desktop named "Test" with Style as Tile

wmic /namespace:nv path profile where id=1465693279 call setStringValueById settingId=1500598283 value="Test;1;C:\Windows\Web\Wallpaper\Architecture\img13.jpg,1"

where id is retrieved by querying profile id that corresponds to nView Global
This example is for a single display and modifies the image for a desktop named "Test" and changes Style to Stretch

```
wmic /namespace:nv path profile where id=1465693279 call setStringValueById
```

Any property could be used to select instance (or multiple instances at once). However, due to WMIC limitations it is not possible to specify numeric values with negative values in a shell command-line, as WMIC treats '-' as an argument separator. See [WMIC Known limitations](#).

**How to create new 3D application profile in WMIC**

The following is an example of creating 3D application profile and modifying its settings in WMIC. "My profile" used as an example of profile name for executable "myApp.exe". **Setting** "OpenGL video-editing mode" used as an example of typical 3D setting. For a list of all available profile settings see

Create new application profile by invoking

**ProfileManager::createProfile** method

```
wmic /namespace:nv path ProfileManager call createProfile name="My profile" type=0	params="myApp.exe"
```

Retrieve setting ID from setting name by invoking

**SettingTable::getIdFromName** method

```
wmic /namespace:nv path SettingTable where type=0 call getIdFromName
```

Change setting value "OpenGL video-editing mode" (id=552469172) to 1 (e.g. "enabled") by calling **ApplicationProfile::setValueById**. For a list of all available profile settings see [Profile settings available in NVWMI](#).

```
wmic /namespace:nv path ApplicationProfile where name="My profile" call setValueById
```

Query current settings

```
wmic /namespace:nv path ApplicationProfile where name="My profile" call getsettings
```

In output you should see a line
Specifying arrays in WMIC

Some methods take arrays as input parameters. Arrays in WMIC are enclosed in braces () and individual elements are separated by comma. **WARNING**: Usage of any other separator (e.g. space) will lead to undefined results.

Specifying multiple executables in the array of strings for `ApplicationProfile::addApplications` and `ApplicationProfile::removeApplications`:

```
wmic /namespace:nv path ApplicationProfile where name="Adobe Photosh ...
```

Specifying multiple display grids in the array of strings for `DisplayManager::validateDisplayGrids` and `DisplayManager::createDisplayGrids`:

```
4 displays, connected to one GPU, windows desktop extended on all 4:
wmic /namespace:nv path DisplayManager call validateDisplayGrids grids=
wmic /namespace:nv path DisplayManager call createDisplayGrids grids=
```

```
4 displays, connected to one GPU, 2 Mosaics 1x2 in 1920x1080, 32-bit pixels at 60 Hz:
wmic /namespace:nv path DisplayManager call validateDisplayGrids grids=
wmic /namespace:nv path DisplayManager call createDisplayGrids grids=
```

**WMIC Common tasks**

Examples of WMIC command lines for several common tasks are given below.

**Determine Video BIOS version**

Determine Video BIOS version of all GPUs in the system:
wmic /namespace:nv path Gpu get verVBIOS

Determine Video BIOS version of the specific GPU:

wmic /namespace:nv path Gpu where id=1 get verVBIOS

Sample output - 70.0.30.0.a is a string representation of the video BIOS version, **Version::strValue** :

```
{} 3 Version.orderedValue=10 Version Version 2 10 70.0
```

Note that GPU could be selected by operator WHERE by any property (e.g. name, uname, id etc.). If a system has single GPU, WHERE operator could be skipped for properties, but it is required for calling a method. Alternative way of obtaining human-readable information about given GPU using **Gpu::info()** method:

wmic /namespace:nv path Gpu where id=1 call info

Search for "Video BIOS version:" in the output.

**Determine Display Driver version**

Similar approach applies to the common task of determining the display driver version. Following WMIC commands:

wmic /namespace:nv path System get verDisplayDriver

Or

wmic /namespace:nv path System call info

Will produce display driver version in human-readable format. Also note that output of **System::info** method will contain exact Video BIOS version string in case when all GPUs in a system are running the same version. Otherwise Video BIOS version will be reported as "mixed".

**Determine Current Display Mode**

Query properties of class **DisplayGrid** to determine current display mode settings, such as resolution, color depth, refresh frequency:
**DisplayGrid::displayModeVirtual** contains "effective" screen dimensions, which are taking into account overlaps between individual physical displays:

```sql
wmic /namespace:nv path DisplayGrid get displayModeVirtual
```

**DisplayGrid::displayModePhysical** contains "raw" physical screen dimensions of individual physical display:

```sql
wmic /namespace:nv path DisplayGrid get displayModePhysical
```

When only one display is active or multiple displays are not combined into display grids, **DisplayGrid::displayModeVirtual** has identical values to the **DisplayGrid::displayModePhysical**.

```sql
wmic /namespace:nv path DisplayGrid where id=1 call info
```

In the output, "current grid mode:" corresponds to the **DisplayGrid::displayModePhysical** and "effective grid mode:" corresponds to the **DisplayGrid::displayModeVirtual**.

**Change Current Display mode**

Example of invoking method **DisplayGrid::setDisplayMode**:

```sql
wmic /namespace:nv path DisplayGrid where id=1 call setDisplayMode
```

**Cloning a Display**

Cloning is useful for displaying identical images on several displays. It could be achieved in NVWMI by creating a **DisplayGrid** and overlapping displays at display dimensions.

Example of horizontally cloned displays (assuming 2 displays attached to the same GPU, both in display mode with screen width 1920):

- create **DisplayGrid 1x2**, physical display mode is 1920x1080, 32 bits per pixel, at 60 Hz refresh rate

  ```sql
  wmic /namespace:nv path DisplayManager call createDisplayGrids
  ```
- set columnar overlap at display width

  `wmic /namespace:nv path DisplayGrid where id=1 call setOverlapCc`

Note that resolution of displays might differ. When resolutions are different, result will be either clone-to-fit or clone-to-pan. Overlap value must match exactly a dimension (width for horizontal clone, height - for vertical clone) for one of displays.

**WMIC Useful options**

- execute WMIC command on remote system:

  `/node:<system name>`

  Note that if the system name contains a hyphen (-) then it must be escaped or the system name enclosed in single quotes (`'`):

  `wmic /node:'wks-1' /namespace:nv path gpu`

  will get all instances of the **Gpu** class on the system with Windows name **wks-1**
- trace execution of the WMI command

  `/trace:on`
- record WMIC commands including their output with timestamps to an XML file

  `/record:<file path>`
- select format of the WMI command output

  `/format:<format specifier>`

  Specifies a key word or XSL file name to format the data. Command

  `wmic process list /format /?`

  will list all available formats.
- list all properties with their names when using GET * verb
wmic /namespace:nv path <classname> get * \value

See also [WMIC documentation on MSDN](https://msdn.microsoft.com/en-us/library/windows/desktop/hh851772(v=vs.85).aspx)

**WMIC Known limitations**

- WMIC doesn't allow specifying negative numbers. A minus character ('-') is treated as a switch. This prevents some legitimate commands from producing correct output. A slash ('/') and backslash ('\') are also special characters and should be either escaped or a string with these characters should be quoted literally (enclosed in apostrophes '). For example, be sure to quote computer name in `node:` parameter if it contains any special characters.
- WMIC is case-insensitive for class names and properties.
- WMIC doesn't display properties which contain array of object references (for example, `Board::gpus`).

---

[NVIDIA](https://www.nvidia.com)

*Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.*
Using NVWMI with the PowerShell

Examples of PowerShell command-lines

PowerShell has cmdlet Get-WmiObject to directly access and manipulate WMI objects. See MSDN article [Using the Get-WmiObject Cmdlet](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/get-wmiobject). GWMI and gwmi are recognized as built-in aliases for Get-WmiObject.

Other helpful cmdlets to examine WMI objects are Get-member, Select-object, Sort-object and Where-object. For more tips of WMI scripting in PowerShell see Microsoft's TechNet article [Scripting for WMI](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/gwmi).

Listing **Gpu::id** property for all **Gpu** instances (note that namespace is case-insensitive):

Get-WmiObject -namespace "root\cimv2\nv" -query "select id from Gpu"

WMI methods invocation:

```
$instance=gwmi -namespace "root\cimv2\nv" -class system
$result=$instance.invokeMethod("info",$null,$null) # $result will be an instance of class __PARAMETERS
$result # print a result
```

Alternate way of accessing WMI objects - retrieving an instance of the **System** singleton, calling method **System::info()** and storing result:
$instance = [wmiclass]"\\.root\cimv2\nv:system"
$result=$instance.invokeMethod("info",$null) # note that invokeMethod
$result # print a result

Shorter version:

([wmiclass]"\\.root\cimv2\nv:system").invokeMethod("info",$null)

Note that it works only for singleton classes (System, DisplayManager, ProfileManager).
Running WQL query which selects certain property from WMI object(s).

$qr = Get-WmiObject -namespace "root\cimv2\nv" -Query "select uname
foreach ($t in $qr) {$t.uname}

In case of more than one GPUs $qr will contain more than one element.

Selecting all instances and then iterating:

$gpus = Get-WmiObject -namespace "root\cimv2\nv" -class gpu
foreach($o in $gpus) # obtain an instance
{
    # print some properties
    "id="+$o.id+" uname="+$o.uname+" GPU clock="+$o.gpuCoreClockCurr
    # invoke info method
    $o.invokeMethod("info",$null)
}

Using profile settings reflection mechanism

Querying all setting names of a given profile type (0 - 3D application profile):

$qr = Get-WmiObject -namespace "root\cimv2\nv" -Query "select settin
$qr.settingNames[0] # print a name of a setting, $qr.settingNames is

Querying all setting IDs of a given profile type (0 - 3D application profile):

$qr = Get-WmiObject -namespace "root\cimv2\nv" -Query "select settin
$qr.settingIds[0] # print an ID of a setting, $qr.settingIds is an a

Obtaining brief information about a profile setting (3D setting with
id=545898348 is "Antialiasing - Line gamma")
$table=gwmi -namespace "root\cimv2\nv" -class SettingTable -filter "
$table.invokeMethod("infoById",545898348) # get information about th
$table.invokeMethod("infoByName","Antialiasing - Line gamma") # get

Creating multiple display grids

Creating 4 display grids, single display per each, 4 displays attached to a single GPU

$dm = Get-WmiObject -namespace "root\cimv2\nv" -Class DisplayManager
[array]$grids_4_1x1 = @("rows=1;cols=1;layout=1.1","rows=1;cols=1;layout=1.2")
$res = Invoke-WmiMethod -Path $dm.__PATH -Name createDisplayGrids -ArgumentList ($grids_4_1x1)
if($res.ReturnValue -eq $true) {"grids are valid"} else {"grids are invalid"}

Creating 1 display grid from 4 displays, 2 displays per column and 2 displays per row, 4 displays attached to a single GPU

$dm = Get-WmiObject -namespace "root\cimv2\nv" -Class DisplayManager
[array]$grids_1_2x2 = @("rows=2;cols=2;layout=1.1 1.2 1.3 1.4")
$res = Invoke-WmiMethod -Path $dm.__PATH -Name createDisplayGrids -ArgumentList ($grids_1_2x2)
if($res.ReturnValue -eq $true) {"success"} else {"failure"}
### Profile settings available in NVWMI

#### OpenGL Settings for 3D profiles

<table>
<thead>
<tr>
<th>Setting ID (32-bit hex)</th>
<th>Setting ID (32-bit decimal)</th>
<th>Type</th>
<th>Type of values</th>
<th>Setting name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2089BF6C</td>
<td>545898348</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Antialiasing - Line gamma</td>
<td>N/A</td>
</tr>
<tr>
<td>0x2097C2F6</td>
<td>546816758</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Deep color for 3D applications</td>
<td>N/A</td>
</tr>
<tr>
<td>0x206A6582</td>
<td>543843714</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>OpenGL default swap interval</td>
<td>Controls number of V-blank signals from display to wait before rendering a frame (SwapInterval) in OpenGL.</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Precision</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>---------------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>0x206C4581</td>
<td>543966593</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>OpenGL default swap interval fraction</td>
<td></td>
</tr>
<tr>
<td>0x20655CFA</td>
<td>543513850</td>
<td>32-bit</td>
<td>boolean</td>
<td>OpenGL default swap interval sign</td>
<td></td>
</tr>
<tr>
<td>0x209DF23E</td>
<td>547222078</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Event Log Severity Threshold</td>
<td></td>
</tr>
<tr>
<td>0x20FF7493</td>
<td>553612435</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Extension String version</td>
<td></td>
</tr>
<tr>
<td>0x201F619F</td>
<td>538927519</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Buffer-flipping mode</td>
<td></td>
</tr>
</tbody>
</table>

To force VSYNC ON or OFF, use VSYNCMODE 0x206C4581.

Controls how the current scan line is evaluated for a (un)synced flip with negative intervals. A value in the range of 0-100%

Swap intervals are treated as negative or positive values depending on value (0-positive, 1-negative).

This setting specifies the severity of events logged into the Windows Event Log by the OpenGL driver.

N/A
<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x204D9A0C</td>
<td>541956620</td>
<td>samples</td>
<td>unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Force Stereo shuttering</td>
<td>N/A</td>
</tr>
<tr>
<td>0x20D0F3E6</td>
<td>550564838</td>
<td>samples</td>
<td>UTF-16 string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Preferred OpenGL GPU</td>
<td>Controls the default render GPU. The string is set by the control panel and not meant to be set directly by the user. The string is always overridden by the default render GPU.</td>
</tr>
<tr>
<td>0x208E55E3</td>
<td>546199011</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum frames allowed</td>
<td>N/A</td>
</tr>
<tr>
<td>0x209AE66F</td>
<td>547022447</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exported Overlay pixel types</td>
<td>N/A</td>
</tr>
<tr>
<td>0x206C28C4</td>
<td>543959236</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enable overlay</td>
<td>N/A</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x20797D6C</td>
<td>544832876</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>High level control of the rendering quality on OpenGL</td>
</tr>
<tr>
<td>0x20A29055</td>
<td>547524693</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unified back/depth buffer</td>
</tr>
<tr>
<td>0x2092D3BE</td>
<td>546493374</td>
<td>bitfields</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Enable NV_gpu_multicast extension</td>
</tr>
<tr>
<td>0x20C1221E</td>
<td>549528094</td>
<td>bitfields</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Threaded optimization</td>
</tr>
<tr>
<td>0x202888C1</td>
<td>539527361</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Event Log Tmon Severity Threshold</td>
</tr>
<tr>
<td>0x20FDD1F9</td>
<td>553505273</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Triple buffering</td>
</tr>
</tbody>
</table>
### Shared Direct3D and OpenGL Settings for 3D profiles

<table>
<thead>
<tr>
<th>Setting ID (32-bit hex)</th>
<th>Setting ID (32-bit decimal)</th>
<th>Type</th>
<th>Type of values</th>
<th>Setting name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10ECDB82</td>
<td>283958146</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Antialiasing - Behavior Flags</td>
<td>Flags for altering how the driver interprets 'Antialiasing - Setting'</td>
</tr>
<tr>
<td>0x10FC2D9C</td>
<td>284962204</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Antialiasing - Transparency Multisampling</td>
<td>N/A</td>
</tr>
<tr>
<td>Address</td>
<td>Memory Size</td>
<td>Data Type</td>
<td>Antialiasing Method</td>
<td>Controls</td>
<td>Antialiasing Number of Samples</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------------</td>
<td>----------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>0x107D39D</td>
<td>276652957</td>
<td>32-bit unsigned integer</td>
<td>N/A</td>
<td>Antialiasing - Gamma correction</td>
<td></td>
</tr>
<tr>
<td>0x10D773D2</td>
<td>282555346</td>
<td>32-bit unsigned integer</td>
<td>Setting</td>
<td>Controls method and number of antialiasing</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Antialiasing - Transparency Supersampling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------------------</td>
<td>------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10D4A85</td>
<td>282364549</td>
<td>32-bit unsigned integer</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x107EFC5B</td>
<td>276757595</td>
<td>32-bit unsigned integer</td>
<td>Antialiasing - Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x107AF5CB</td>
<td>276495451</td>
<td>boolean</td>
<td>Antialiasing - SLI AA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101E61A9</td>
<td>270426537</td>
<td>32-bit unsigned integer</td>
<td>Controls number of samples</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of anisotropic filtering</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Anisotropic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Filtering Mode</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>------------------</td>
<td>-----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x10D2BB16</td>
<td>282245910</td>
<td>unsigned integer</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1035DB89</td>
<td>271965065</td>
<td>boolean</td>
<td>NVIDIA Predefined Ansel Usage</td>
<td>Empowers an app profile to disallow Ansel</td>
<td></td>
</tr>
<tr>
<td>0x1075D972</td>
<td>276158834</td>
<td>boolean</td>
<td>Enable Ansel</td>
<td>Toggle Ansel on or off</td>
<td></td>
</tr>
<tr>
<td>0x1085DA8A</td>
<td>277207690</td>
<td>boolean</td>
<td>Ansel flags for enabled applications</td>
<td>Temporary whitelisting of apps allowed to enable Ansel</td>
<td></td>
</tr>
<tr>
<td>0x104554B6</td>
<td>272979126</td>
<td>32-bit unsigned integer</td>
<td>Application Profile Notification Popup Timeout</td>
<td>This setting specifies how many seconds the popup displaying information about which profile is being applied should appear. Zero counts as disabled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Steam Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x107CDDBC</td>
<td>276618684</td>
<td>32-bit unsigned integer</td>
<td>Steam Application ID is used to identify which Steam applications are installed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10115C89</td>
<td>269573257</td>
<td>32-bit unsigned integer</td>
<td>Battery Boost enables the Battery Boost functionality, caps FPS for DC mode only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x106D5CFF</td>
<td>275602687</td>
<td>boolean</td>
<td>This setting indicates to the Control Panel that a given profile should not be displayed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10354FF8</td>
<td>271929336</td>
<td>UTF-16 string</td>
<td>List of Universal GPU ids, item separator is ';'. Exposed in UI as 'CUDA - GPUs'.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 0x10D1EF29  | 282193705 | UTF-16 string | Maximum GPU Power is the maximum power that the GPU is capable of.
<table>
<thead>
<tr>
<th>Address</th>
<th>Samples</th>
<th>Type</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x108F0841</td>
<td>277809217</td>
<td>samples</td>
<td>boolean Export Performance Counters</td>
<td>N/A</td>
</tr>
<tr>
<td>0x1034CB89</td>
<td>271895433</td>
<td>samples</td>
<td>boolean NVIDIA Predefined FXAA Usage</td>
<td>Empowers an app profile to disallow FXAA</td>
</tr>
<tr>
<td>0x1074C972</td>
<td>276089202</td>
<td>samples</td>
<td>boolean Enable FXAA</td>
<td>Toggle FXAA on or off</td>
</tr>
<tr>
<td>0x1068FB9C</td>
<td>275315612</td>
<td>samples</td>
<td>boolean Enable FXAA Indicator</td>
<td>Toggle FXAA Indicator or or off</td>
</tr>
<tr>
<td>0x10287051</td>
<td>271085649</td>
<td>samples</td>
<td>boolean SLI indicator</td>
<td>Show or hide the SLI on-screen indicator</td>
</tr>
<tr>
<td>0x10444444</td>
<td>272909380</td>
<td>samples</td>
<td>boolean NVIDIA Quality upscaling</td>
<td>Toggle NVIDIA Quality upscaling on or off</td>
</tr>
<tr>
<td>0x10F9DC83</td>
<td>284810371</td>
<td>range</td>
<td>32-bit unsigned integer Maximum AA samples allowed for a given application</td>
<td>Maximum AA we are going to allow for a given application</td>
</tr>
<tr>
<td>0x1094F16F</td>
<td>278196591</td>
<td>samples</td>
<td>boolean Display the PhysX indicator</td>
<td>N/A</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Field Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x1057EB71</td>
<td>274197361</td>
<td>range</td>
<td>Power management mode</td>
<td></td>
</tr>
<tr>
<td>0x103BCCB5</td>
<td>272354485</td>
<td>samples</td>
<td>No override of Anisotropic filtering</td>
<td></td>
</tr>
<tr>
<td>0x10834FEE</td>
<td>277041134</td>
<td>bitfields</td>
<td>Frame Rate Limiter</td>
<td></td>
</tr>
</tbody>
</table>

- This setting tells our UI that it cannot override Anisotropic filtering for this application.
- Framerate Limiter parameters, bit 31 to enable and LOWBYTE set to frames/sec.
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Bitfields</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10834FF</td>
<td>277041151</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Frame Rate Limiter 2 Control</td>
</tr>
<tr>
<td>0x10834F0</td>
<td>277040897</td>
<td>32-bit</td>
<td>unsigned integer</td>
<td>Frame Rate Monitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FRM parameters (deprecated)</td>
</tr>
<tr>
<td>Offset</td>
<td>Value</td>
<td>Description</td>
<td>Frame Rate Monitor Control Parameters</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-----------------------------</td>
<td>----------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x1083F05</td>
<td>277040901</td>
<td>32-bit unsigned integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10F9DC82</td>
<td>284810370</td>
<td>32-bit unsigned integer</td>
<td>Maximum resolution allowed for a given application</td>
<td></td>
</tr>
<tr>
<td>0x10F9DC80</td>
<td>284810368</td>
<td>32-bit unsigned integer</td>
<td>Optimus flags for enabled applications</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Description</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>------------------------------------------</td>
<td>----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x10F9DC81</td>
<td>284810369 bitfields</td>
<td>32-bit unsigned integer</td>
<td>Enable application for Optimus</td>
<td></td>
</tr>
<tr>
<td>0x10F9DC84</td>
<td>284810372 bitfields</td>
<td>32-bit unsigned integer</td>
<td>Shim Rendering Mode Options for Optimus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rendering Mode Options for shim layer per application</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x1033DCD1</td>
<td>271834321</td>
<td>32-bit unsigned integer</td>
<td>Number of GPUs to use on SLI rendering mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>User visible exposed number of GPUs to use on SLI</td>
<td></td>
</tr>
<tr>
<td>0x1033DCD2</td>
<td>271834322</td>
<td>32-bit unsigned integer</td>
<td>NVIDIA predefined number of GPUs to use on SLI rendering mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Setting to indicate in the Control Panel how many GPUs to use by default on this SLI profile</td>
<td></td>
</tr>
<tr>
<td>0x1033DCD3</td>
<td>271834323</td>
<td>32-bit unsigned integer</td>
<td>NVIDIA predefined number of GPUs to use on SLI rendering mode on DirectX 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Setting to indicate in the Control Panel how many GPUs to use by default on this SLI profile on DirectX10</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Count</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x1033CEC1</td>
<td>271830721</td>
<td>samples</td>
<td>NVIDIA predefined SLI mode indicate in the Control Panel which SLI mode is active by default</td>
<td></td>
</tr>
<tr>
<td>0x1033CEC2</td>
<td>271830722</td>
<td>samples</td>
<td>NVIDIA predefined SLI mode on DirectX 10 Setting to indicate in the Control Panel which SLI mode is active by default on DirectX 10</td>
<td></td>
</tr>
<tr>
<td>0x1033CED1</td>
<td>271830737</td>
<td>samples</td>
<td>SLI rendering mode User visible exposed SLI Modes</td>
<td></td>
</tr>
<tr>
<td>0x10111133</td>
<td>269553971</td>
<td>range</td>
<td>Virtual Reality pre-rendered frames N/A</td>
<td></td>
</tr>
<tr>
<td>0x1094F157</td>
<td>278196567</td>
<td>samples</td>
<td>Toggle the VRR global feature N/A</td>
<td></td>
</tr>
<tr>
<td>0x1095F16F</td>
<td>278262127</td>
<td>samples</td>
<td>Display the VRR Overlay Indicator N/A</td>
<td></td>
</tr>
<tr>
<td>0x1094F1F7</td>
<td>278196727</td>
<td>samples</td>
<td>VRR requested N/A</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Samples</td>
<td>Type</td>
<td>State</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x10A879CF</td>
<td>279476687</td>
<td>integer</td>
<td>state</td>
<td>Profile-specific override. Overrides the VRR_MOD global setting</td>
</tr>
<tr>
<td>0x10A879AC</td>
<td>279476652</td>
<td>integer</td>
<td>state</td>
<td>Profile-specific override. Overrides the VRR_MOD global setting</td>
</tr>
<tr>
<td>0x1194F158</td>
<td>294973784</td>
<td>integer</td>
<td>state</td>
<td>Enable G-SYNC globally</td>
</tr>
<tr>
<td>0x101AE763</td>
<td>270198627</td>
<td>boolean</td>
<td>state</td>
<td>Key to control smooth AFR behavior</td>
</tr>
<tr>
<td>0x10A879CE</td>
<td>279476686</td>
<td>integer</td>
<td>state</td>
<td>Controls enabling or disabling VRR for OGL or D3I</td>
</tr>
<tr>
<td>0x10FDEC23</td>
<td>285076515</td>
<td>integer</td>
<td>state</td>
<td>Flags for altering how the driver interprets VSYNC</td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x11AE435C</td>
<td>296633180</td>
<td>32-bit unsigned integer</td>
<td>Stereo - Swap eyes</td>
<td></td>
</tr>
<tr>
<td>0x11E91A61</td>
<td>300489313</td>
<td>32-bit unsigned integer</td>
<td>Stereo - Display mode</td>
<td></td>
</tr>
</tbody>
</table>

Swaps image for the left eye with image for the right eye.

Display mode to use when stereo is enabled.
<table>
<thead>
<tr>
<th>Address</th>
<th>Samples</th>
<th>Format</th>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1112233</td>
<td>2863539</td>
<td>32-bit unsigned integer</td>
<td>Memory Allocation Policy</td>
<td>N/A</td>
</tr>
<tr>
<td>0x112493BD</td>
<td>28760789</td>
<td>32-bit unsigned integer</td>
<td>Stereo - Dongle Support</td>
<td>Control of the stereo dongle</td>
</tr>
<tr>
<td>0x11AA9E99</td>
<td>29634393</td>
<td>32-bit unsigned integer</td>
<td>Stereo - Enable</td>
<td>Support of the stereo API for workstation</td>
</tr>
<tr>
<td>0x11333333</td>
<td>28856115</td>
<td>32-bit unsigned integer</td>
<td>Stereo swap mode</td>
<td>N/A</td>
</tr>
</tbody>
</table>
## System Profile Settings

<table>
<thead>
<tr>
<th>Setting ID (32-bit hex)</th>
<th>Setting ID (32-bit decimal)</th>
<th>Type</th>
<th>Type of values</th>
<th>Setting name</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5ABCCB6D</td>
<td>1522322285</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Name of display profile</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>0x5A362416</td>
<td>1513497622</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Name of global 3D profile</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>0x5A22291F</td>
<td>1512188191</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Name of nView profile</td>
<td>N/A</td>
<td>-</td>
</tr>
</tbody>
</table>
### Direct3D Settings for 3D profiles

<table>
<thead>
<tr>
<th>Setting ID (32-bit hex)</th>
<th>Setting ID (32-bit decimal)</th>
<th>Type</th>
<th>Type of values</th>
<th>Setting name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00667329</td>
<td>6714153</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Ambient Occlusion</td>
<td>Ambient occlusion mode</td>
</tr>
<tr>
<td>0x00664339</td>
<td>6701881</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>NVIDIA Predefined Ambient Occlusion Usage</td>
<td>Setting provided to CPL to indicate whether the NVIDIA driver uses Ambient Occlusion or not</td>
</tr>
<tr>
<td>0x00638E8F</td>
<td>6524559</td>
<td>samples</td>
<td>boolean</td>
<td>Texture filtering - Driver Controlled LOD Bias</td>
<td>Ignored if LODBIASADJ is set</td>
</tr>
<tr>
<td>0x00041807</td>
<td>268295</td>
<td>samples</td>
<td>boolean</td>
<td>Enable GTX950 specific features</td>
<td>Enable GTX950 specific features. Currently used to enable FRM for GTX950</td>
</tr>
<tr>
<td>0x00B65E72</td>
<td>11951730</td>
<td>samples</td>
<td>boolean</td>
<td>Export Performance Counters for DX9 only</td>
<td>N/A</td>
</tr>
<tr>
<td>0x00DB1337</td>
<td>14357303</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>ICafe Settings</td>
<td>Configure Script for ICafe Logo display in XP (Dx9 only).</td>
</tr>
<tr>
<td>0x0078E8F</td>
<td>7573135</td>
<td>range</td>
<td>32-bit unsigned integer</td>
<td>Texture filtering - LOD Bias</td>
<td>N/A</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0x0098C1AC</td>
<td>10011052</td>
<td>samples</td>
<td>boolean</td>
<td>Enable sample interleaving (MFAA)</td>
<td>When enabled the driver will change the sample pattern position each frame or each refresh cycle to emulate a higher quality of antialiasing</td>
</tr>
<tr>
<td>0x007BA09E</td>
<td>8102046</td>
<td>range</td>
<td>32-bit unsigned integer</td>
<td>Maximum pre-rendered frames</td>
<td>Set the maximum number of presents that can be queued before CPU blocks.</td>
</tr>
<tr>
<td>0x00198FFF</td>
<td>1675263</td>
<td>samples</td>
<td>boolean</td>
<td>Shader Cache</td>
<td>Enables/Disables strategy</td>
</tr>
<tr>
<td>0x00E73211</td>
<td>15151633</td>
<td>samples</td>
<td>boolean</td>
<td>Texture filtering - Anisotropic sample optimization</td>
<td>N/A</td>
</tr>
<tr>
<td>0x0084CD70</td>
<td>8703344</td>
<td>samples</td>
<td>boolean</td>
<td>Texture filtering - Anisotropic filter optimization</td>
<td>N/A</td>
</tr>
<tr>
<td>0x002ECF2</td>
<td>3066610</td>
<td>samples</td>
<td>boolean</td>
<td>Texture filtering - Trilinear optimization</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Texture</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Samples</td>
<td>Type</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>0x001BB68</td>
<td>1686376</td>
<td>boolean</td>
<td>filtering - Negative LOD bias</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0x00CE2691</td>
<td>13510289</td>
<td>32-bit unsigned integer</td>
<td>Texture filtering - Quality</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0x0064B541</td>
<td>6600001</td>
<td>32-bit unsigned integer</td>
<td>Preferred refresh rate</td>
<td>Enables refresh rate override for digital monitors that support 110Hz/120Hz</td>
<td></td>
</tr>
<tr>
<td>0x00AE785C</td>
<td>11434076</td>
<td>32-bit unsigned integer</td>
<td>PowerThrottle</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>0x00AB8687</td>
<td>11241095</td>
<td>32-bit unsigned integer</td>
<td>VAB Default Data</td>
<td>This regkey defines the value set in the VAB</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Samples</td>
<td>Type</td>
<td>Vertical Sync</td>
<td>Tear Control</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>--------------------</td>
<td>------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>0x00A897CF</td>
<td>11041231</td>
<td>32-bit unsigned integer</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x005A375C</td>
<td>5912412</td>
<td>32-bit unsigned integer</td>
<td>Vertical Sync</td>
<td>Tear Control</td>
<td></td>
</tr>
</tbody>
</table>
|           |           |                    |                  | Tear control: if enabled and vsync is forced on, framerates clamped to 60 (30/20/15) hz, rates lower than that run with vsync disabled.
## Display Profile Settings

<table>
<thead>
<tr>
<th>Setting ID (32-bit hex)</th>
<th>Setting ID (32-bit decimal)</th>
<th>Type</th>
<th>Type of values</th>
<th>Setting name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5818A91D</td>
<td>1478011165</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>cloneGroupIds</td>
<td>Clone group IDs in a grid</td>
</tr>
<tr>
<td>0x58E21BD4</td>
<td>1491213268</td>
<td>range</td>
<td>32-bit unsigned integer</td>
<td>cols</td>
<td>Number of columns in a display grid</td>
</tr>
<tr>
<td>0x5822D5EE</td>
<td>1478677998</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>gridCSC</td>
<td>File path to Color Space Conversion data for a display grid</td>
</tr>
<tr>
<td>0x58D3388F</td>
<td>1490237583</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>gridGammaRamp</td>
<td>File path to Gamma Ramp data for a display grid</td>
</tr>
<tr>
<td>0x58BA728B</td>
<td>1488614027</td>
<td>range</td>
<td>32-bit unsigned integer</td>
<td>positionCol</td>
<td>Display GDI column position (in pixels)</td>
</tr>
<tr>
<td>0x584B70FE</td>
<td>1481339134</td>
<td>range</td>
<td>32-bit unsigned integer</td>
<td>positionRow</td>
<td>Display GDI row position (in pixels)</td>
</tr>
<tr>
<td>Offset</td>
<td>Length</td>
<td>Type</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x58D4B0B4</td>
<td>1490333876</td>
<td>samples</td>
<td>gridScaling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x58B21E43</td>
<td>1488068163</td>
<td>samples</td>
<td>displayIds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x58C7B07C</td>
<td>1489481852</td>
<td>samples</td>
<td>displayMode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x58DECFA8</td>
<td>1490997160</td>
<td>samples</td>
<td>rotation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5822918D</td>
<td>1478660493</td>
<td>range</td>
<td>rows</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Number of rows in display</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTF-16</td>
<td>Distance between displays</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>string</td>
<td>Positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayMode</td>
<td>Mode of display</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rotation</td>
<td>Per-disp rotation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Number of rows in display</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTF-16</td>
<td>Distance between displays</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>string</td>
<td>Positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x58EB619D</td>
<td>1491820957</td>
<td>samples string</td>
<td>overlapCols</td>
<td>number indicates overlap, negative gap (in pixels)</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x58DD36C1</td>
<td>1490892481</td>
<td>samples UTF-16 string</td>
<td>overlapRows</td>
<td>Distance between displays per row. Positive number indicates overlap, negative gap (in pixels)</td>
<td></td>
</tr>
<tr>
<td>Setting ID (32-bit hex)</td>
<td>Setting ID (32-bit decimal)</td>
<td>Type</td>
<td>Type of values</td>
<td>Setting name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>----------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x596265E8</td>
<td>1499620840</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Active Desktop Name</td>
<td>The name of the active desktop</td>
</tr>
<tr>
<td>0x5971500B</td>
<td>1500598283</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Add a desktop</td>
<td>Add a new desktop with specified properties. Format is desktop name;per-monitor flag;path to wallpaper image;wallpaper option. Per-monitor flag could be 0 or 1, wallpaper options are 0 - center, 1 - tile, 2 - stretch. Example: MyDesktop;1;wallpaper.jpg,3</td>
</tr>
<tr>
<td>0x59AFFA30</td>
<td>1504705072</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Desktop Properties</td>
<td>Get the properties like desktop name and wallpaper path for the specified desktop</td>
</tr>
<tr>
<td>0x59E85D5D</td>
<td>1508400477</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>List of Available Desktops</td>
<td>List with all available desktops</td>
</tr>
<tr>
<td>0x5932B06F</td>
<td>1496494191</td>
<td>samples</td>
<td>UTF-16 string</td>
<td>Desktop - Modify Properties</td>
<td>Modify properties of a desktop. Format is desktop name;per-monitor flag;path to wallpaper image;wallpaper option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UTF-16</td>
<td>Desktop</td>
<td></td>
</tr>
<tr>
<td>Samples ID</td>
<td>Samples Size</td>
<td>Type</td>
<td>Function</td>
<td>Enable/Disable</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>------------</td>
<td>-----------------------------------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>0x5935FCD2</td>
<td>1496710354</td>
<td>String</td>
<td>Remove</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x59224B67</td>
<td>1495419751</td>
<td>32-bit unsigned integer</td>
<td>Enable Menu Collapse Desktop</td>
<td>Enable/Disable</td>
<td></td>
</tr>
<tr>
<td>0x591C2281</td>
<td>1495016065</td>
<td>32-bit unsigned integer</td>
<td>Enable Desktop Active Notification</td>
<td>Enable/Disable</td>
<td></td>
</tr>
<tr>
<td>0x599EE73D</td>
<td>1503586109</td>
<td>32-bit unsigned integer</td>
<td>Enable Desktop Allow Different Resolutions</td>
<td>Enable/Disable</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>32-bit unsigned integer</td>
<td>Enable/Disable</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-------------------------</td>
<td>----------------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x59FB742F</td>
<td>1509651503</td>
<td>Enable Desktop Name</td>
<td>Enable/Disable</td>
<td>Enable Desktop Name when switching name</td>
<td></td>
</tr>
<tr>
<td>0x59D055DC</td>
<td>1506825692</td>
<td>Enable Desktop Switching Speed</td>
<td>Enable/Disable</td>
<td>Enable Desktop Switching Speed</td>
<td></td>
</tr>
<tr>
<td>0x599A2B4E</td>
<td>1503275854</td>
<td>Enable Desktop Multiple Desksops</td>
<td>Enable/Disable</td>
<td>Enable Desktop Multiple Desksops</td>
<td></td>
</tr>
<tr>
<td>0x59F95C1D</td>
<td>1509514269</td>
<td>Enable Menu Individual settings</td>
<td>Enable/Disable</td>
<td>Enable Menu Individual settings</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Value</td>
<td>Samples</td>
<td>Integer Size</td>
<td>Enable Menu</td>
<td>Lock Title Bar</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>0x590CE624</td>
<td>1494017572</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Enable Menu</td>
<td>Lock Title Bar</td>
</tr>
<tr>
<td>0x59D49F0B</td>
<td>1507106571</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Enable Menu</td>
<td>Maximize Desktop</td>
</tr>
<tr>
<td>0x592A2071</td>
<td>1495933041</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Enable Menu</td>
<td>Send Application to Desktop</td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Function</td>
<td>Enable/Disable</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>------------------</td>
<td>-----------------------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>0x59873C39</td>
<td>1502035001</td>
<td>32-bit unsigned integer</td>
<td>Send Application Display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x59718CA3</td>
<td>1500613795</td>
<td>32-bit unsigned integer</td>
<td>Send Window Desktop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x59F48756</td>
<td>1509197654</td>
<td>32-bit unsigned integer</td>
<td>Send Window Display</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

512-bit unsigned integer
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Samples</th>
<th>Type</th>
<th>Description</th>
<th>Enable/Disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5999A457</td>
<td>1503241303</td>
<td>0x5999A457</td>
<td>unsigned integer</td>
<td>Enable Menu Always Top Option 'Always Top'</td>
<td>Enable/Disable</td>
</tr>
<tr>
<td>0x59CF13F4</td>
<td>1506743284</td>
<td>0x59CF13F4</td>
<td>32-bit unsigned integer</td>
<td>Enable nView Options System Menus</td>
<td>Enable/Disable</td>
</tr>
<tr>
<td>0x590B712C</td>
<td>1493922092</td>
<td>0x590B712C</td>
<td>32-bit unsigned integer</td>
<td>Enable Title Bar Collapse</td>
<td>Enable/Disable</td>
</tr>
<tr>
<td>0x599F9564</td>
<td>1503630692</td>
<td>0x599F9564</td>
<td>32-bit unsigned integer</td>
<td>Enable Title Bar Maximize</td>
<td>Enable/Disable</td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>32-bit unsigned integer</td>
<td>Enable Title Bar Next Display</td>
<td>Enable/Disable nView Options</td>
<td>Enable Menu Transparent</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-------------------------</td>
<td>--------------------------------</td>
<td>-----------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0x59F14C39</td>
<td>1508985913</td>
<td>samples</td>
<td>Enable Title Bar Next Display</td>
<td>Enable/Disable nView Options</td>
<td>Enable Menu Transparent</td>
</tr>
<tr>
<td>0x59BBCC16</td>
<td>1505479702</td>
<td>samples</td>
<td>Enable Title Bar Next Display</td>
<td>Enable/Disable nView Options</td>
<td>Enable Menu Transparent</td>
</tr>
<tr>
<td>0x59F9425B</td>
<td>1509507675</td>
<td>samples</td>
<td>Enable Title Bar Next Display</td>
<td>Enable/Disable nView Options</td>
<td>Enable Menu Transparent</td>
</tr>
<tr>
<td>Address</td>
<td>Sample Count</td>
<td>Type</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5951917B</td>
<td>1498517883</td>
<td>32-bit unsigned integer</td>
<td>Enable Menu Visible All Desktops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5971E144</td>
<td>1500635460</td>
<td>32-bit unsigned integer</td>
<td>Show Gridline Editor feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x59C8134B</td>
<td>1506284363</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Save Workspace State</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x597D8B59</td>
<td>1501399897</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Toggle Show All Desktops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x59414AE2</td>
<td>1497451234</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Display Grid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Samples</th>
<th>32-bit Unsigned Integer</th>
<th>Show Hotkey Action</th>
<th>Hide/Show Hotkey Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x59987658</td>
<td>1503163992</td>
<td>Show Hotkey</td>
<td>Activate Desktop</td>
<td>Show/Hide Hotkey will Activate</td>
</tr>
<tr>
<td>0x5915308A</td>
<td>1494560906</td>
<td>Show Hotkey</td>
<td>Adjust Display Gamma</td>
<td>Show/Hide adjust display</td>
</tr>
<tr>
<td>0x59CFA545</td>
<td>1506780485</td>
<td>Show Hotkey</td>
<td>Toggle Always Top</td>
<td>Show/Hide Toggle always top</td>
</tr>
<tr>
<td>0x5933274D</td>
<td>1496524621</td>
<td>Show Hotkey</td>
<td>Adjust Display Brightness</td>
<td>Show/Hide adjust display</td>
</tr>
<tr>
<td>0x590FC640</td>
<td>1494206016</td>
<td>Show Hotkey</td>
<td>Collapse All Windows</td>
<td>Show/Hide Collapse all</td>
</tr>
<tr>
<td>Address</td>
<td>Time</td>
<td>Samples</td>
<td>Type</td>
<td>Hotkey Action</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>0x59C3330F</td>
<td>1505964815</td>
<td></td>
<td>unsigned integer</td>
<td>Collapse To Desktop</td>
</tr>
<tr>
<td>0x591AE39E</td>
<td>1494934430</td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Adjust Display Contrast</td>
</tr>
<tr>
<td>0x592CEC41</td>
<td>1496116289</td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Desktop Lock</td>
</tr>
<tr>
<td>0x5910E51F</td>
<td>1494279455</td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Gamma Reset Brightness Contrast</td>
</tr>
<tr>
<td>0x59FEA9B8</td>
<td>1509861816</td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Display Gather All Monitor1</td>
</tr>
<tr>
<td>0x596F125A</td>
<td>1500451418</td>
<td></td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Load Profile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Count</td>
<td>Type</td>
<td>Function Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x592C5C9D</td>
<td>1496079517 samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Max Restore</td>
<td></td>
</tr>
<tr>
<td>0x59F10483</td>
<td>1508967555 samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Rotate Display</td>
<td></td>
</tr>
<tr>
<td>0x593A2A63</td>
<td>1496984163 samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Run Application</td>
<td></td>
</tr>
<tr>
<td>0x595EB7DA</td>
<td>1499379674 samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Task Toggle All Applications</td>
<td></td>
</tr>
<tr>
<td>0x596C172F</td>
<td>1500256047 samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Task Switcher Toggle Desktop Applications</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will rotate the desktop (or desktop) by specifying amount</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>that will rotate the display (or the desktop) by amount specify</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will run a user specified application</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will allow you to switch between applications on all desktops</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide allows to toggle desktop applications</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Description</td>
<td>Action</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>0x596BF876</td>
<td>1500248182</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Task Switcher Toggle Desktops</td>
<td>Show/Hide allows to to</td>
</tr>
<tr>
<td>0x59F512C8</td>
<td>1509233352</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey nViewToolbar Toggle</td>
<td>Show/Hide will show at toolbar</td>
</tr>
<tr>
<td>0x597DDC7F</td>
<td>1501420671</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Restore Workspace State</td>
<td>Show/Hide will restore state including desktop management and open applications</td>
</tr>
<tr>
<td>0x59EEC40F</td>
<td>1508819983</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Run Control Panel</td>
<td>Show/Hide will open the panel</td>
</tr>
<tr>
<td>0x59A0B829</td>
<td>1503705129</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Save Profile</td>
<td>Show/Hide will save profile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32-bit</td>
<td>Show Hotkey</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Action</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>------------</td>
<td>---------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>0x59979785</td>
<td>1503106949</td>
<td>samples</td>
<td>unsigned integer</td>
<td>Move Window Desktop</td>
</tr>
<tr>
<td>0x59AE7814</td>
<td>1504606228</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Move Window</td>
</tr>
<tr>
<td>0x5989E671</td>
<td>1502209649</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Move Window Next Display</td>
</tr>
<tr>
<td>0x591C9979</td>
<td>1495046521</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Desktop Name</td>
</tr>
<tr>
<td>0x5908B520</td>
<td>1493742880</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey nView Options</td>
</tr>
<tr>
<td>0x5944AE99</td>
<td>1497673369</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Toggle Clone</td>
</tr>
<tr>
<td>Address</td>
<td>Value</td>
<td>Description</td>
<td>Action Information</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------</td>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0x5917CCEE</td>
<td>1494732014</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Toggle LCD Scaling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will toggle LCD Scaling off</td>
<td></td>
</tr>
<tr>
<td>0x5900FA12</td>
<td>1493236242</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Toggle Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will Switch next device</td>
<td></td>
</tr>
<tr>
<td>0x59D5D836</td>
<td>1507186742</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Toggle Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will Toggle Transparency</td>
<td></td>
</tr>
<tr>
<td>0x5993EB80</td>
<td>1502866304</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Zoom Window</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide will show and hide the zoom window</td>
<td></td>
</tr>
<tr>
<td>0x59400FAE</td>
<td>1497370542</td>
<td>samples</td>
<td>32-bit unsigned integer</td>
<td>Show Hotkey Zoom Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Show/Hide toggles zoom window style between Cursor &amp; Magnifying Glass and Fixed Frame</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>--------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x59D169D7</td>
<td>1506896343</td>
<td>UTF-16 string</td>
<td>Profile Current</td>
<td>The name of the current profile</td>
</tr>
<tr>
<td>0x59ED4CAE</td>
<td>1508723886</td>
<td>UTF-16 string</td>
<td>Profile Delete</td>
<td>Delete the specified profile from nView Desktop Manager</td>
</tr>
<tr>
<td>0x59C6EED</td>
<td>1497132781</td>
<td>32-bit unsigned integer</td>
<td>Profile Load</td>
<td>Load the selected profile in nView Desktop Manager. All current nView Desktop Manager settings will be replaced with the profile's settings.</td>
</tr>
<tr>
<td>0x59EF38A2</td>
<td>1508849826</td>
<td>32-bit unsigned integer</td>
<td>Profile Lock</td>
<td>Lock the selected profile</td>
</tr>
<tr>
<td>0x5980CD81</td>
<td>1501613441</td>
<td>UTF-16 string</td>
<td>Profile Save</td>
<td>Save current nView Desktop Manager setting to selected profile</td>
</tr>
<tr>
<td>0x59A27815</td>
<td>1503819797</td>
<td>32-bit unsigned integer</td>
<td>Show Applications</td>
<td>Show/Hide Applications feature</td>
</tr>
<tr>
<td>0x598170AB</td>
<td>1501655211</td>
<td>32-bit unsigned integer</td>
<td>Show Virtual Desktop Editor</td>
<td>Show/Hide Virtual Desktop Editor feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32-bit</td>
<td>Show Hotkey</td>
<td>Show/Hide Hotkey Manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Samples</td>
<td>Type</td>
<td>Feature</td>
<td>Action</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
<td>------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>0x59014748</td>
<td>1493256008</td>
<td>32-bit integer</td>
<td>Manager feature</td>
<td>Show/Hide</td>
</tr>
<tr>
<td>0x59A7D007</td>
<td>1504169991</td>
<td>32-bit integer</td>
<td>Show Profile Manager feature</td>
<td>Show/Hide</td>
</tr>
<tr>
<td>0x5935F891</td>
<td>1496709265</td>
<td>32-bit integer</td>
<td>Show User Interface feature</td>
<td>Show/Hide</td>
</tr>
<tr>
<td>0x5972BABB</td>
<td>1500691131</td>
<td>32-bit integer</td>
<td>Show Window Manager feature</td>
<td>Show/Hide</td>
</tr>
<tr>
<td>0x59F49C46</td>
<td>1509203014</td>
<td>32-bit integer</td>
<td>Show Zoom feature</td>
<td>Show/Hide</td>
</tr>
</tbody>
</table>

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Third-Party Notice

Rapid XML is used under the following MIT license:

Copyright (c) 2006, 2007 Marcin Kalicinski

Permission is hereby granted, free of charge, to any person obtaining
of this software and associated documentation files (the "Software")
in the Software without restriction, including without limitation to
use, copy, modify, merge, publish, distribute, sublicense, and/or
of the Software, and to permit persons to whom the Software is furnished
subject to the following conditions:

The above copyright notice and this permission notice shall be included
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE SOFTWARE.
Application Class Reference

**Application**, associated with an application profile. [More...](#)

[List of all members.](#)

### Public Member Functions

```cpp
string info ()
```
Formats basic information about this **Application** into a human-readable string.

### Public Attributes

**Version verClass**

```cpp
uint32 id
```
Unique id of an application, used as a key.
string **name**
    User-friendly name of an application.

string **subPaths**
    Unique subpaths to identify an application, separated by ';'.

string **launcher**
    When non-empty, subpath to a launcher.

boolean **isPredefined**
    If true, application is predefined and cannot be removed.

boolean **isSupported**
    If true, profile is predefined and cannot be removed.

---

**Detailed Description**

**Application**, associated with an application profile.

---

**Member Function Documentation**

string info ( )

Formats basic information about this **Application** into a human-readable string.

---

**Member Data Documentation**

**uint32 id**

Unique id of an application, used as a key.

**boolean isPredefined**

If true, application is predefined and cannot be removed.

**boolean isSupported**
If true, profile is predefined and cannot be removed.

string **launcher**

When non-empty, subpath to a launcher.

string **name**

User-friendly name of an application.

string **subPaths**

Unique subpaths to identify an application, separated by ';'.

**Version verClass**

**Version of Application** class

The documentation for this class was generated from the following file:

- nvwmi.mof
ApplicationProfile Class Reference

**Application** profile.  More...

List of all members.

Public Member Functions

```cpp
boolean setValueById ([in]uint32 settingId,[in]uint32 value)
Set the 32-bit value by a setting ID.

boolean setBinaryValueById ([in]uint32 settingId,[in]uint8 value[])  
Set the binary value by a setting ID.

boolean setValueByld ([in]uint32 settingId,[in]uint32 value)
Set the string value by a setting ID.

boolean restoreSettings ([in]uint32 settingIds[]) 
Restore or delete settings, specified by IDs. Predefined
```
settings get restored and non-predefined settings get deleted.

string info ()
formats basic information about a profile into a human-readable string

boolean addApplications ([in]string appNames[])
Add applications to a Profile.

boolean removeApplications ([in]string appNames[])
Remove applications from a profile. Only non-predefined applications can be removed.

Public Attributes

Version verClass
uint32 id
  Unique id of a profile, used as a key.

string name
  Name of a profile.

boolean isPredefined
  If true, profile is predefined and cannot be removed.

boolean isSupported
  If true, profile is supported on this system.

Setting settings []
  Array of settings for a profile.

Application applications []
  uint32 type
  Application Profile type. Possible values are: 0 - 3D Application, 2 - nView Application.

DATETIME startTime
  Time when a profile was activated.

Detailed Description

Application profile.
Member Function Documentation

boolean addApplications ([in] string appNames[])

Add applications to a Profile.

Parameters:

appNames Array of Application names. In WMIC array of strings is: ("app1.exe", "app2.exe", "app3.exe")

string info()

formats basic information about a profile into a human-readable string

boolean removeApplications ([in] string appNames[])

Remove applications from a profile. Only non-predefined applications can be removed.

Parameters:

appNames Array of Application names. In WMIC array of strings is: ("app1.exe", "app2.exe", "app3.exe")

boolean restoreSettings ([in] uint32 settingIds[])

Restore or delete settings, specified by IDs. Predefined settings get restored and non-predefined settings get deleted.

Parameters:

settingIds Array of Setting IDs

boolean setBinaryValueById ([in] uint32 settingId, [in] uint8 value[])

Set the binary value by a setting ID.
Parameters:

settingId Setting ID
value Binary value as an array of bytes

boolean setValueById ([in] uint32 settingId,
                        [in] string value)

Set the string value by a setting ID.

Parameters:

settingId Setting ID
value String value

boolean setValueById ([in] uint32 settingId,
                        [in] uint32 value)

Set the 32-bit value by a setting ID.

Parameters:

settingId Setting ID
value 32-bit value

---------------------------------------

Member Data Documentation

Application applications[]

Array of applications associated to the Application Profile

uint32 id

Unique id of a profile, used as a key.

boolean isPredefined
If true, profile is predefined and cannot be removed.

boolean isSupported

If true, profile is supported on this system.

string name

Name of a profile.

Setting settings[]

Array of settings for a profile.

DATETIME startTime

Time when a profile was activated.

uint32 type

Application Profile type. Possible values are: 0 - 3D Application, 2 - nView Application.

Version verClass

Version of a profile class

The documentation for this class was generated from the following file:

- nvwmi.mof

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Board Class Reference

Represents a board with NVIDIA GPU(s). More...

List of all members.

Public Member Functions

```cpp
string info ()
    Formats basic information about Board instance into a human-readable string.
```

Public Attributes

```cpp
Version ver
    Unique id of named object, used as a key.
```
string name
  Name of an object.
string uname
  Unique name of an object.
sint32 ordinal
  Ordinal number of named object among objects with same names.
sint32 count
  Total number of named objects with same name.
string serialNumber
string nvapild
  Board ID from NVAPI.
string chipSKU
  SKU of the GPU chip.
string chipSKUMod
  SKU modifier of the GPU chip.
string project
  Project(Board) number.
string projectSKU
  Project(Board) SKU number.
Gpu ref gpus []
  Array of references to GPUs installed on this board.
ThermalProbe ref thermalProbes []
  Array of references to thermal probes of this board.
Cooler ref coolers []
  Array of references to coolers of this board.

---

**Detailed Description**

Represents a board with NVIDIA GPU(s).

---

**Member Function Documentation**

string info ()
Formats basic information about Board instance into a human-readable string.

---

**Member Data Documentation**

string *chipSKU*

SKU of the GPU chip.

string *chipSKUMod*

SKU modifier of the GPU chip.

*Cooler* ref *coolers[]*

Array of references to coolers of this board.

sint32 *count*

Total number of named objects with same name.

*Gpu* ref *gpus[]*

Array of references to GPUs installed on this board.

uint32 *id*

Unique id of named object, used as a key.

string *name*

Name of an object.

uint32 *nvapild*

*Board* ID from NVAPI.
sint32 ordinal

Ordinal number of named object among objects with same names.

string project

Project(Board) number.

string projectSKU

Project(Board) SKU number.

string serialNumber

Board Serial Number stored in the InfoROM

ThermalProbe ref thermalProbes[]

Array of references to thermal probes of this board.

string uname

Unique name of an object.

Version ver

Version of named object

The documentation for this class was generated from the following file:

- nvwmi.mof
Cooler Class Reference

Represents a cooler (fan, liquid system etc.). More...

List of all members.

Public Member Functions

```cpp
string info ()

Formats basic information about the Cooler object in a system into a human-readable string.
```

Public Attributes

```cpp
Version verClass
uint32 id
```

Unique ID.
uint32 handle
   Unique ID - deprecated, please use 'id'.

uint32 coolerType
   The type of cooler present on the card (fan, water or liquid).
   Possible values are: 0 - unknown, 1 - fan, 2 - water, 3 - liquid.

uint32 percentCoolerRate
   Current percentage speed of a cooler fan (or liquid flow rate %).

uint32 fanSpeed
   The tachometer reading for fan speed in rpm.

uint32 maxSpeed
   Maximal fan speed in rpm.

uint32 minSpeed
   Minimal fan speed in rpm.

uint32 coolerLevel
   Cooler fan speed event level. Possible values are: 0 - unknown,
   1 - normal, 2 - warning, 3 - critical.

---

**Detailed Description**

Represents a cooler (fan, liquid system etc.).

---

**Member Function Documentation**

string info()

Formats basic information about the Cooler object in a system into a human-readable string.

---

**Member Data Documentation**

uint32 coolerLevel

Cooler fan speed event level. Possible values are: 0 - unknown, 1 -
normal, 2 - warning, 3 - critical.

uint32 \texttt{coolerType}

The type of cooler present on the card (fan, water or liquid). Possible values are: 0 - unknown, 1 - fan, 2 - water, 3 - liquid.

uint32 \texttt{fanSpeed}

The tachometer reading for fan speed in rpm.

uint32 \texttt{handle}

Unique ID - deprecated, please use 'id'.

uint32 \texttt{id}

Unique ID.

uint32 \texttt{maxSpeed}

Maximal fan speed in rpm.

uint32 \texttt{minSpeed}

Minimal fan speed in rpm.

uint32 \texttt{percentCoolerRate}

Current percentage speed of a cooler fan (or liquid flow rate %).

\textbf{Version verClass}

\textbf{Cooler} class version

The documentation for this class was generated from the following file:

- \texttt{nvwm.mof}
CoolerEvent Class Reference

 Represents GPU cooler events. More...

 List of all members.

Public Attributes

uint32 handleGpu
uint32 coolerIndex

Cooler index in array of coolers on a given GPU.

uint32 coolerLevel

Cooler level to identify the cooler zone. Possible values are: 0 - unknown, 1 - normal, 2 - warning, 3 - critical.

Detailed Description
Represents GPU cooler events.

**Member Data Documentation**

uint32 `coolerIndex`

**Cooler** index in array of coolers on a given GPU.

uint32 `coolerLevel`

**Cooler** level to identify the cooler zone. Possible values are: 0 - unknown, 1 - normal, 2 - warning, 3 - critical.

uint32 `handleGpu`

Unique identification of the NVIDIA GPU

The documentation for this class was generated from the following file:

- `nvwmi.mof`
DesktopManager Class Reference

Management of nView desktops. More...

List of all members.

Public Member Functions

string **getAllDesktops** ()
Get a string with names of all Virtual Desktops. Desktop names are separated by semicolon. Returns empty string when Desktop Manager is disabled.

boolean **createDesktop** ([in]string name, [in]string backgrounds)
Create Virtual Desktop for nView Desktop Manager.

boolean **editDesktop** ([in]string name, [in]string backgrounds)
Edit properties of existing Virtual Desktop for nView Desktop Manager.

boolean **deleteDesktop** ([in]string name)
Delete Virtual Desktop from nView Desktop Manager.

Public Attributes

Version verClass

Detailed Description

Management of nView desktops.

Member Function Documentation

boolean createDesktop ( [in] string name,
                        [in] string backgrounds )

Create Virtual Desktop for nView Desktop Manager.

Parameters:

name nView desktop name

backgrounds string with per-display array of paths to the file with background image and style, separated by semicolon. Default style is "center" (0). for example:
"C:\ProgramData\Documents\My Pictures\Sample Pictures\Jellyfish.jpg#1;C:\ProgramData\Documents\My Pictures\Sample Pictures\Koala.jpg#0"

boolean deleteDesktop ( [in] string name )

Delete Virtual Desktop from nView Desktop Manager.

Parameters:

name nView desktop name
boolean editDesktop ([in] string name,
    [in] string backgrounds
    )

Edit properties of existing Virtual Desktop for nView Desktop Manager.

Parameters:
    name nView desktop name
    backgrounds string with per-display array of paths to the file with background image and style, separated by semicolon. Default style is "center" (0). for example: "C:\ProgramData\Documents\My Pictures\Sample Pictures\Jellyfish.jpg#1;C:\ProgramData\Documents\My Pictures\Sample Pictures\Koala.jpg#0"

string getAllDesktops ( )

Get a string with names of all Virtual Desktops. Desktop names are separated by semicolon. Returns empty string when Desktop Manager is disabled.

---

Member Data Documentation

Version verClass

Object version

---

The documentation for this class was generated from the following file:

- nvwmi.mof

---

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Display Class Reference

Represents physical display. More...

List of all members.

Public Member Functions

boolean restoreNativeDisplayMode ()
Restore the display mode to native.

boolean saveEDID ([in]string filePath)
save EDID into specified file (in text and binary format).

boolean setEDID ([in]string filePath)
Override EDID with specified data. File can be text or a

boolean setScaling ([in, ValueMap{"0","1","2","3","5","6","7"},
Values{"Default","Closest","Native","ScanoutToNative"},
uint32 scaling)
Change scaling mode of this display.
boolean setRotation ([in, ValueMap{"0","1","2","3"}, Values{"No rotation"}], [in] uint32 rotation)
    Rotate display.

string getCurrentTiming ()
    Get display's current timing into a human-readable string.

boolean saveGammaRamp ([in] string filePath)
    Save display Gamma Ramp into specified file (binary format).

boolean setGammaRamp ([in] string filePath)
    Override display Gamma Ramp data from specified file.

    Calculate and set Gamma Ramp from basic color controls.

boolean saveCSC ([in] string filePath)
    Save Color Space Conversion data into specified file (binary format).

boolean setCSC ([in] string filePath)
    Override Color Space Conversion data from specified file.

    Change display mode of a display grid, which contains this display.

boolean setDisplayModeById ([in] uint32 id)
    Change display mode of a display grid, which contains this display.

boolean setDisplayModeByRef ([in] DisplayMode ref mode)
    Change display mode of a display grid, which contains this display.

boolean setDither ([in, ValueMap{"0","1","2"}, Values{"Default","Enable","Disable"}], [in] uint32 state, [in, ValueMap{"6 bit","8 bit","10 bit"}], [in] uint32 bits, [in, ValueMap{"0","1","2","3","4"}, Values{"SpatialDynamic","SpatialStatic","SpatialDynamic2x2","SpatialStatic2x2","Temporal"}], [in] uint32 mode)
    Set dithering parameters.

string info ()
    Formats basic information about this display into a human-readable string.

Public Attributes

Version ver
    uint32 id
        Unique id of named object, used as a key.

string name
    Name of an object.
string `uname`
  Unique name of an object.

sint32 `ordinal`
  Ordinal number of named object among objects with same name.

sint32 `count`
  Total number of named objects with same name.

string `locus`

uint32 `nvapild`
  NVAPI `Display` ID.

**Version `verFirmware`**
  Firmware version of this display.

string `make`
  Manufacturer of this display.

string `model`
  Model of this display.

**DisplayMode `displayModeNative`**
  Native display mode.

uint32 `scaling`
  Scaling mode - Default, Closest, Native, ScanoutToNative, ScanoutToClosest. Possible values are: 0 - Default, 1 - Closest, 2 - Native, 3 - ScanoutToNative, 5 - ScanoutToClosest.

uint32 `rotation`
  Rotation angle. Possible values are: 0 - None, 1 - 90 degrees, 2 - 180 degrees, 3 - 270 degrees, 4 - Ignored.

uint8 `EDID` []
  Raw EDID of this display.

uint32 `EDIDSize`
  Size of the EDID in bytes.

sint32 `gpuConnectorType`
  Type of physical connector on a GPU side. Possible values:
  0x10 - TV Composite, 0x11 - TV S-Video, 0x13 - TV HDTV Component, 0x14 - TV SCAR 4120, 0x18 - PcPod HDTV-YPrPb, 0x19 - PcPod S-video Composite, 0x30 - DVI-i, 0x31 - DVI-d, 0x32 - ADC, 0x3 external, 0x47 - DP internal, 0x48 - DP mini, external, 0x49 - LFH DP 2.

sint32 `displayConnectorType`
  Type of physical connector on a display side. Possible values:
  0x10 - TV Composite, 0x11 - TV S-Video, 0x13 - TV HDTV Component, 0x14 - TV SCAR 4120, 0x18 - PcPod HDTV-YPrPb, 0x19 - PcPod S-video Composite, 0x30 - DVI-i, 0x31 - DVI-d, 0x32 - ADC, 0x3 external, 0x47 - DP internal, 0x48 - DP mini, external, 0x49 - LFH DP 2.
4120, 0x18 - PcPod HDTV-YPrPb, 0x19 - PcPod S-vide Composite, 0x30 - DVI-i, 0x31 - DVI-d, 0x32 - ADC, 0x3 external, 0x47 - DP internal, 0x48 - DP mini, external, 0
LFH DP 2.

boolean isActive
True if display is active.
sint32 ditherState
dithering state. Possible values are: -1 - N/A, 0 - Default
sint32 ditherBits
dithering to specified number of bits per channel. Possible values are:
-1 - N/A, 0 - 6 bit, 1 - 8 bit, 2 - 10 bit
sint32 ditherMode
dithering mode. Possible values are: -1 - N/A, 0 - Spatial - Temporal

DisplayMode ref displayModes []
Array of references to the Display modes.

---

**Detailed Description**

Represents physical display.

**Member Function Documentation**

`string getCurrentTiming()`
Get display's current timing into a human-readable string.

`string info()`
Formats basic information about this display into a human-readable string.

`boolean restoreNativeDisplayMode()`
Restore the display mode to native.
boolean saveCSC ( [in] string filePath )

Save Color Space Conversion data into specified file (binary format). Using file path with '.csc' extension is recommended.

Parameters:
   filePath Path to the file with CSC data. File will be in binary format, specifying '.csc' extension explicitly is recommended.

boolean saveEDID ( [in] string filePath )

Save EDID into specified file (in text and binary format). File path with extension '.bin' is recommended.

Parameters:
   filePath Path to the file with EDID data. File can be a binary or a text file, file path with extension '.bin' is recommended.

boolean saveGammaRamp ( [in] string filePath )

Save display Gamma Ramp into specified file (binary format). Using file path with '.gr' extension is recommended.

Parameters:
   filePath Path to the file with Gamma Ramp. File will be in binary format, specifying '.gr' extension explicitly is recommended.

boolean setCSC ( [in] string filePath )

Override Color Space Conversion data from specified file (binary format). Using file path with '.csc' extension is recommended.

Parameters:
   filePath Path to the file with CSC data. Using file path with '.csc' extension is recommended.

boolean setDisplayMode ( [in] uint32 width,
uint32 height,
real32 refreshRate,
uint32 depth)

Change display mode of a display grid, which contains this display, by specifying a display mode properties.

**Parameters:**
- *width* Width of the display grid (in pixels)
- *height* Height of the display grid (in pixels)
- *refreshRate* Refresh rate of the display grid (in Hz)
- *depth* Color depth of the display grid (in bits per pixel)

boolean setDisplayModeById([in] uint32 id)

Change display mode of a display grid, which contains this display, by specifying a display mode ID.

**Parameters:**
- *id* DisplayMode ID

boolean setDisplayModeByRef([in] DisplayMode ref mode)

Change display mode of a display grid, which contains this display, by specifying a reference to a display mode.

**Parameters:**
- *mode* A reference to the display mode to be applied

boolean setDither([in, ValueMap{"0","1","2"}, Values{"Default","Enable","Disable"}],
[in, ValueMap{"0","1","2"}, Values{"6 bit","8 bit","10 bit"}])
uint32
[in, ValueMap{"0","1","2","3","4"},
Values{"SpatialDynamic","SpatialStatic","SpatialDynamic2x2","Temporal"}],
uint32
Set dithering parameters.

**Parameters:**

- `state` New dithering state. Required
- `bits` Dither to specified number of bits per channel. Optional
- `mode` Dithering mode. Optional

```cpp
boolean setEDID ([in] string filePath )
```

Override EDID with specified data. File can be text or a binary file. File path with extension '.bin' is recommended.

**Parameters:**

- `filePath` Path to the file with EDID data. This parameter must be empty string "" for removing the forced EDID.

```cpp
boolean setGammaRamp ([in] string filePath )
```

Override display Gamma Ramp data from specified file (binary format). Using file path with '.gr' extension is recommended.

**Parameters:**

- `filePath` Path to the file with Gamma Ramp. Using file path with '.gr' extension is recommended.

```cpp
boolean setGammaRampBasic ( [in] real32 brightness,
                          [in] real32 contrast,
                          [in] real32 gamma )
```

Calculate and set Gamma Ramp from basic color controls.

**Parameters:**

- `brightness` brightness: -0.2 .. +0.2 default: 0.0
- `contrast` contrast: -0.2 .. +0.2 default: 0.0
**gamma**

gamma: 0.3 .. 2.8 default: 1.0

**boolean setRotation**

[in, ValueMap{"0","1","2","3"}, Values{"No rotation","rotate 90 degrees","rotate 180 degrees","rotate 270 degrees"}] uint32 rotation

Rotate display.

**Parameters:**

*rotation* rotation to be set

**boolean setScaling**

[in, ValueMap{"0","1","2","3","5","6","7"}, Values{"Default","Closest","Native","ScanoutToNative","AspectScanoutToNative","AspectScanoutToClosest","ScanoutToClosest"}] uint32 scaling

Change scaling mode of this display.

**Parameters:**

*scaling* scaling mode to be set

---

**Member Data Documentation**

**sint32 count**

Total number of named objects with same name.

**sint32 displayConnectorType**

Type of physical connector on a display side. Possible values are: -1 - Unknown, 0 - VGA 15-pin, 0x10 - TV Composite, 0x11 - TV S-Video, 0x13 - TV HDTV Component, 0x14 - TV SCART, 0x16 - TV Composite SCART on EIAJ-4120, 0x17 - TV HDTV EIAJ-4120, 0x18 - PcPod HDTV-YPrPb, 0x19 - PcPod S-video, 0x1A - PcPod Composite, 0x20 - DVI-i TV S-Video, 0x21 - DVI-i TV Composite, 0x30 - DVI-i, 0x31 - DVI-d, 0x32 - ADC, 0x38 - Lfh DVI-i 1, 0x39 - Lfh DVI-i 2, 0x40 - SPWG, 0x41 - OEM, 0x46 - DP external, 0x47 - DP internal, 0x48 - DP mini, external, 0x61 -
HDMI type A, 0x63 - HDMI type C mini, 0x64 - LFH DP 1, 0x65 - LFH DP 2.

**DisplayMode displayModeNative**

Native display mode.

**DisplayMode ref displayModes[]**

Array of references to the Display modes.

sint32 **ditherBits**

dithering to specified number of bits per channel. Possible values are: -1 - N/A, 0 - 6 bit, 1 - 8 bit, 2 - 10 bit

sint32 **ditherMode**

dithering mode. Possible values are: -1 - N/A, 0 - SpatialDynamic, 1 - SpatialStatic, 2 - SpatialDynamic2x2, 3 - SpatialStatic2x2, 4 - Temporal

sint32 **ditherState**

dithering state. Possible values are: -1 - N/A, 0 - Default, 1 - Enable, 2 - Disable

uint8 **EDID[]**

Raw EDID of this display.

uint32 **EDIDSize**

Size of the EDID in bytes.

sint32 **gpuConnectorType**

Type of physical connector on a GPU side. Possible values are: -1 - Unknown, 0 - VGA 15-pin, 0x10 - TV Composite, 0x11 - TV S-Video,
0x13 - TV HDTV Component, 0x14 - TV SCART, 0x16 - TV Composite SCART on EIAJ-4120, 0x17 - TV HDTV EIAJ-4120, 0x18 - PcPod HDTV-YPbPr, 0x19 - PcPod S-video, 0x1A - PcPod Composite, 0x20 - DVI-i TV S-Video, 0x21 - DVI-i TV Composite, 0x30 - DVI-i, 0x31 - DVI-d, 0x32 - ADC, 0x38 - Lfh DVI-i 1, 0x39 - Lfh DVI-i 2, 0x40 - SPWG, 0x41 - OEM, 0x46 - DP external, 0x47 - DP internal, 0x48 - DP mini, external, 0x61 - HDMI type A, 0x63 - HDMI type C mini, 0x64 - LFH DP 1, 0x65 - LFH DP 2.

```
uint32 id
```

Unique id of named object, used as a key.

```
boolean isActive
```

True if display is active.

```
string locus
```

locus of the display - <gpu#>.<output#>

```
string make
```

Manufacturer of this display.

```
string model
```

Model of this display.

```
string name
```

Name of an object.

```
uint32 nvapild
```

NVAPI Display ID.

```
sint32 ordinal
```

Ordinal number of named object among objects with same names.

**uint32 rotation**

rotation angle. Possible values are: 0 - None, 1 - 90 degree, 2 - 180 degree, 3 - 270 degree, 4 - Ignored

**uint32 scaling**

**Display** scaling mode - Default, Closest, Native, ScanoutToNative, AspectScanoutToNative, AspectScanoutToClosest, ScanoutToClosest. Possible values are: 0 - Default, 1 - Closest, 2 - Native, 3 - ScanoutToNative, 5 - AspectScanoutToNative, 6 - AspectScanoutToClosest, 7 - ScanoutToClosest.

**string uname**

Unique name of an object.

**Version ver**

**Version** of named object

**Version verFirmware**

Firmware version of this display.

---

The documentation for this class was generated from the following file:

- nvwmimof

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
DisplayGrid Class Reference

Represents physical displays, organized into a regular grid (rows by columns). More...

List of all members.

Public Member Functions

boolean setOverlapRow ([in]sint32 index,[in]sint32 overlap)
  Change per-row overlap of the display grid item with given index.

boolean setOverlapCol ([in]sint32 index,[in]sint32 overlap)
  Change per-col overlap of the display grid item with given index.

boolean setOverlaps ([in]sint32 index,[in]sint32 overlapRow,
   [in]sint32 overlapCol)
  Change overlap of the display grid item with given
index.

Change display mode of this display grid by specifying a display mode properties.

boolean setDisplayModeById ([in]uint32 id)  
Change mode of this display grid by specifying a display mode ID.

boolean setDisplayModeByRef ([in]DisplayMode ref gridMode)  
Change mode of this display grid by specifying a reference to a display mode.

boolean setRotation ([in]uint32 rotation[])  
Change rotation of the displays in this grid by specifying the per-display rotation mode. Currently only common mode for all displays in a grid is supported.

boolean saveGammaRamp ([in]string filePath)  
Save Gamma Ramp data from all displays into specified file (binary format). Using file path with '.grg' extension is recommended.

boolean setGammaRamp ([in]string filePath)  
Override Gamma Ramp data for all displays from specified file (binary format). Using file path with '.grg' extension is recommended.

boolean setGammaRampBasic ([in]real32 brightness,[in]real32 contrast,[in]real32 gamma)  
Calculate and set Gamma Ramp from basic color controls.

boolean saveCSC ([in]string filePath)  
Save Color Space Conversion data into specified file (binary format). Using file path with '.cscg' extension is recommended.

boolean setCSC ([in]string filePath)  
Override Color Space Conversion data from specified file (binary format). Using file path with '.cscg' extension is recommended.
info ()
Formats basic information about this Display grid into a human-readable string.

Public Attributes

Version ver
  uint32 id
    Unique id of named object, used as a key.
  string name
    Name of an object.
  string uname
    Unique name of an object.
  sint32 ordinal
    Ordinal number of named object among objects with same names.
  sint32 count
    Total number of named objects with same name.
  uint32 rows
  uint32 cols
    Number of columns in the grid.
  sint32 overlapRows []
    Distance between displays per row. Positive number indicates overlap, negative - gap (in pixels).
  sint32 overlapCols []
    Distance between displays per column. Positive number indicates overlap, negative - gap (in pixels).

OverlapLimits overlapLimits
  Overlap limits (in pixels) for this display grid.

DisplayMode displayModeVirtual
  Virtual mode of this display grid (including overlaps).

DisplayMode displayModePhysical
  Current mode of this display grid (per-display).
  sint32 positionCol
    Column-wise (horizontal) position of the display grid (in
sint32 positionRow  
Row-wise (vertical) position of the display grid (in pixels).

uint32 rotation []  
per-display rotation modes in a grid. Possible values are: 0 - No rotation, 1 - rotate 90 degrees, 2 - rotate 180 degrees, 3 - rotate 270 degrees

DisplayMode ref displayModes []  
Array of references to modes available on this display grid.

Display ref displays []  
Array of references to displays bound into this display grid.

Gpu ref gpus []  
Array of references to GPUs, driving displays on this display grid.

---

**Detailed Description**

Represents physical displays, organized into a regular grid (rows by columns).

---

**Member Function Documentation**

string info ( )

Formats basic information about this Display grid into a human-readable string.

boolean saveCSC ([in] string filePath )

Save Color Space Conversion data into specified file (binary format). Using file path with '.cscg' extension is recommended.
Parameters:

\textit{filePath} Path to the file with CSC data. File will be in binary format, specifying '.cscg' extension explicitly is recommended.

boolean saveGammaRamp ([in] string \textit{filePath} )

Save Gamma Ramp data from all displays into specified file (binary format). Using file path with '.grg' extension is recommended.

Parameters:

\textit{filePath} Path to the file with Gamma Ramp. File will be in binary format, specifying '.grg' extension explicitly is recommended.

boolean setCSC ([in] string \textit{filePath} )

Override Color Space Conversion data from specified file (binary format). Using file path with '.cscg' extension is recommended.

Parameters:

\textit{filePath} Path to the file with CSC data. Using file path with '.cscg' extension is recommended.

boolean setDisplayMode ([in] uint32 \textit{width},
[in] uint32 \textit{height},
[in] real32 \textit{refreshRate},
[in] uint32 \textit{depth}
)

Change display mode of this display grid by specifying a display mode properties.

Parameters:

\textit{width} Width of the display grid (in pixels)
\textit{height} Height of the display grid (in pixels)
\textit{refreshRate} Refresh rate of the display grid (in Hz)
**depth**  Color depth of the display grid (in bits per pixel)

boolean setDisplayModeById ([in] uint32  *id*)

Change mode of this display grid by specifying a display mode ID.

**Parameters:**

* id  **DisplayMode** ID

boolean setDisplayModeByRef ([in] **DisplayMode** ref  *gridMode*)

Change mode of this display grid by specifying a reference to a display mode.

**Parameters:**

* gridMode  A reference to the display mode to be applied

boolean setGammaRamp ([in] string  *filePath*)

Override Gamma Ramp data for all displays from specified file (binary format). Using file path with '.grg' extension is recommended.

**Parameters:**

* filePath  Path to the file with Gamma Ramp. Using file path with '.grg' extension is recommended.

boolean setGammaRampBasic ([in] real32  *brightness*,
                           [in] real32  *contrast*,
                           [in] real32  *gamma*)

Calculate and set Gamma Ramp from basic color controls.

**Parameters:**

* brightness  brightness: -0.2 .. +0.2 default: 0.0
* contrast    contrast: -0.2 .. +0.2 default: 0.0
* gamma       gamma: 0.3 .. 2.8 default: 1.0
boolean setOverlapCol ([in] sint32 index,
                  [in] sint32 overlap
                    )

Change per-col overlap of the display grid item with given index.

Parameters:
  index     Index of the column. Value -1 could be used to change
             overlap for all columns at once
  overlap   Overlap. Positive number indicates overlap, negative - gap
             (in pixels)

boolean setOverlapRow ([in] sint32 index,
                     [in] sint32 overlap
                      )

Change per-row overlap of the display grid item with given index.

Parameters:
  index     Index of the column. Value -1 could be used to change
             overlap for all rows at once
  overlap   Overlap. Positive number indicates overlap, negative - gap
             (in pixels)

boolean setOverlaps ([in] sint32 index,
                  [in] sint32 overlapRow,
                  [in] sint32 overlapCol
                    )

Change overlap of the display grid item with given index.

Parameters:
  index     Index of the item. Value -1 could be used to change
             overlap for all items at once
  overlapRow Overlap. Positive number indicates overlap, negative - gap (in pixels)
**overlapCol**  
Overlap. Positive number indicates overlap, negative - gap (in pixels)

boolean setRotation ([in] uint32 rotation[])

Change rotation of the displays in this grid by specifying the per-display rotation mode. Currently only common mode for all displays in a grid is supported.

**Parameters:**

- **rotation** per-display rotation modes to be set in the grid. Currently only identical rotation angle for all displays in a grid is supported.
  
  Example 1: for 1x4 grid, "setRotation 1" will apply "rotate 90 degrees" to all displays in the grid.
  
  Example 2: specifying same angle for all displays is supported: "setRotation rotation=(1,1,1,1)"
  
  Example 3: attempt to rotate to non-identical angles "setRotation rotation=(1,2,1,1)" will fail

---

**Member Data Documentation**

uint32 **cols**

Number of columns in the grid.

sint32 **count**

Total number of named objects with same name.

**DisplayMode** **displayModePhysical**

Current mode of this display grid (per-display).

**DisplayMode** **ref** **displayModes[]**
Array of references to modes available on this display grid.

**DisplayMode displayModeVirtual**

Virtual mode of this display grid (including overlaps).

**Display** ref **displays**

Array of references to displays bound into this display grid.

**Gpu** ref **gpus**

Array of references to GPUs, driving displays on this display grid.

**uint32 id**

Unique id of named object, used as a key.

**string name**

Name of an object.

**sint32 ordinal**

Ordinal number of named object among objects with same names.

**sint32 overlapCols**

Distance between displays per column. Positive number indicates overlap, negative - gap (in pixels).

**OverlapLimits overlapLimits**

Overlap limits (in pixels) for this display grid.

**sint32 overlapRows**

Distance between displays per row. Positive number indicates overlap,
negative - gap (in pixels).

\texttt{sint32 positionCol}

Column-wise (horizontal) position of the display grid (in pixels).

\texttt{sint32 positionRow}

Row-wise (vertical) position of the display grid (in pixels).

\texttt{uint32 rotation[]}

Per-display rotation modes in a grid. Possible values are: 0 - No rotation, 1 - rotate 90 degrees, 2 - rotate 180 degrees, 3 - rotate 270 degrees

\texttt{uint32 rows}

Number of rows in the grid

\texttt{string uname}

Unique name of an object.

\textbf{Version ver}

\textbf{Version} of named object

The documentation for this class was generated from the following file:

- \texttt{nvwmi.mof}

\textbf{NVIDIA}

\textit{Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.}
DisplayGridInfo Class Reference

Information about a display grid. More...

List of all members.

Public Member Functions

string info ()
  Formats basic information about the display grid information into a human-readable string.

Public Attributes

Version verClass
  uint32 id
  Unique id of a display grid information, used as a key.
uint32 rows
   Number of rows in the display grid.
uint32 cols
   Number of columns in the display grid.
uint32 primaryId
   ID of the primary display.
uint32 displayIds []
   Array of display IDs (Display.id).
string unames
   string of unique display names (Display.uname), separated by semicolon. Order corresponds to displayIds property
string cscFilePath
   string, containing a file path to CSC data
string gammaRampFilePath
   path to the file which contains desktop gamma ramps for every display in the grid

---

**Detailed Description**

Information about a display grid.

---

**Member Function Documentation**

string info ()

Formats basic information about the display grid information into a human-readable string.

---

**Member Data Documentation**

uint32 cols

Number of columns in the display grid.
string **cscFilePath**

string, containing a file path to CSC data

uint32 **displayIds[]**

Array of display IDs (**Display.id**).

string **gammaRampFilePath**

path to the file which contains desktop gamma ramps for every display in the grid

uint32 **id**

Unique id of a display grid information, used as a key.

uint32 **primaryId**

ID of the primary display.

uint32 **rows**

Number of rows in the display grid.

string **unames**

string of unique display names (**Display.uname**), separated by semicolon. Order corresponds to **displayIds** property

**Version verClass**

**Version** of a display grid information

The documentation for this class was generated from the following file:

- nvwmi.mof
DisplayManager Class Reference

Management of display-related task. More...

List of all members.

Public Member Functions

boolean **setScalingAll** ([in, ValueMap{"0","1","2","3","5","6","7"}], Values{"Default","Closest","Native","ScanoutToNative","AspectScanoutToNative","AspectScanoutToClosest","ScanoutToClosest"}, uint32 scaling)
Change scaling mode of all display targets in a system.

boolean **validateDisplayGridById** ([in]uint32 cols,[in]uint32 rows,[in]uint32 displayIds[])
Validate the display grid by specifying display IDs, before creating validateDisplayGrids method.

boolean **createDisplayGridById** ([in]uint32 cols,[in]uint32 rows,[in]uint32 displayIds[])
Create a display grid by specifying display IDs. Creates only a sir Will disable all previous grids.
boolean createDisplayGridByName ([in]uint32 cols, [in]uint32 rows, [in]string displayNames)
Create a display grid by specifying unique display names. Create method. Will disable all previous grids.

Create a display grid by specifying references to the Display class. Will disable all previous grids.

boolean validateDisplayGrids ([in]string grids[])
Validate multiple display grid topologies.

boolean createDisplayGrids ([in]string grids[])
Create multiple display grid topologies.

boolean setGridPositions ([in]uint32 positionCol[], [in]uint32 positionRow[])
Set desktop positions of all active display grids. All active display taskbar must be positioned at (0,0).

boolean fakeEDIDAll ([in]string filePath, [in, ValueMap{"-1","0","1","2","3","unknown","uninitialized","VGA","Component","S-Video"}] uint32 output)
Fake given EDID across all GPUs, on all or only specified display.

boolean tryCustomTiming ([in]string timing)
Try the new custom timing without applying it.

boolean createCustomTiming ([in]string timing)
Create the new Custom timing. This will only save the new timing.

boolean editCustomTiming ([in]uint32 modeId, [in]string newTiming)
Modify the existing Custom timing. This will only save the new timing.

boolean deleteCustomTiming ([in]string timing)
Delete the existing Custom timing. This will permanently delete the.

string enumCustomTimings ([in]uint32 displays[])
Enumerates all existing Custom timing for the given displays.

boolean saveCustomTimings ([in]uint32 displays[], [in]string filePath)
Saves all existing Custom Timings for the given displays to given

boolean loadCustomTimings ([in]uint32 displays[], [in]string filePath)
Loads all existing Custom Timings into the given displays from gi

boolean createClone ([in, ValueMap{"0","1"}, Values{"basic","smart"}] uint32 type, [in]uint32 source, [in]uint32 targets[])
Clone displays. Note that all displays must be connected to the same GPU.

boolean createModeFilter ([in]uint32 displays[], [in]string filter)
Create display mode filter for given displays.

boolean deleteModeFilter ([in]uint32 displays[])
Create display mode filter for given displays.
Enumerate display mode filters.

```cpp
string info ()
```

Formats basic information about `DisplayManager` objects in a system.

---

**Public Attributes**

**Version verClass**

---

**Detailed Description**

Management of display-related task.

---

**Member Function Documentation**

```cpp
boolean createClone ([in, ValueMap{"0","1"}, Values{"basic","smart"}] uint32 type,
                   [in] uint32 source,
                   [in] uint32 targets[])
```

Clone displays. Note that all displays must be connected to the same GPU.

**Parameters:**

- `type` type of clone - basic (same display resolution) or smart (pan and scan target to a source display)
- `source` display ID of clone source (`Display.id`)
- `targets` array of clone target display IDs (`Display.id`)

```cpp
boolean createCustomTiming ([in] string timing )
```

Create the new Custom timing. This will only save the new timing and not
apply. To apply, call setDisplayMode.

Parameters:

```
 timing String of custom timing parameters "displayLocus=1001
  1002 2001 2002...; mode=1720 1000 32 60.45; override=6
 possible override values: CURRENT=0, AUTO=1, EDID=2, DMT=3, DMT_RB=4, CVT=5, CVT_RB=6, GFT=7, EIA861=8, ANALOG_TV=9, CUST=10, NV_PSF=11, NV_ASPR=12, SDI=13. Note - not all override values may be supported. Default is CVT_RB Other parameters are optional and will be deduced from default
```

boolean createDisplayGridById ([in] uint32 cols,
                          [in] uint32 rows,
                          [in] uint32 displayIds[])

Create a display grid by specifying display IDs. Creates only a single grid, for multigrid, please use createDisplayGrids method. Will disable all previous grids.

Parameters:

cols number of columns in a grid (must be non-zero)
rows number of rows in a grid (must be non-zero)
displayIds array of display IDs (Display.id). Order is important. Convention for array - grid element at column=i and row=j must be at index=j*cols+i

boolean createDisplayGridByName ([in] uint32 cols,
                          [in] uint32 rows,
                          [in] string displayNames)

Create a display grid by specifying unique display names. Creates only a single grid, for multigrid, please use createDisplayGrids method. Will disable all previous grids.
**Parameters:**

- **cols** number of columns in a grid (must be non-zero)
- **rows** number of rows in a grid (must be non-zero)
- **displayNames** string of unique display names (Display.uname), separated by semicolon. Order is important. C convention for array - grid element at column=i and row=j must be at index=j*cols+i

```cpp
boolean createDisplayGridByRef ( [in] uint32 cols,
                          [in] uint32 rows,
                          [in] string displayRefs
)
```

Create a display grid by specifying references to the `Display` class instances. Creates only a single grid, for multigrid, please use createDisplayGrids method. Will disable all previous grids.

**Parameters:**

- **cols** number of columns in a grid (must be non-zero)
- **rows** number of rows in a grid (must be non-zero)
- **displayRefs** string of references to `Display` instances, separated by semicolon. Order is important. C convention for array - grid element at column=i and row=j must be at index=j*cols+i

```cpp
boolean createDisplayGrids ( [in] string grids[] )
```

Create multiple display grid topologies.

**Parameters:**

- **grids** array of grid topologies to be created, each grid specified by a string with key=value pairs
  
  "rows=2;cols=2;stereo=0;layout=1.1 1.2 1.3 1.4;mode=1920 1200 32 60;rotation=0 0 0 0",
  
  "rows=1;cols=4;stereo=0;layout=2.1 2.2 2.3 2.4;mode=1920 1200 32 60;rotation=1 1 1 1", ...

Only "rows" and "cols" are the
mandatory keys in a string with grid specification Other parameters are optional and will be deduced from a default topology

boolean createModeFilter ( [in] uint32  displays[],
                          [in] string  filter
                      )

Create display mode filter for given displays.

Parameters:
  displays  Array of display IDs (Display.id)
  filter    filter definition in format "<type>;<limits>;". Type could be <, > or =. Limits are given as <width>x<height>x<color depth><frequency>. Zero value means no filtering will occur for given limit.
Example 1: ">;999x888x31@0" - all display modes with resolution more than 999x998 and color depth 32 or more, at any refresh rate will pass.
Example 2: "=;1024x768x0@59.95" - only display mode 1024x768 with any color depth at refresh rate 59.95 Hz will pass.

boolean deleteCustomTiming ( [in] string  timing )

Delete the existing Custom timing. This will permanently delete the timing of the display.

Parameters:
  timing  String of custom timing parameters "displayLocus=1001 1002 2001 2002...; mode=1720 1000 32 60.45;
override=CVT_RB possible override values: CURRENT=0,
AUTO=1, EDID=2, DMT=3, DMT_RB=4, CVT=5,
CVT_RB=6, GFT=7, EIA861=8, ANALOG_TV=9, CUST=10,
NV_PSF=11, NV_ASPR=12, SDI=13. Note - not all override values may be supported. Default is CVT_RB Other parameters are optional and will be deduced from default
boolean deleteModeFilter ( [in] uint32 displays[] )

Create display mode filter for given displays.

**Parameters:**

displays Array of display IDs (Display.id)

boolean editCustomTiming ( [in] uint32 modeId,
                        [in] string newTiming )

Modify the existing Custom timing. This will only save the new timing and not apply. To apply, call setDisplayMode.

**Parameters:**

modeId Existing custom timing source mode Id
newTiming String of new custom timing parameters
"displayLocus=1001 1002 2001 2002...; mode=1720 1000 32 60.45; override=6 possible override values: CURRENT=0, AUTO=1, EDID=2, DMT=3, DMT_RB=4, CVT=5, CVT_RB=6, GFT=7, EIA861=8, ANALOG_TV=9, CUST=10, NV_PSF=11, NV_ASPR=12, SDI=13. Note - not all override values may be supported. Default is CVT_RB Other parameters are optional and will be deduced from default

string enumCustomTimings ( [in] uint32 displays[] )

Enumerates all existing Custom timing for the given displays.

**Parameters:**

displays Array of display IDs (Display.id)

string enumModeFilters ( [in] uint32 displays[] )

Enumerate display mode filters.
Parameters:

*displays* Array of display IDs (**Display.id**)

```c
boolean fakeEDIDAll ( [in] string filePath, [in, ValueMap{"-1","0","1","2","3","4","5","6","7","8","9"}, Values{"unknown","uninitialized","VGA","Component","S-Video","HDMI","DVI","LVDS","DP","Composite","All"}] uint32 output )
```

Fake given EDID across all GPUs, on all or only specified display outputs.

Parameters:

*filePath* Full path to the file with the custom EDID. In order to remove the forced EDID, specify an empty string

*output* **Display** output type for faking EDID. In order to force/remove the EDID on all display outputs, specify the output value "All"

```c
string info ( )
```

Formats basic information about **DisplayManager** objects in a system into a human-readable string.

```c
boolean loadCustomTimings ( [in] uint32 displays[],
                          [in] string filePath )
```

Loads all existing Custom Timings into the given displays from given XML file. File format is compatible with Control Panel.

Parameters:

*displays* Array of display IDs (**Display.id**)

*filePath* Path to the file with Custom Timings
boolean saveCustomTimings ( [in] uint32 displays[],
                      [in] string filePath
                     )

Saves all existing Custom Timings for the given displays to given XML file. File format is compatible with Control Panel.

Parameters:

   displays  Array of display IDs (Display.id)
   filePath  Path to the file with Custom Timings

boolean setGridPositions ( [in] uint32 positionCol[],
                           [in] uint32 positionRow[]
                          )

Set desktop positions of all active display grids. All active display grids have to be re-positioned together. Primary grid (with taskbar) must be positioned at (0,0).

Parameters:

   positionCol  array of valid grid column positions in pixels. Order is important. Also, DisplayGrid::colOverlap (if any) should be considered.

   positionRow  array of valid grid row positions in pixels. Order is important. Also, DisplayGrid::rowOverlap (if any) should be considered.

boolean setScalingAll ( [in, ValueMap{"0","1","2","3","5","6","7"},
                         Values{"Default","Closest","Native","ScanoutToNative","AspectScanoutToNative","AspectScanoutToClosest","ScanoutToClosest"}],
                       uint32
                      )

Change scaling mode of all display targets in a system.

Parameters:

   scaling  Scaling mode to be set
boolean tryCustomTiming ( [in] string timing )

Try the new custom timing without applying it.

Parameters:

  timing String of custom timing parameters "displayLocus=1001
  1002 2001 2002...; mode=1720 1000 32 60.45; override=6
  possible override values: CURRENT=0, AUTO=1, EDID=2,
  DMT=3, DMT_RB=4, CVT=5, CVT_RB=6, GFT=7,
  EIA861=8, ANALOG_TV=9, CUST=10, NV_PSF=11,
  NV_ASPR=12, SDI=13. Note - not all override values may be
  supported. Default is CVT_RB Other parameters are optional
  and will be deduced from default

boolean validateDisplayGridByIds ( [in] uint32 cols,
  [in] uint32 rows,
  [in] uint32 displayIds[] )

Validate the display grid by specifying display IDs, before creating the
grid. Validates only a single grid, for multigrid, please use
validateDisplayGrids method.

Parameters:

  cols number of columns in a grid (must be non-zero)
  rows number of rows in a grid (must be non-zero)
  displayIds array of display IDs (Display.id). Order is important. C
  convention for array - grid element at column=i and row=j
  must be at index=j*cols+i

boolean validateDisplayGrids ( [in] string grids[] )

Validate multiple display grid topologies.

Parameters:

  grids array of grid topologies to be validated, each grid specified by
  a string with key=value pairs
"rows=2;cols=2;stereo=0;layout=1.1 1.2 1.3 1.4;mode=1920 1200 32 60;rotation=0 0 0 0",
"rows=1;cols=4;stereo=0;layout=2.1 2.2 2.3 2.4;mode=1920 1200 32 60;rotation=1 1 1 1", ... Only "rows" and "cols" are the mandatory keys in the string with grid specification Other parameters are optional and will be deduced from a default topology

---

**Member Data Documentation**

**Version verClass**

Object version

The documentation for this class was generated from the following file:

- nvwmi.mof

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
DisplayMode Class Reference

Represents display mode. More...

List of all members.

Public Member Functions

string info ()
    Formats basic information about this display mode into a human-readable string.

Public Attributes

Version verClass
    Unique identification of the display mode.
sint32 width
Display mode width.
sint32 height
Display mode height.
real32 refreshRate
Display mode refresh rate.
sint32 colorDepth
Display mode color depth in bits per pixel (bpp).

---

**Detailed Description**

Represents display mode.

---

**Member Function Documentation**

string info()

Formats basic information about this display mode into a human-readable string.

---

**Member Data Documentation**

sint32 colorDepth

*Display* mode color depth in bits per pixel (bpp).

sint32 height

*Display* mode height.

uint32 id

Unique identification of the display mode.
real32 refreshRate

Display mode refresh rate.

Version verClass

Object version

sint32 width

Display mode width.

The documentation for this class was generated from the following file:

- nvwmi.mof

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
**DisplayProfile Class Reference**

**Display** profile. More...

[List of all members.](#)

### Public Member Functions

- `boolean setValueById ([in]uint32 settingId,[in]uint32 value)`
  Set the 32-bit value by a setting ID.

- `boolean setBinaryValueById ([in]uint32 settingId,[in]uint8 value[])`
  Set the binary value by a setting ID.

- `boolean setStringValueById ([in]uint32 settingId,[in]string value)`
  Set the string value by a setting ID.

- `boolean restoreSettings ([in]uint32 settingIds[])`
  Restore or delete settings, specified by IDs. Predefined
settings get restored and non-predefined settings get deleted.

string info ()
formats basic information about a profile into a human-readable string

Public Attributes

**Version verClass**

*uint32 id*
Unique id of a profile, used as a key.

*string name*
Name of a profile.

*boolean isPredefined*
If true, profile is predefined and cannot be removed.

*boolean isSupported*
If true, profile is supported on this system.

**Setting settings []**
Array of settings for a profile.

*uint32 type*

*DisplayGridInfo grid*
information about a display grid

Detailed Description

**Display** profile.

Member Function Documentation

string info ()

formats basic information about a profile into a human-readable string
boolean restoreSettings ( [in] uint32 settingIds[] )

Restore or delete settings, specified by IDs. Predefined settings get restored and non-predefined settings get deleted.

**Parameters:**

*settingIds* Array of **Setting** IDs

boolean setBinaryValueById ( [in] uint32 settingId, [in] uint8 value[] )

Set the binary value by a setting ID.

**Parameters:**

*settingId* **Setting** ID

*value* Binary value as an array of bytes

boolean setStringValueById ( [in] uint32 settingId, [in] string value )

Set the string value by a setting ID.

**Parameters:**

*settingId* **Setting** ID

*value* String value

boolean setValueById ( [in] uint32 settingId, [in] uint32 value )

Set the 32-bit value by a setting ID.

**Parameters:**

*settingId* **Setting** ID
Member Data Documentation

**DisplayGridInfo grid**

Information about a display grid

**uint32 id**

Unique id of a profile, used as a key.

**boolean isPredefined**

If true, profile is predefined and cannot be removed.

**boolean isSupported**

If true, profile is supported on this system.

**string name**

Name of a profile.

**Setting settings**

Array of settings for a profile.

**uint32 type**

Display Profile type. Possible values are: 4 - Display

**Version verClass**

Version of a profile class
The documentation for this class was generated from the following file:

- nvwmi.mof
Ecc Class Reference

Represents the GPU Error Correction Code. This feature is not supported if there are multiple GPU topologies enabled. More...

List of all members.

Public Member Functions

boolean resetCounters ([in]boolean bResetCurrent,[in]boolean bResetAggregate)
Resets memory error counters. This method will fail if option=0 (notSupported).

boolean setConfiguration ([in]boolean bEnable,[in]boolean bEnableImmediately)
Updates the ECC memory configuration settings. This method will fail if option=0 (notSupported).

string info ()
Formats basic information about ECC objects into a human-readable string.

Public Attributes

Version verClass
  uint32 id
  Unique identification of the NVIDIA GPU.

boolean isSupported
  Is ECC memory feature supported?

boolean isEnabled
  Is ECC memory setting enabled?

boolean isWritable
  Is current ECC configuration stored in non-volatile memory?

boolean isEnabledByDefault
  Is factory default ECC configuration enabled?

uint32 option
  Possible ECC memory configuration options. Possible values are: 0 - notSupported, 1 - deferred, 2 - immediate.

uint64 currentSingleBitErrors
  Number of single bit ECC errors detected since last boot.

uint64 currentDoubleBitErrors
  Number of double bit ECC errors detected since last boot.

uint64 aggregateSingleBitErrors
  Number of single bit ECC errors detected since last counter reset.

uint64 aggregateDoubleBitErrors
  Number of double bit ECC errors detected since last counter reset.

Detailed Description

Represents the GPU Error Correction Code. This feature is not supported if there are multiple GPU topologies enabled.
**Member Function Documentation**

string info ( )

Formats basic information about ECC objects into a human-readable string.

boolean resetCounters ( [in] boolean bResetCurrent,
                        [in] boolean bResetAggregate )

Resets memory error counters. This method will fail if option=0 (notSupported).

**Parameters:**
- **bResetCurrent**     Reset the current ECC error counters
- **bResetAggregate**   Reset the aggregate ECC error counters

boolean setConfiguration ( [in] boolean bEnable,
                          [in] boolean bEnableImmediately )

Updates the ECC memory configuration settings. This method will fail if option=0 (notSupported).

**Parameters:**
- **bEnable**            Enable new ECC configuration setting
- **bEnableImmediately** New configuration setting should take effect immediately. This flag is valid only if option=2 (immediate).

---

**Member Data Documentation**

uint64 aggregateDoubleBitErrors
Number of double bit ECC errors detected since last counter reset.

`uint64 aggregateSingleBitErrors`

Number of single bit ECC errors detected since last counter reset.

`uint64 currentDoubleBitErrors`

Number of double bit ECC errors detected since last boot.

`uint64 currentSingleBitErrors`

Number of single bit ECC errors detected since last boot.

`uint32 id`

Unique identification of the NVIDIA GPU.

`boolean isEnabled`

Is ECC memory setting enabled?

`boolean isEnabledByDefault`

Is factory default ECC configuration enabled?

`boolean isSupported`

Is ECC memory feature supported?

`boolean isWritable`

Is current ECC configuration stored in non-volatile memory?

`uint32 option`

Possible ECC memory configuration options. Possible values are: 0 - notSupported, 1 - deferred, 2 - immediate.
Version verClass

Ecc class version

The documentation for this class was generated from the following file:

- nvwmi.mof

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
EncoderSessions Class Reference

Represents encoder sessions info. More...

List of all members.

Public Member Functions

```cpp
string info ()
    Formats basic information about per process utilization into a human-readable string.
```

Public Attributes

```cpp
Version verClass
```
uint32 id
    Unique ID.
uint32 sessionsCount
    Total no of sessions.
uint32 sessionId []
    Session Id.
uint32 processId []
    Owning process ID.
uint32 vgpuInstance []
    Owning vGPU instance ID (only valid on vGPU hosts, otherwise zero).
uint32 codecType []
    Video encoder type.. Possible values are: 0 - Unknown, 1 - H264, 2 - HEVC.
uint32 hResolution []
    Current encode horizontal resolution.
uint32 vResolution []
    Current encode vertical resolution.
uint32 averageEncodeFps []
    Moving average encode frames per second.
uint32 averageEncodeLatency []
    Moving average encode latency in milliseconds.

---

Detailed Description

Represents encoder sessions info.

---

Member Function Documentation

string info ()

Formats basic information about per process utilization into a human-readable string.
**Member Data Documentation**

`uint32 averageEncodeFps[]`
Moving average encode frames per second.

`uint32 averageEncodeLatency[]`
Moving average encode latency in milliseconds.

`uint32 codecType[]`
Video encoder type. Possible values are: 0 - Unknown, 1 - H264, 2 - HEVC.

`uint32 hResolution[]`
Current encode horizontal resolution.

`uint32 id`
Unique ID.

`uint32 processId[]`
Owning process ID.

`uint32 sessionId[]`
Session Id.

`uint32 sessionsCount`
Total no of sessions.

**Version verClass**
**EncoderSessions** class version.

`uint32 vgpuInstance[]`

Owning vGPU instance ID (only valid on vGPU hosts, otherwise zero).

`uint32 vResolution[]`

Current encode vertical resolution.

The documentation for this class was generated from the following file:

- `nvwmi.mof`

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Gpu Class Reference

Represents an NVIDIA GPU. More...

List of all members.

Public Member Functions

boolean **enableLicensedFeature** ([in]string feature)
Enable licensed feature with given name.

boolean **disableLicensedFeature** ([in]string feature)
Disable licensed feature with given name.

boolean **fakeEDID** ([in]string filePath,[in,
ValueMap{"-1","0","1","2","3","4","5","6","7","8","9"}
Values{"unknown","uninitialized","VGA","Component","S-Video","HDMI","DVI","LVDS","DP","Composite","All"}
output)
Fake given EDID on all display outputs or on spec
display outputs, of the given GPU.

boolean **fakeEDIDOnPort** ([in]string filePath,[in]uint32 port
  [in, ValueMap{-1,"0","1","2","3","4","5","6","7","8"}
   Values{"unknown","uninitialized","VGA","Component","S-
   Video","HDMI","DVI","LVDS","DP","Composite"}] u
   output)
Fake given EDID on specified display output, of the given GPU port.

boolean **createUtilizationEvent** ([in]sint32 lower,[in]sint32
   upper,[in, ValueMap{-1,"0","1","2","3"},
   Values{"All","GPU","Frame Buffer","Video Engine",
   bus"}] sint32 domain)
Create utilization event with given bounds in given performance domain.

boolean **deleteUtilizationEvents** ([in,
   ValueMap{-1,"0","1","2","3"},
   Values{"All","GPU","Frame Buffer","Video Engine",
   bus"}] sint32 domain)
Delete all utilization events in given performance domain.

string **getAllUtilizationEvents** ([in,
   ValueMap{-1,"0","1","2","3"},
   Values{"All","GPU","Frame Buffer","Video Engine",
   bus"}] sint32 domain)
Get all utilization events in given performance domain.

string **info** ()
Formats basic information about this GPU into a human-readable string.

---

**Public Attributes**

**Version** ver

**uint32 id**
Unique id of named object, used as a key.

**string name**
Name of an object.

**string** `uname`
Unique name of an object.

**sint32** `ordinal`
Ordinal number of named object among objects with same names.

**sint32** `count`
Total number of named objects with same name.

**uint32** `handle`
**uint32** `nvapild`
NVAPI ID of the GPU.

**uint32** `productType`
GPU product type. Possible values are: 0 - unknown, 1 - GeForce, 2 - Quadro, 3 - NVS, 4 - Tesla.

**uint32** `memoryType`
The type of memory associated with this GPU. Possible values are: 0 - unknown, 1 - SDRAM, 2 - DDR1, 3 DDR2, 4 - GDDR2, 5 - GDDR3, 6 - GDDR4, 7 - DDR, 8 - GDDR5, 9 - LPDDR2, 10 - GDDR5X, 11 - HBM, 12 - HBM2.

**uint32** `memorySizeVirtual`
The virtual memory size in MB (Physical Video Memory + System Video Memory + Shared System Memory).

**uint32** `memorySizePhysical`
The physical video memory size in MB.

**uint32** `memorySizeAvailable`
The currently available physical video memory size in MB.

**uint32** `memoryBusWidth`
The width of the GPU's RAM memory bus in MB.

**uint32** `coreCount`
The total number of Cores defined for a GPU.

**sint32** `memoryClockCurrent`
The current memory clock frequency in MHz, 0 if not available, -1 if not supported in current environment.

**sint32** `gpuCoreClockCurrent`
The current GPU clock frequency in MHz, 0 if not
available, -1 if not supported in current environment.

**PcieLink pcieGpu**
Parameters of PCI-Express link to this GPU.

**sint32 pcieDownstreamWidth**
Width of the PCI-Express downstream link, in count of PCI-Express lanes.

**VideoCodec videoCodec**
Parameters of Video Engine encoder and decoder this GPU.

**real32 power**
The total GPU power drawn over last iteration of sampling in watts. -1 if not supported in current environment.

**sint32 powerSampleCount**
The number of power sensor samples per iteration not supported in current environment.

**sint32 powerSamplingPeriod**
The period (in milliseconds) between the power samples. -1 if not supported in current environment.

**sint32 percentGpuUsage**
The percentage of GPU utilization. -1 if not supported in current environment.

**sint32 percentGpuMemoryUsage**
The percentage of GPU memory utilization. -1 if not supported in current environment.

**ThermalProbe ref thermalProbes []**
Array of thermal probes of this GPU.

**Cooler ref coolers []**
Array of references to coolers of this GPU.

**Ecc ref ecc []**
Array of references to Error Correction Code instances on this GPU.

**string productName**
The name of the graphics card to which this GPU belongs.

**Version verVBIOS**
The video BIOS version associated with this GPU.
string **deviceInfo**
The display adapter string comprising of Vendor Id, Device Id, Sub-device Id, Revision Id.

string **archName**
The name of the GPU architecture - 'Curie', 'Tesla', 'Fermi', 'Tegra', 'Kepler', 'Maxwell', 'Volta'

sint32 **archId**
Numerical ID of the GPU architecture. Possible values are: -1 - unknown, 0 - Legacy, 1 - Curie, 2 - Tesla, Fermi, 4 - Tegra, 5 - Kepler, 6 - Maxwell, 7 - Pascal, Volta

string ** licensableFeatures**
Licensable features.

sint32 **licensableStatus**
Status of licensable features. Possible values are: N/A, 0 - Disabled, 1 - Enabled.

**EncoderSessions ref encoderSessionsInfo[]**
Array of references to encoder sessions info.

**ProcessUtilization ref processUtilization[]**
Array of references to per process utilization info for GPU.

---

**Detailed Description**

Represents an NVIDIA GPU.

---

**Member Function Documentation**

```cpp
boolean createUtilizationEvent(
    [in] sint32 lower,
    [in] sint32 upper,
    [in, ValueMap{"-1","0","1","2","3"}, Values{"All","GPU","Frame Buffer","Video Engine","PCIe bus"}] domain
)```

Create utilization event with given bounds in given performance domain.

Parameters:

- **lower** Lower bound. When utilization value drops below this value, event notification will be posted to the OS event log
- **upper** Upper bound. When utilization value grows above this value, event notification will be posted to the OS event log
- **domain** Performance domain of the GPU. "All" means that given bounds will trigger notification events for all domains

```
sint32
    )

Delete all utilization events in given performance domain.

Parameters:

- **domain** Performance domain of the GPU. "All" means that all event triggers for all domains will be removed

```
boolean
    deleteUtilizationEvents ( [in, ValueMap{"-1","0","1","2","3"},
        Values{"All","GPU","Frame Buffer","Video Engine","PCIe bus"}] domain )

Disable licensed feature with given name.

Parameters:

- **feature** Licensed feature to disable

```
boolean
disableLicensedFeature ( [in] string feature )

Enable licensed feature with given name.

Parameters:

- **feature** Licensable feature to enable

```
boolean
    enableLicensedFeature ( [in] string feature )
boolean fakeEDID ( [in] string filePath, [in, ValueMap{-1,"0","1","2","3","4","5","6","7","8","9"}, Values{"unknown","uninitialized","VGA","Component","S-Video","HDMI","DVI","LVDS","DP","Composite","All"}] output uint32 )

Fake given EDID on all display outputs or on specified display outputs, of the given GPU.

Parameters:
filePath Full path to the file with the custom EDID. In order to remove the forced EDID, specify an empty string
output Display output type for faking EDID. In order to force/remove the EDID on all display outputs, specify the output value "All"

boolean fakeEDIDOnPort ( [in] string filePath, [in] uint32 portIndex, [in, ValueMap{-1,"0","1","2","3"}, Values{"unknown","uninitialized","VGA","Component","S-Video","HDMI","DVI","LVDS","DP","Composite"}] output uint32 )

Fake given EDID on specified display output, of the given GPU port.

Parameters:
filePath Full path to the file with the custom EDID. In order to remove the forced EDID, specify an empty string
portIndex Physical location of the display port on the GPU.
output Display output type for faking EDID.

string domain ( [in, ValueMap{-1,"0","1","2","3"}, Values{"All","GPU","Frame domain"}])
getAllUtilizationEvents   Buffer","Video Engine","PCIe bus"]
sint32

Get all utilization events in given performance domain.

**Parameters:**

`domain` Performance domain of the GPU. "All" means that all event triggers for all domains will be listed

string info ()

Formats basic information about this GPU into a human-readable string.

---

**Member Data Documentation**

sint32 archId

numerical ID of the GPU architecture. Possible values are: -1 - unknown, 0 - Legacy, 1 - Curie, 2 - Tesla, 3 - Fermi, 4 - Tegra, 5 - Kepler, 6 - Maxwell, 7 - Pascal, 8 - Volta

string archName

The name of the GPU architecture - 'Curie','Tesla','Fermi','Tegra','Kepler','Maxwell','Volta' etc.

**Cooler** ref coolers[]

Array of references to coolers of this GPU.

uint32 coreCount

The total number of Cores defined for a GPU.

sint32 count
Total number of named objects with same name.

string **deviceInfo**

The display adapter string comprising of Vendor Id, Device Id, Sub-device Id, Revision Id.

**Ecc** ref **ecc[]**

Array of references to Error Correction Code instances on this GPU.

**EncoderSessions** ref **encoderSessionsInfo[]**

Array of references to encoder sessions info.

sint32 **gpuCoreClockCurrent**

The current GPU clock frequency in MHz, 0 if not available, -1 if not supported in current environment.

uint32 **handle**

same as NVAPI ID of the GPU. Unique ID - deprecated, please use 'id'

uint32 **id**

Unique id of named object, used as a key.

string **licensableFeatures[]**

Licensable features.

sint32 **licensableStatus[]**

Status of licensable features. Possible values are: -1 - N/A, 0 - Disabled, 1 - Enabled.

uint32 **memoryBusWidth**
The width of the GPU's RAM memory bus in MB.

`sint32 memoryClockCurrent`

The current memory clock frequency in MHz, 0 if not available, -1 if not supported in current environment.

`uint32 memorySizeAvailable`

The currently available physical video memory size in MB.

`uint32 memorySizePhysical`

The physical video memory size in MB.

`uint32 memorySizeVirtual`

The virtual memory size in MB (Physical Video Memory + System Video Memory + Shared System Memory).

`uint32 memoryType`

The type of memory associated with this GPU. Possible values are: 0 - unknown, 1 - SDRAM, 2 - DDR1, 3 - DDR2, 4 - GDDR2, 5 - GDDR3, 6 - GDDR4, 7 - DDR3, 8 - GDDR5, 9 - LPDDR2, 10 - GDDR5X, 11 - HBM1, 12 - HBM2.

`string name`

Name of an object.

`uint32 nvapild`

NVAPI ID of the GPU.

`sint32 ordinal`

Ordinal number of named object among objects with same names.
sint32 **pcieDownstreamWidth**

Width of the PCI-Express downstream link, in count of PCI-Express lanes.

**PcieLink pcieGpu**

Parameters of PCI-Express link to this GPU.

sint32 **percentGpuMemoryUsage**

The percentage of GPU memory utilization. -1 if not supported in current environment.

sint32 **percentGpuUsage**

The percentage of GPU utilization. -1 if not supported in current environment.

real32 **power**

The total GPU power drawn over last iteration of sampling in watts. -1 if not supported in current environment.

sint32 **powerSampleCount**

The number of power sensor samples per iteration. -1 if not supported in current environment.

sint32 **powerSamplingPeriod**

The period (in milliseconds) between the power samples. -1 if not supported in current environment.

**ProcessUtilization** ref **processUtilization[]**

Array of references to per process utilization info for this GPU.
string **productName**

The name of the graphics card to which this GPU belongs.

uint32 **productType**

GPU product type. Possible values are: 0 - unknown, 1 - GeForce, 2 - Quadro, 3 - NVS, 4 - Tesla.

**ThermalProbe** ref **thermalProbes**[]

Array of thermal probes of this GPU.

string **uname**

Unique name of an object.

**Version** **ver**

**Version** of named object

**Version** **verVBIOS**

The video BIOS version associated with this GPU.

**VideoCodec** **videoCodec**

Parameters of Video Engine encoder and decoder in this GPU.

The documentation for this class was generated from the following file:

- nvwmi.mof

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
NamedObject Class Reference

Base class for all named NV objects. More...

List of all members.

Public Attributes

Version ver
  uint32 id
    Unique id of named object, used as a key.
  string name
    Name of an object.
  string uname
    Unique name of an object.
  sint32 ordinal
    Ordinal number of named object among objects with same names.
**sint32 count**
Total number of named objects with same name.

---

**Detailed Description**

Base class for all named NV objects.

---

**Member Data Documentation**

**sint32 count**
Total number of named objects with same name.

**uint32 id**
Unique id of named object, used as a key.

**string name**
Name of an object.

**sint32 ordinal**
Ordinal number of named object among objects with same names.

**string uname**
Unique name of an object.

**Version ver**

*Version* of named object

---

The documentation for this class was generated from the following file:
- nvwmimof

---

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
OverlapLimits Class Reference

Represents overlap limits of individual display in a display grid. More...

List of all members.

Public Member Functions

```c
string info ()
```

Formats basic information about `OverlapLimits` instance into a human-readable string.

Public Attributes

```c
Version verClass
uint32 id
```

Unique identification of the overlap limits.
sint32 minOverlapCol
   The minimum column overlap for each display (in pixels).
sint32 maxOverlapCol
   The maximum column overlap for each display (in pixels).
sint32 minOverlapRow
   The minimum row overlap for each display (in pixels).
sint32 maxOverlapRow
   The maximum row overlap for each display (in pixels).
sint32 minTotalCol
   The minimum total column overlap for the entire grid (in pixels).
sint32 maxTotalCol
   The maximum total column overlap for the entire grid (in pixels).
sint32 minTotalRow
   The minimum total row overlap for the entire grid (in pixels).
sint32 maxTotalRow
   The maximum total row overlap for the entire grid (in pixels).

---

**Detailed Description**

Represents overlap limits of individual display in a display grid.

---

**Member Function Documentation**

string info ( )

Formats basic information about *OverlapLimits* instance into a human-readable string.

---

**Member Data Documentation**

uint32 id

Unique identification of the overlap limits.
sint32 **maxOverlapCol**

The maximum column overlap for each display (in pixels).

sint32 **maxOverlapRow**

The maximum row overlap for each display (in pixels).

sint32 **maxTotalCol**

The maximum total column overlap for the entire grid (in pixels).

sint32 **maxTotalRow**

The maximum total row overlap for the entire grid (in pixels).

sint32 **minOverlapCol**

The minimum column overlap for each display (in pixels).

sint32 **minOverlapRow**

The minimum row overlap for each display (in pixels).

sint32 **minTotalCol**

The minimum total column overlap for the entire grid (in pixels).

sint32 **minTotalRow**

The minimum total row overlap for the entire grid (in pixels).

**Version verClass**

Object version

The documentation for this class was generated from the following file:
• nvwmi.mof

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
PcieLink Class Reference

Represents parameters of PCI Express bus link. More...

List of all members.

Public Member Functions

```plaintext
string info ()
```
Formats basic information about this PCI-Express link into a human-readable string.

Public Attributes

```plaintext
Version verClass
sint32 maxSpeed
```
Maximum speed attainable on this link in Mbits per second.
Detailed Description

Represents parameters of PCI Express bus link.

Member Function Documentation

    string info ( )

Formats basic information about this PCI-Express link into a human-readable string.

Member Data Documentation

    sint32 curGen

Currently negotiated generation of the PCIe bus protocol.

    sint32 curSpeed

Current speed on this link in Mbits per second.

    sint32 curWidth

Current width, in # of PCI-Express lanes.
Current width, in # of PCI-Express lanes.

sint32 **maxGen**

Maximum supported generation of the PCIe bus protocol.

sint32 **maxSpeed**

Maximum speed attainable on this link in Mbits per second.

sint32 **maxWidth**

Maximum width, in # of PCI-Express lanes.

**Version verClass**

Object version

The documentation for this class was generated from the following file:

- nvwmi.mof
ProcessUtilization Class Reference

Represents per process GPU encoder utilization values. More...

List of all members.

Public Member Functions

    string info ()
    Formats basic information about per process utilization into a human-readable string.

Public Attributes

    Version verClass
uint32 id
  Unique ID.

uint32 processCount
  Total no of process.

uint32 pid []
  Process Id.

string time_stamp []
  CPU Timestamp.

uint32 smUtilization []
  SM Utilization value.

uint32 memUtilization []
  Mem Utilization value.

uint32 encUtilization []
  Enc Utilization value.

uint32 decUtilization []
  Dec Utilization value.

---

**Detailed Description**

Represents per process GPU encoder utilization values.

---

**Member Function Documentation**

string info ( )

Formats basic information about per process utilization into a human-readable string.

---

**Member Data Documentation**

uint32 decUtilization[]

Dec Utilization value.
The documentation for this class was generated from the following file:

- nvwmi.mof
Profile Class Reference

Global profile class. [More...]

[List of all members.]

Public Member Functions

boolean **setValueById** ([in]uint32 settingId,[in]uint32 value)
Set the 32-bit value by a setting ID.

boolean **setBinaryValueById** ([in]uint32 settingId,[in]uint8 value[])
Set the binary value by a setting ID.

boolean **setStringValueById** ([in]uint32 settingId,[in]string value)
Set the string value by a setting ID.

boolean **restoreSettings** ([in]uint32 settingIds[])  
Restore or delete settings, specified by IDs. Predefined settings get restored and non-predefined settings get deleted.

string **info** ()
Public Attributes

Version verClass
  uint32 id
    Unique id of a profile, used as a key.
  string name
    Name of a profile.
  uint32 type
    Type of global profile. Possible values are: 1 - 3D global, 3 - nView global, 5 - system.
  boolean isPredefined
    If true, profile is predefined and cannot be removed.
  boolean isSupported
    If true, profile is supported on this system.

Setting settings []
  Array of settings for a profile.

Detailed Description

Global profile class.

Member Function Documentation

string info ( )

formats basic information about a profile into a human-readable string

boolean restoreSettings ( [in] uint32 settingIds[] )

Restore or delete settings, specified by IDs. Predefined settings get
restored and non-predefined settings get deleted.

**Parameters:**

- `settingIds` Array of **Setting** IDs

```c
boolean setBinaryValueById ([in] uint32 settingId,
                          [in] uint8 value[])
```

Set the binary value by a setting ID.

**Parameters:**

- `settingId` **Setting** ID
- `value` Binary value as an array of bytes

```c
boolean setStringValueById ([in] uint32 settingId,
                            [in] string value)
```

Set the string value by a setting ID.

**Parameters:**

- `settingId` **Setting** ID
- `value` String value

```c
boolean setValueById ([in] uint32 settingId,
                     [in] uint32 value)
```

Set the 32-bit value by a setting ID.

**Parameters:**

- `settingId` **Setting** ID
- `value` 32-bit value
Member Data Documentation

uint32 id

Unique id of a profile, used as a key.

boolean isPredefined

If true, profile is predefined and cannot be removed.

boolean isSupported

If true, profile is supported on this system.

string name

Name of a profile.

Setting settings[]

Array of settings for a profile.

uint32 type

Type of global profile. Possible values are: 1 - 3D global, 3 - nView global, 5 - system.

Version verClass

Version of a profile class

The documentation for this class was generated from the following file:

- nvwmi.mof
ProfileManager Class Reference

Represents root object for performing Profile related task. More...

List of all members.

Public Member Functions

string getAllProfiles ([in, ValueMap{"0","1","3","4","5"}, Values{"3D Application Profile","3D Global Profile","nView Profile","Display Profile","System Profile"}] uint32 type)
Get name and id of all profiles for a given profile type. The return value is a string in format profile id: profile name e.g.profile1 id: profile 1 name; profile2 id: profile 2 name;.

boolean restoreDefaults3D ()
Restore all profiles to default.

boolean setCurrentProfile3D ([in]string name)
Set current profile.
boolean **setVSync** ([in, ValueMap{"0","1","2","3","4"}, Values{"Passive","Off","On","Adaptive","Adaptive half"}] uint32 vsyncMode)
Set mode of vertical synchronization in the current global profile.

boolean **loadDesktopProfile** ([in]string name)
Load a selected profile into nView Desktop Manager.

boolean **lockDesktopProfile** ([in]string name,[in]uint32 lock)
Lock a nView Desktop profile, once locked the setting in this profile will not be modified through nView Dekstop Manager.

boolean **saveDesktopProfile** ([in]string name)
Save nView Desktop Manager settings to the profile.

boolean **deleteDesktopProfile** ([in]string name)
Delete the nView Desktop profile.

boolean **saveSystemProfile** ([in]string name)
Save system profile.

boolean **applySystemProfile** ([in]string name)
Apply system profile.

boolean **saveDisplayProfiles** ([in]string prefix)
Save current display state in display profiles with given prefixes. One display profile per DisplayGrid. DisplayGrid might contain more than one physical display.

boolean **applyDisplayProfiles** ([in]string prefix)
Apply saved state in display profiles with given prefixes. One display profile per DisplayGrid.

boolean **createProfile** ([in]string name,[in]uint32 type,[in]string params)
Create a new Profile.

boolean **restoreProfile** ([in]string name)
Restore/Delete the Profile. Predefined Profile gets restored and non-predefined Profile gets deleted.

string **info** ()
Formats basic information about Profile Manager into a human-readable string.

**Public Attributes**
**Detailed Description**

Represents root object for performing Profile related task.

**Member Function Documentation**

boolean applyDisplayProfiles ( [in] string prefix )

Apply saved state in display profiles with given prefixes. One display profile per DisplayGrid.

**Parameters:**

prefix Common prefix - e.g. "my display" Ordinals will be added per every display profile (e.g. "my display 1 of 4", "my display 2 of 4" etc.)

boolean applySystemProfile ( [in] string name )

Apply system profile.

**Parameters:**
name Name of system profile to be applied

boolean createProfile ( [in] string name,
                      [in] uint32 type,
                      [in] string params )

Create a new Profile.

Parameters:

- **name** Name of the profile
- **type** Type of the profile. 0 - 3D application profile
- **params** Additional parameters separated by semicolon. For a 3D Application Profile specify a minimal path to the binary which uses Direct3D or OpenGL, e.g. "Application1.exe;Application2.exe". Note that white space at the beginning or end of a path will be trimmed and empty entries discarded.

boolean deleteDesktopProfile ( [in] string name )

Delete the nView Desktop profile.

Parameters:

- **name** Name of the existing profile which will be deleted

string getAllProfiles ( [in, ValueMap{"0","1","3","4","5"}, Values{"3D Application Profile","3D Global Profile","nView Profile","Display Profile","System Profile"}] uint32 type )

Get name and id of all profiles for a given profile type. The return value is a string in format profile id: profile name e.g.profile1 id: profile 1 name; profile2 id: profile 2 name;.

Parameters:

- **type** Type of the profile
string info ( )

Formats basic information about Profile Manager into a human-readable string.

boolean loadDesktopProfile ([in] string name)

Load a selected profile into nView Desktop Manager.

Parameters:
    name Name of existing profile to be loaded into nView Desktop Manager

boolean lockDesktopProfile ([in] string name, [in] uint32 lock)

Lock a nView Desktop profile, once locked the setting in this profile will not be modified through nView Desktop Manager.

Parameters:
    name Name of the existing profile which needs to be locked
    lock Parameter to lock/unlock the profile. Pass 1 to indicate lock and pass 0 to unlock the profile

boolean restoreDefaults3D ( )

Restore all profiles to default.

boolean restoreProfile ([in] string name)

Restore/Delete the Profile. Predefined Profile gets restored and non-predefined Profile gets deleted.

Parameters:
    name Profile name
boolean saveDesktopProfile ([in] string name )

Save nView Desktop Manager settings to the profile.

**Parameters:**

- **name** Name of profile where the nView Desktop Manager settings will be saved

boolean saveDisplayProfiles ([in] string prefix )

Save current display state in display profiles with given prefixes. One display profile per **DisplayGrid**. **DisplayGrid** might contain more than one physical display.

**Parameters:**

- **prefix** Common prefix - e.g. "my display". Ordinals will be added per every display profile (e.g. "my display 1 of 4", "my display 2 of 4" etc.)

boolean saveSystemProfile ([in] string name )

Save system profile.

**Parameters:**

- **name** Name of system profile to be saved

boolean setCurrentProfile3D ([in] string name )

Set current profile.

**Parameters:**

- **name** Name of existing profile to set as current

boolean setVSync ([in, Values{"Passive"","Off","On","Adaptive","Adaptive
half"}] uint32 vsyncMode )
Set mode of vertical synchronization in the current global profile.

**Parameters:**

\[\textit{vsyncMode}\] V-Sync mode to be applied to the current global profile values might be Application-Controlled (same as Passive) = 0, Off = 1, On = 2, Adaptive = 3 and Adaptive (half refresh rate) = 4

---

**Member Data Documentation**

string `currentProfile3D`

Current Global 3D profile (D3D, OGL).

string `currentProfileDesktop`

Current Desktop Profile (nView).

string `currentSystemProfile`

Current system profile.

string `defaultProfile3D`

Default Global 3D profile (D3D, OGL).

string `defaultProfileDesktop`

Default Desktop Profile (nView).

string `defaultSystemProfile`

Default system profile.

**Version** `verClass`
Profile Manager class version

The documentation for this class was generated from the following file:

- nvwmi.mof

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Setting Class Reference

Profile setting. More...

List of all members.

Public Member Functions

string getStringValue ()
Get the string value for a setting.

string info ()
Formats basic information about the Profile setting object into a human-readable string.

Public Attributes

Version verClass
uint32 id
    Unique id of a setting, used as a key.
uint32 type
    Type of value of the setting. Possible values are: 0 - invalid, 1 - uint32, 2 - sint32, 3 - string, 4 - binary.
uint8 value []
    Value for the Setting.

---

Detailed Description

Profile setting.

---

Member Function Documentation

string getStringValue ( )
Get the string value for a setting.

string info ( )
Formats basic information about the Profile setting object into a human-readable string.

---

Member Data Documentation

uint32 id
Unique id of a setting, used as a key.

uint32 type
Type of value of the setting. Possible values are: 0 - invalid, 1 - uint32, 2 - sint32, 3 - string, 4 - binary.
uint8 value[]

Value for the Setting.

Version verClass

Version of a Setting class

The documentation for this class was generated from the following file:

- nvwmi.mof
SettingInfo Class Reference

Useful information about a profile setting. More...

List of all members.

Public Member Functions

```cpp
string info ()
formats information about a setting into a human-readable string
```

Public Attributes

```cpp
Version verClass
uint32 id
    Unique id of a setting, used as a key.
string name
```
Setting name.
string description
Setting description.

---

**Detailed Description**

Useful information about a profile setting.

---

**Member Function Documentation**

string info ()

formats information about a setting into a human-readable string

---

**Member Data Documentation**

string description

Setting description.

uint32 id

Unique id of a setting, used as a key.

string name

Setting name.

Version verClass

Version of a SettingInfo class

---

The documentation for this class was generated from the following file:
• nvwmi.mof
SettingTable Class Reference

Table which describes profile settings of certain type. More...

List of all members.

Public Member Functions

uint32 **getIdFromName** ([in]string name)
get setting ID from setting name

string **getNameFromId** ([in]uint32 id)
get setting name from setting ID

string **infoByName** ([in]string name)
formats information about given setting with given name

string **infoById** ([in]uint32 id)
formats information about given setting with given ID

string **info** ()
formats basic information about all settings into a human-
Public Attributes

Version verClass
  uint32 id
    Unique id of a setting table, used as a key.
  uint32 type
    Profile type for settings in this table. Possible values are: 0 - 3D Application, 1 - 3D Global, 2 - nView Application, 3 - nView Global, 4 - Display, 5 - System.
  uint32 settingIds []
    Array of all setting IDs, described in this table.
  string settingNames []
    Array of all setting names, described in this table.

SettingInfo settings []
  Array of settings, described in this table.

Detailed Description

Table which describes profile settings of certain type.

Member Function Documentation

uint32 getIdFromName ( [in] string name )

get setting ID from setting name

Parameters:
  name Setting name

string getNameFromId ( [in] uint32 id )
get setting name from setting ID

**Parameters:**

\[ id \text{ Setting ID} \]

string info()  
formats basic information about all settings into a human-readable string

string infoById ([in] uint32 \[id\] )  
formats information about given setting with given ID

**Parameters:**

\[ id \text{ Setting ID} \]

string infoByName ([in] string \[name\] )  
formats information about given setting with given name

**Parameters:**

\[ name \text{ Setting name} \]

---

**Member Data Documentation**

uint32 \[id\]

Unique id of a setting table, used as a key.

uint32 \[settingIds\][]

Array of all setting IDs, described in this table.

string \[settingNames\][]

Array of all setting names, described in this table.
SettingInfo settings

Array of settings, described in this table.

uint32 type

Profile type for settings in this table. Possible values are: 0 - 3D Application, 1 - 3D Global, 2 - nView Application, 3 - nView Global, 4 - Display, 5 - System.

Version verClass

Version of a SettingTable class

The documentation for this class was generated from the following file:

- nvwmimof

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Sync Class Reference

Represents Sync-capable devices. More...

List of all members.

Public Member Functions

```cpp
boolean setSyncStateById ([in]uint32 syncDisplayIds[],
                          [in]uint32 syncState[])
Setup the synchronization between displays with given IDs.

boolean setSyncStateByName ([in]string syncDisplayNames,
                             [in]uint32 syncState[])
Setup the synchronization between displays with given Names.

boolean toggleSource ()
Toggle the sync signal source between internal and
external house sync.

boolean setPolarity (in, ValueMap{"0","1","2"}, Values{"Rising Edge","Falling Edge","Both Edges"}) uint32 polarity
Set synchronization signal polarity.

boolean setVmode (in, ValueMap{"0","1","2","3","4"},
Values{"none","TTL","NTSC-PAL-SECAM","HDTV","composite"}) uint32 vmode
Set video mode type.

boolean setInterval (in)uint32 interval
Set number of pulses to wait between framlock signal generation.

boolean setInterlaceMode (in, ValueMap{"0","1"},
Values{"Disabled","Enabled"}) uint32 interlaceMode
Set interlace mode for a Sync device.

boolean setSyncSkew (in)uint32 numOfPixels,(in)uint32 numOfLines
Set the amount of delay between the frame sync signal and the GPUs signal. Sync device must be in Client mode or if external Housesync signal present.

boolean setStartupDelay (in)uint32 numOfPixels,(in)uint32 numOfLines
Set the amount of delay the frame lock card should wait, until generating sync pulse. Sync device must be in Server mode.

string info ()
Formats basic information about Sync objects into a human-readable string.

Public Attributes

Version ver
uint32 id
Unique id of named object, used as a key.

string name
Name of an object.
string **uname**
Unique name of an object.

sint32 **ordinal**
Ordinal number of named object among objects with same names.

sint32 **count**
Total number of named objects with same name.

**Version** verSyncFirmware

uint32 **nvapild**
NVAPI ID of the **Sync** device.

**SyncTopology** ref **syncDisplays** []
Array of references to **SyncTopology** objects.

boolean **isSynced**
Is sync enabled on this device?

boolean **isHouseSync**
Is house sync present?

boolean **isStereoSynced**
Is stereo enabled?

uint32 **polarity**
Polarity of a synchronization signal. Possible values are: 0 - Rising Edge, 1 - Falling Edge, 2 - Both Edges.

uint32 **vmode**
Video mode. Possible values are: 0 - none, 1 - TTL, 2 - NTSC-PAL-SECAM, 3 - HDTV, 4 - composite.

uint32 **interval**
Number of pulses to wait between framelock signal generation.

uint32 **source**
Source of a synchronization signal. Possible values are: 0 - VSync, 1 - HouseSync.

uint32 **interlaceMode**
Interlace mode for a **Sync** device. Possible values are: 0 - Disabled, 1 - Enabled.

real32 **syncSignalRate**
**Sync** signal refresh rate in Hz. If the house sync is present and selected source is HouseSync, then it will
be the house sync signal refresh rate. Otherwise, it will be an internal display signal refresh rate.

```c
uint32 flStatus []
```

RJ-45 signal status. Possible values are: 0 - output, 1 - input, 2 - unused.

**SyncDelay** `syncSkew`

`syncSkew` object (The time delay between the frame sync signal and the GPUs signal). Available only in Client mode or if external Housesync signal is present.

**SyncDelay** `startupDelay`

`startupDelay` object (The amount of time the frame lock card should wait until generating sync pulse). Available only in Server mode.

---

### Detailed Description

Represents Sync-capable devices.

---

### Member Function Documentation

```c
string info ()

```

Formats basic information about **Sync** objects into a human-readable string.

```c
boolean setInterlaceMode [in, ValueMap{"0","1"},
 Values{"Disabled","Enabled"}] uint32
```

Set interlace mode for a **Sync** device.

**Parameters:**

- `interlaceMode` Interlace mode for a **Sync** device. Possible values are: 0 - Disabled, 1- Enabled.
boolean setInterval ([in] uint32 interval )

Set number of pulses to wait between framelock signal generation.

**Parameters:**

*interval* The number of pulses to skip for signal generation. 0 uses source, 1 gives half the frequency, and so on.

boolean setPolarity ([in, ValueMap{"0","1","2"}, Values{"Rising Edge","Falling Edge","Both Edges"}] uint32 polarity )

Set synchronization signal polarity.

**Parameters:**

*polarity* Possible values are: 0 - rising edge, 1 - falling edge, 2 - both


Set the amount of delay the frame lock card should wait, until generating sync pulse. **Sync** device must be in Server mode.

**Parameters:**

*numOfPixels* Number of pixels to induce startup delay
*numOfLines* Number of horizontal lines to induce startup delay


Set the amount of delay between the frame sync signal and the GPUs signal. **Sync** device must be in Client mode or if external Housesync signal present.

**Parameters:**

*numOfPixels* Number of pixels to induce sync skew
boolean setSyncStateByld ([in] uint32 syncDisplayIds[], [in] uint32 syncState[])

Setup the synchronization between displays with given IDs.

**Parameters:**
- **syncDisplayIds** Array of display IDs to synchronize(*SyncTopology.id*), separated by comma. Order is important. For Mosaic topologies, user can specify either a master display or all displays in a Mosaic topology. If user does not specify any displays, the method will choose the best topology to synchronize. To un-synchronize the displays, user will have to pass all the displays with displaySyncState as UnSynced.
- **syncState** Possible values are: 0 - UnSynced, 1 - Slave, 2 - Master

boolean setSyncStateByName ([in] string syncDisplayNames, [in] uint32 syncState[])

Setup the synchronization between displays with given Names.

**Parameters:**
- **syncDisplayNames** Display Names to synchronize. String of unique display names (*SyncTopology.uname*), separated by semicolon. Order is important. For Mosaic topologies, user can specify either a master display or all displays in a Mosaic topology. If user does not specify any displays, the method will choose the best topology to synchronize. To un-synchronize the displays,
user will have to pass all the displays with displaySyncState as UnSynced.

**syncState** Possible values are: 0 - UnSynced, 1 - Slave, 2 - Master

```c
boolean setVmode (in, ValueMap{"0","1","2","3","4"},
        (Values{"none","TTL","NTSC-PAL-SECAM","HDTV","composite"}) vmode )
```

Set video mode type.

**Parameters:**

vmode Possible values are: 0 - none, 1 - TTL, 2 - NTSC/PAL/SECAM, 3 - HDTV, 4 - composite

```c
boolean toggleSource ( )
```

Toggle the sync signal source between internal and external house sync.

---

**Member Data Documentation**

```c
sint32 count
```

Total number of named objects with same name.

```c
uint32 flStatus[]
```

RJ-45 signal status. Possible values are: 0 - output, 1 - input, 2 - unused.

```c
uint32 id
```

Unique id of named object, used as a key.

```c
uint32 interlaceMode
```

Interlace mode for a **Sync** device. Possible values are: 0 - Disabled, 1 -
Enabled.

uint32 interval
Number of pulses to wait between frameloop signal generation.

boolean isHouseSync
Is house sync present?

boolean isStereoSynced
Is stereo enabled?

boolean isSynced
Is sync enabled on this device?

string name
Name of an object.

uint32 nvapild
NVAPI ID of the Sync device.

sint32 ordinal
Ordinal number of named object among objects with same names.

uint32 polarity
Polarity of a synchronization signal. Possible values are: 0 - Rising Edge, 1 - Falling Edge, 2 - Both Edges.

uint32 source
Source of a synchronization signal. Possible values are: 0 - VSync, 1 -
HouseSync.

**SyncDelay startupDelay**

startupDelay object (The amount of time the frame lock card should wait until generating sync pulse). Available only in Server mode.

**SyncTopology** ref **syncDisplays[]**

Array of references to **SyncTopology** objects.

real32 **syncSignalRate**

**Sync** signal refresh rate in Hz. If the house sync is present and selected source is HouseSync, then it will be the house sync signal refresh rate. Otherwise, it will be an internal display signal refresh rate.

**SyncDelay syncSkew**

syncSkew object (The time delay between the frame sync signal and the GPUs signal). Available only in Client mode or if external Housesync signal is present.

string **uname**

Unique name of an object.

**Version ver**

**Version** of named object

**Version verSyncFirmware**

**Version** of the **Sync** board firmware

uint32 **vmode**

Video mode. Possible values are: 0 - none, 1 - TTL, 2 - NTSC-PAL-
SECAM, 3 - HDTV, 4 - composite.

The documentation for this class was generated from the following file:

- nvwmi.mof
SyncDelay Class Reference

Represents the Sync Delay (Sync Skew or Startup Delay). [More...](#)

[List of all members.](#)

Public Member Functions

```c
string info ()
```

Formats basic information about [SyncTopology](#) objects into a human-readable string.

Public Attributes

```
Version verClass
```

```c
uint32 id
```

Unique identification of [SyncDelay](#) objects.
uint32 delayType
Type of Sync Delay. Possible values are: 0 - unknown, 1 - SyncSkew, 2 - StartupDelay.

uint32 minPixels
Minimum number of pixels required at current display mode to induce sync delay.

uint32 maxLines
Maximum number of lines supported at current display mode to induce sync delay.

uint32 numOfPixels
Sync delay to be induced in unit pixels.

uint32 numOfLines
Sync delay to be induced in unit horizontal lines.

---

**Detailed Description**

Represents the Sync Delay (Sync Skew or Startup Delay).

---

**Member Function Documentation**

string info()  

Formats basic information about SyncTopology objects into a human-readable string.

---

**Member Data Documentation**

uint32 delayType

Type of Sync Delay. Possible values are: 0 - unknown, 1 - SyncSkew, 2 - StartupDelay.

uint32 id
Unique identification of **SyncDelay** objects.

**uint32 maxLines**

Maximum number of lines supported at current display mode to induce sync delay.

**uint32 minPixels**

Minimum number of pixels required at current display mode to induce sync delay.

**uint32 numOfLines**

**Sync** delay to be induced in unit horizontal lines.

**uint32 numOfPixels**

**Sync** delay to be induced in unit pixels.

**Version verClass**

**SyncDelay** class version

The documentation for this class was generated from the following file:

- nvwmi.mof

---

**NVIDIA**

*Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.*
SyncEvent Class Reference

Represent Sync events. More...

List of all members.

Public Attributes

**Version verClass**
uint32 **syncDeviceId**
Unique sync device ID.

uint32 **eventType**
Sync event type. Possible values are: 0 - Unknown, 1 - SyncLoss, 2 - SyncGain, 4 - StereoLoss, 8 - StereoGain, 0x10 - HouseSyncGain, 0x20 - HouseSyncLoss, 0x40 - FlGain, 0x80 - FlLoss, 0x10000 - SyncStateChanged.
Detailed Description

Represent **Sync** events.

Member Data Documentation

```markdown
uint32 **eventType**

**Sync** event type. Possible values are: 0 - Unknown, 1 - SyncLoss, 2 - SyncGain, 4 - StereoLoss, 8 - StereoGain, 0x10 - HouseSyncGain, 0x20 - HouseSyncLoss, 0x40 - FLGain, 0x80 - FLoss, 0x10000 - SyncStateChange.
```

```markdown
uint32 **syncDeviceId**

Unique sync device ID.
```

**Version verClass**

**Version** of the **SyncEvent** class

The documentation for this class was generated from the following file:

- nvwm.mof

---

**NVIDIA**

*Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.*
SyncTopology Class Reference

Represents the Sync Topology. More...

List of all members.

Public Member Functions

boolean isGPUSynced ()
   Returns true if GPU is synchronized (in phase) with the Sync device.

string info ()
   Formats basic information about SyncTopology objects into a human-readable string.

Public Attributes
**Version**

`uint32 id`  
Unique id of named object, used as a key.

`string name`  
Name of an object.

`string uname`  
Unique name of an object.

`sint32 ordinal`  
Ordinal number of named object among objects with same names.

`sint32 count`  
Total number of named objects with same name.

`uint32 displaySyncState`  
Can the current display be set as a sync master?

---

**Detailed Description**

Represents the **Sync** Topology.

---

**Member Function Documentation**

`string info()`  
Forms basic information about **SyncTopology** objects into a human-readable string.

`boolean isGPUSynced()`  
Returns true if GPU is synchronized (in phase) with the **Sync** device.

---

**Member Data Documentation**
sint32 **count**

Total number of named objects with same name.

uint32 **displaySyncState**

Synchronization state of a display in the **Sync** topology. Possible values are: 0 - UnSynced, 1 - Slave, 2 - Master

uint32 **id**

Unique id of named object, used as a key.

boolean **isDisplayMasterable**

Can the current display be set as a sync master?

string **name**

Name of an object.

sint32 **ordinal**

Ordinal number of named object among objects with same names.

string **uname**

Unique name of an object.

**Version** **ver**

**Version** of named object

The documentation for this class was generated from the following file:

- `nvwwmi.mof`
System Class Reference

Represents root object for NV-related HW and SW in the system. More...

List of all members.

Public Member Functions

boolean **setLicensingServer** ([in] string address,[in]uint16 port)
Change address of a licensing server.

boolean **unsetLicensingServer** ()
Clear licensing server address and forfeit license after reboot.

boolean **setnViewState** ([in, ValueMap{"0","1"},
Values{"Disable","Enable"}] uint32 state)
Set nView Desktop Manager State.

boolean **setLogState** ([in, BitMap{"0","1","2"},
BitValues{"debugger","file","eventLog"}] uint32 types,[in]uint32 filter,[in]uint32 options)
Change logging settings.

string enableEncoderSessionsPerfCounter ()
    Enable NVIDIA GPU Encoder Sessions performance counter.

string disableEncoderSessionsPerfCounter ()
    Disable NVIDIA GPU Encoder Sessions performance counter.

string enableProcessUtilizationPerfCounter ()
    Enable NVIDIA GPU Process Utilization performance counter.

string disableProcessUtilizationPerfCounter ()
    Disable NVIDIA GPU Process Utilization performance counter.

string info ()
    Formats basic information about System objects into a human-readable string.

Public Attributes

Version verClass
Version verNVWMI
    Version of the NVWMI driver.
Version verDisplayDriver
    Version of the NV display driver.
Version verSDICaptureFirmware
    Firmware version of the SDI Capture board.
Version verSDIOutputFirmware
    Firmware version of the SDI output board.
uint32 logTypes
    log types. Default is 5 (e.g. debugger and OS event log). 0 - disabled.
uint32 logFilter
    bitmask for filtering out messages. Recommended values are 0 - quiet, 16 - errors only, 272 - errors and warnings, 16777216 - performance profiling
uint32 logOptions
    reserved. Default is 0
uint16 licensingPort
    Network port on a licensing server. Default value is 0. When 0 is
specified, default server port value will be used.

string **licensingServer**

Network address of licensing server.

**Version verVBIOS**
The video BIOS version associated with the GPU's.

**Version vernViewDesktopManager**
The version of nView Desktop Manager.

**uint32 machineType**
Type of machine whether desktop or mobile. Possible values are: 0 - Desktop, 1 - Mobile.

**uint32 nViewState**
nView Desktop Manager current state. Possible values are: 0 - Disable, 1 - Enable, 2 - Not Installed

---

**Detailed Description**

Represents root object for NV-related HW and SW in the system.

---

**Member Function Documentation**

string **disableEncoderSessionsPerfCounter** ( )

Disable NVIDIA GPU Encoder Sessions performance counter.

string **disableProcessUtilizationPerfCounter** ( )

Disable NVIDIA GPU Process Utilization performance counter.

string **enableEncoderSessionsPerfCounter** ( )

Enable NVIDIA GPU Encoder Sessions performance counter.

string **enableProcessUtilizationPerfCounter** ( )
Enable NVIDIA GPU Process Utilization performance counter.

string info ()

Formats basic information about System objects into a human-readable string.

boolean setLicensingServer ([in] string address, [in] uint16 port)

Change address of a licensing server.

Parameters:

  address  Network address of licensing server
  port     Network port on licensing server. This is optional parameter and will be set to default value of 0 when omitted

boolean setLogState ([in, BitMap{"0","1","2"}], BitValues{"debugger","file","eventLog"}) uint32 types, [in] uint32 filter, [in] uint32 options)

Change logging settings.

Parameters:

  types  enabled log types. Default is 5 (e.g. debugger and OS event log)
  filter bitmask for filtering out messages. Recommended values are 0 - quiet, 16 - errors only, 272 - errors and warnings, 16777216 - performance profiling
  options optional parameter, reserved. Set to 0

boolean
setnViewState (Values{"Disable","Enable"}] uint32 state)

Set nView Desktop Manager State.

**Parameters:**

- `state` nView state to set

boolean unsetLicensingServer()

Clear licensing server address and forfeit license after reboot.

---

**Member Data Documentation**

- **uint16 licensingPort**

  Network port on a licensing server. Default value is 0. When 0 is specified, default server port value will be used.

- **string licensingServer**

  Network address of licensing server.

- **uint32 logFilter**

  bitmask for filtering out messages. Recommended values are 0 - quiet, 16 - errors only, 272 - errors and warnings, 16777216 - performance profiling

- **uint32 logOptions**

  reserved. Default is 0

- **uint32 logTypes**

  log types. Default is 5 (e.g. debugger and OS event log). 0 - disabled.
uint32 **machineType**

Type of machine whether desktop or mobile. Possible values are: 0 - Desktop, 1 - Mobile.

uint32 **nViewState**

nView Desktop Manager current state. Possible values are: 0 - Disable, 1 - Enable, 2 - Not Installed

**Version** **verClass**

**System** root class version

**Version** **verDisplayDriver**

**Version** of the NV display driver.

**Version** **vernViewDesktopManager**

The version of nView Desktop Manager.

**Version** **verNVWMI**

**Version** of the NVWMI driver.

**Version** **verSDICaptureFirmware**

Firmware version of the SDI Capture board.

**Version** **verSDIOOutputFirmware**

Firmware version of the SDI output board.

**Version** **verVBIOS**

The video BIOS version associated with the GPU's.
The documentation for this class was generated from the following file:

- nvwmi.mof

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
ThermalEvent Class Reference

Represents GPU thermal events. More...

List of all members.

Public Attributes

- uint32 handleGpu
- uint32 probeIndex
- Thermal probe index in array of probes on a given GPU.
- uint32 thermalLevel
- Thermal level to identify the temperature zone. Possible values are: 0 - unknown, 1 - normal, 2 - warning, 3 - critical.

Detailed Description
Represents GPU thermal events.

## Member Data Documentation

**uint32 handleGpu**

Unique identification of the NVIDIA GPU

**uint32 probeIndex**

Thermal probe index in array of probes on a given GPU.

**uint32 thermalLevel**

Thermal level to identify the temperature zone. Possible values are: 0 - unknown, 1 - normal, 2 - warning, 3 - critical.

The documentation for this class was generated from the following file:

- nvwmi.mof
ThermalProbe Class Reference

Represents a thermal probe. More...

List of all members.

Public Member Functions

string info ()
    Formats basic information about the Thermal object in a system into a human-readable string.

Public Attributes

Version verClass
    uint32 id
        Unique ID.
uint32 handle
Unique ID - deprecated, please use 'id'.

uint32 type
The type of thermal probe controller type. Possible values are:
-1 - unknown, 0 - none, 1 - gpUInternal, 2 - adm1032, 3 -
max6649, 4 - max1617, 5 - lm99, 6 - lm89, 7 - lm64, 8 - adt7473,
9 - sbmax6649, 10 - vbiosEvt, 11 - os.

sint32 temperature
The current temperature value of the thermal sensor in degrees
Celsius.

sint32 defaultMinTemperature
Default minimum temperature value of the thermal sensor in
degrees Celsius.

sint32 defaultMaxTemperature
Default maximum temperature value of the thermal sensor in
degrees Celsius.

uint32 thermalLevel
Current temperature event level. Possible values are: 0 -
unknown, 1 - normal, 2 - warning, 3 - critical

---

**Detailed Description**

Represents a thermal probe.

---

**Member Function Documentation**

string info()

Formats basic information about the Thermal object in a system into a
human-readable string.

---

**Member Data Documentation**
sint32 **defaultMaxTemperature**

Default maximum temperature value of the thermal sensor in degrees Celsius.

sint32 **defaultMinTemperature**

Default minimum temperature value of the thermal sensor in degrees Celsius.

uint32 **handle**

Unique ID - deprecated, please use 'id'.

uint32 **id**

Unique ID.

sint32 **temperature**

The current temperature value of the thermal sensor in degrees Celsius.

uint32 **thermalLevel**

current temperature event level. Possible values are: 0 - unknown, 1 - normal, 2 - warning, 3 - critical

uint32 **type**

The type of thermal probe controller type. Possible values are: -1 - unknown, 0 - none, 1 - gpuInternal, 2 - adm1032, 3 - max6649, 4 - max1617, 5 - lm99, 6 - lm89, 7 - lm64, 8 - adt7473, 9 - sbmax6649, 10 - vbiosEvt, 11 - os.

**Version verClass**

Thermal probe class version
The documentation for this class was generated from the following file:

- nvwmi.mof

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Version Class Reference

Internal version of NV objects. More...

List of all members.

Public Attributes

uint32 orderedValue
uint32 value
    Raw version value.
string strValue
    Human-readable string with version value.

Detailed Description

Internal version of NV objects.
Member Data Documentation

uint32 orderedValue

Ordered version value, guaranteed to grow monotonically

string strValue

Human-readable string with version value.

uint32 value

Raw version value.

The documentation for this class was generated from the following file:

- nvwmi.mof

---

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
VideoCodec Class Reference

Represents parameters of Video Engine encoder and decoder. More...

List of all members.

Public Member Functions

string info ()

Formats basic information about this PCI-Express link into a human-readable string.

Public Attributes

Version verClass
sint32 percentEncoderUsage

The percentage of Video Encoder utilization during sampling
period.
sint32 **percentDecoderUsage**
The percentage of Video Decoder utilization during sampling period.

sint32 **encoderSamplingPeriod**
The period (in milliseconds) between Video Encoder samples. -1 if not supported in current environment.

sint32 **decoderSamplingPeriod**
The period (in milliseconds) between Video Decoder samples. -1 if not supported in current environment.

sint32 **encoderSessionsCount**
Count of active encoder sessions. 0 if not supported in current environment or when encoder inactive.

sint32 **averageFps**
Frames per second encoded, as a trailing average for all active sessions. -1 if not supported in current environment.

sint32 **averageLatency**
Encoding latency in milliseconds. -1 if not supported in current environment.

---

**Detailed Description**

Represents parameters of Video Engine encoder and decoder.

---

**Member Function Documentation**

string info()

Formats basic information about this PCI-Express link into a human-readable string.

---

**Member Data Documentation**
sint32 **averageFps**

Frames per second encoded, as a trailing average for all active sessions. -1 if not supported in current environment.

sint32 **averageLatency**

Encoding latency in milliseconds. -1 if not supported in current environment.

sint32 **decoderSamplingPeriod**

The period (in milliseconds) between Video Decoder samples. -1 if not supported in current environment.

sint32 **encoderSamplingPeriod**

The period (in milliseconds) between Video Encoder samples. -1 if not supported in current environment.

sint32 **encoderSessionsCount**

Count of active encoder sessions. 0 if not supported in current environment or when encoder inactive.

sint32 **percentDecoderUsage**

The percentage of Video Decoder utilization during sampling period.

sint32 **percentEncoderUsage**

The percentage of Video Encoder utilization during sampling period.

**Version verClass**

Object version
The documentation for this class was generated from the following file:

- nvwmi.mof

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Here is a list of all class members with links to the classes they belong to:

- a -

- addApplications() : ApplicationProfile
- aggregateDoubleBitErrors : Ecc
- aggregateSingleBitErrors : Ecc
- applications : ApplicationProfile
- applyDisplayProfiles() : ProfileManager
- applySystemProfile() : ProfileManager
- archId : Gpu
- archName : Gpu
- averageEncodeFps : EncoderSessions
- averageEncodeLatency : EncoderSessions
- averageFps : VideoCodec
- averageLatency : VideoCodec

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Related Pages

Here is a list of all related documentation pages:

- Legal Notice
- Version-specific Implementation Details
- NVWMI compatibility
- Cooler and Thermal Events
- Using NVWMI
- NVIDIA Performance Counters
- Logging and tracing NVWMI activity
- Using NVWMI with the WMIC tool
- Using NVWMI with the PowerShell
- Profile settings available in NVWMI
- Third-Party Notice

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Here are the classes, structs, unions and interfaces with brief descriptions:

- **Application**: Application, associated with an application profile
- **ApplicationProfile**: Application profile
- **Board**: Represents a board with NVIDIA GPU(s)
- **Cooler**: Represents a cooler (fan, liquid system etc.)
- **CoolerEvent**: Represents GPU cooler events
- **DesktopManager**: Management of nView desktops
- **Display**: Represents physical display
- **DisplayGrid**: Represents physical displays, organized into a regular grid (rows by columns)
- **DisplayGridInfo**: Information about a display grid
- **DisplayManager**: Management of display-related task
- **DisplayMode**: Represents display mode
- **DisplayProfile**: Display profile
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ecc</td>
<td>Represents the GPU Error Correction Code. This feature is not supported if there are multiple GPU topologies enabled.</td>
</tr>
<tr>
<td>EncoderSessions</td>
<td>Represents encoder sessions info</td>
</tr>
<tr>
<td>Gpu</td>
<td>Represents an NVIDIA GPU</td>
</tr>
<tr>
<td>NamedObject</td>
<td>Base class for all named NV objects</td>
</tr>
<tr>
<td>OverlapLimits</td>
<td>Represents overlap limits of individual display in a display grid</td>
</tr>
<tr>
<td>PcieLink</td>
<td>Represents parameters of PCI Express bus link</td>
</tr>
<tr>
<td>ProcessUtilization</td>
<td>Represents per process GPU encoder utilization values</td>
</tr>
<tr>
<td>Profile</td>
<td>Global profile class</td>
</tr>
<tr>
<td>ProfileManager</td>
<td>Represents root object for performing Profile related task</td>
</tr>
<tr>
<td>Setting</td>
<td>Profile setting</td>
</tr>
<tr>
<td>SettingInfo</td>
<td>Useful information about a profile setting</td>
</tr>
<tr>
<td>SettingTable</td>
<td>Table which describes profile settings of certain type</td>
</tr>
<tr>
<td>Sync</td>
<td>Represents Sync-capable devices</td>
</tr>
<tr>
<td>SyncDelay</td>
<td>Represents the Sync Delay (Sync Skew or Startup Delay)</td>
</tr>
<tr>
<td>SyncEvent</td>
<td>Represent Sync events</td>
</tr>
<tr>
<td>SyncTopology</td>
<td>Represents the Sync Topology</td>
</tr>
<tr>
<td>System</td>
<td>Represents root object for NV-related HW and SW in the system</td>
</tr>
<tr>
<td>ThermalEvent</td>
<td>Represents GPU thermal events</td>
</tr>
<tr>
<td>ThermalProbe</td>
<td>Represents a thermal probe</td>
</tr>
<tr>
<td>Version</td>
<td>Internal version of NV objects</td>
</tr>
<tr>
<td>VideoCodec</td>
<td>Represents parameters of Video Engine encoder and decoder</td>
</tr>
</tbody>
</table>
Class Index

A | B | C | D | E | G | N | O | P | S | T | V

A
Application
ApplicationProfile
B
Board
C
Cooler
CoolerEvent
D
DesktopManager
Display
DisplayGrid
DisplayGridInfo
DisplayManager
DisplayMode
DisplayProfile
E
Ecc
EncoderSessions
G
Gpu
N
NamedObject
O
OverlapLimits
P
PcieLink
ProcessUtilization
Profile
ProfileManager
S
Setting
SettingInfo
SettingTab
Sync
SyncDelay

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Application Member List

This is the complete list of members for Application, including all inherited members.

<table>
<thead>
<tr>
<th>id</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>info()</td>
<td>Application</td>
</tr>
<tr>
<td>isPredefined</td>
<td>Application</td>
</tr>
<tr>
<td>isSupported</td>
<td>Application</td>
</tr>
<tr>
<td>launcher</td>
<td>Application</td>
</tr>
<tr>
<td>name</td>
<td>Application</td>
</tr>
<tr>
<td>subPaths</td>
<td>Application</td>
</tr>
<tr>
<td>verClass</td>
<td>Application</td>
</tr>
</tbody>
</table>
ApplicationProfile Member List

This is the complete list of members for ApplicationProfile, including all inherited members.

- `addApplications([in]string appNames[])`  
- `applications`  
- `id`  
- `info()`  
- `isPredefined`  
- `isSupported`  
- `name`  
- `removeApplications([in]string appNames[])`  
- `restoreSettings([in]uint32 settingIds[])`  
- `setBinaryValueById([in]uint32 settingId,[in]uint8 value[])`  
- `setStringValueById([in]uint32 settingId,[in]string value)`
settings

```
setValueById([in]uint32 settingId, [in]uint32 value)
```

startTime
type
verClass
Board Member List

This is the complete list of members for `Board`, including all inherited members.

- `chipSKU` Board
- `chipSKUMod` Board
- `coolers` Board
- `count` Board
- `gpus` Board
- `id` Board
- `info()` Board
- `name` Board
- `nvapild` Board
- `ordinal` Board
- `project` Board
- `projectSKU` Board
- `serialNumber` Board
<table>
<thead>
<tr>
<th>thermalProbes Board</th>
<th>Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>uname Board</td>
<td>Board</td>
</tr>
<tr>
<td>ver Board</td>
<td></td>
</tr>
</tbody>
</table>

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
### Cooler Member List

This is the complete list of members for **Cooler**, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coolerLevel</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>coolerType</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>fanSpeed</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>handle</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>info()</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>maxSpeed</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>minSpeed</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>percentCoolerRate</td>
<td>Cooler</td>
<td></td>
</tr>
<tr>
<td>verClass</td>
<td>Cooler</td>
<td></td>
</tr>
</tbody>
</table>
CoolerEvent Member List

This is the complete list of members for **CoolerEvent**, including all inherited members.

- `coolerIndex` CoolerEvent
- `coolerLevel` CoolerEvent
- `handleGpu` CoolerEvent

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
DesktopManager Member List

This is the complete list of members for DesktopManager, including all inherited members.

createDesktop([in]string name,[in]string backgrounds) | DesktopManager
deleteDesktop([in]string name) | DesktopManager
editDesktop([in]string name,[in]string backgrounds) | DesktopManager
getAllDesktops() | DesktopManager
verClass | DesktopManager
Display Member List

This is the complete list of members for Display, including all inherited members.

- `count`
- `displayConnectorType`
- `displayModeNative`
- `displayModes`
- `ditherBits`
- `ditherMode`
- `ditherState`
- `EDID`
- `EDIDSize`
- `getCurrentTiming()`
- `gpuConnectorType`
- `id`
- `info()`
isActive
locus
make
model
name
nvapild
ordinal
restoreNativeDisplayMode()
rotation
saveCSC([in]string filePath)
saveEDID([in]string filePath)
saveGammaRamp([in]string filePath)
scaling
setCSC([in]string filePath)
setDisplayMode([in]uint32 width,[in]uint32 height,[in]real32 refreshRate,[
setDisplayModeById([in]uint32 id)
setDisplayModeByRef([in]DisplayMode ref mode)
setDither([in, ValueMap{"0","1","2"}, Values{"Default","Enable","Disable"}] bit","10 bit"}) uint32 bits,[in, ValueMap{"0","1","2","3","4"}, Values{"SpatialDynamic","SpatialStatic","SpatialDynamic2x2","SpatialStat
setEDID([in]string filePath)
setGammaRamp([in]string filePath)
setGammaRampBasic([in]real32 brightness,[in]real32 contrast,[in]real32
setRotation([in, ValueMap{"0","1","2","3"}, Values{"No rotation","rotate 90 uint32 rotation})
setScaling([in, ValueMap{"0","1","2","3","5","6","7"}, Values{"Default","Closest","Native","ScanoutToNative","AspectScanoutToNative"} uint32 scaling)
uname
er
ver
verFirmware
This is the complete list of members for `DisplayGrid`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cols</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>count</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>displayModePhysical</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>displayModes</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>displayModeVirtual</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>displays</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>gpus</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>id</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>info()</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>name</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>ordinal</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>overlapCols</td>
<td>DisplayGrid</td>
</tr>
<tr>
<td>overlapLimits</td>
<td>DisplayGrid</td>
</tr>
</tbody>
</table>
overlapRows
positionCol
positionRow
rotation
rows
saveCSC([in]string filePath)
saveGammaRamp([in]string filePath)
setCSC([in]string filePath)
setDisplayMode([in]uint32 width,[in]uint32 height,
[in]real32 refreshRate,[in]uint32 depth)
setDisplayModeById([in]uint32 id)
setDisplayModeByRef([in]DisplayMode ref gridMode)
setGammaRamp([in]string filePath)
setGammaRampBasic([in]real32 brightness,[in]real32 contrast,[in]real32 gamma)
setOverlapCol([in]sint32 index,[in]sint32 overlap)
setOverlapRow([in]sint32 index,[in]sint32 overlap)
setOverlaps([in]sint32 index,[in]sint32 overlapRow,
[in]sint32 overlapCol)
setRotation([in]uint32 rotation[])
uname
ver

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
DisplayGridInfo Member List

This is the complete list of members for DisplayGridInfo, including all inherited members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cols</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>cscFilePath</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>displayIds</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>gammaRampFilePath</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>id</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>info()</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>primaryId</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>rows</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>unames</td>
<td>DisplayGridInfo</td>
</tr>
<tr>
<td>verClass</td>
<td>DisplayGridInfo</td>
</tr>
</tbody>
</table>
DisplayManager Member List

This is the complete list of members for DisplayManager, including all inherited members.

createClone([in, ValueMap{"0","1"}, Values{"basic","smart"}] uint32 type,[in]
createCustomTiming([in]string timing)
createDisplayGridById([in]uint32 cols,[in]uint32 rows,[in]uint32 displayId
createDisplayGridByName([in]uint32 cols,[in]uint32 rows,[in]string displayName
createDisplayGridByRef([in]uint32 cols,[in]uint32 rows,[in]string displayRef
createDisplayGrids([in]string grids[])
createModeFilter([in]uint32 displays[],[in]string filter)
deleteCustomTiming([in]string timing)
deleteModeFilter([in]uint32 displays[])
editCustomTiming([in]uint32 modelId,[in]string newTiming)
enumCustomTimings([in]uint32 displays[])
enumModeFilters([in]uint32 displays[])
fakeEDIDAll([in]string filePath,[in, ValueMap{"-1","0","1","2","3","4","5","6"...]
Values{"unknown","uninitialized","VGA","Component","S-Video","HDMI","["DVI","LVDS","DP","Composite","All"]

loadCustomTimings([in]uint32 displays[],[in]string filePath)

saveCustomTimings([in]uint32 displays[],[in]string filePath)

setGridPositions([in]uint32 positionCol[],[in]uint32 positionRow[])

setScalingAll([in, ValueMap{"0","1","2","3","5","6","7"}, Values{"Default","Closest","Native","ScanoutToNative","AspectScanoutToNative","AspectRatioScanoutToClosest","ScanoutToClosest"}]

uint32 scaling)

tryCustomTiming([in]string timing)

validateDisplayGridById([in]uint32 cols,[in]uint32 rows,[in]uint32 displayId)

validateDisplayGrids([in]string grids[])

verClass
DisplayMode Member List

This is the complete list of members for `DisplayMode`, including all inherited members.

- `colorDepth` `DisplayMode`
- `height` `DisplayMode`
- `id` `DisplayMode`
- `info()` `DisplayMode`
- `refreshRate` `DisplayMode`
- `verClass` `DisplayMode`
- `width` `DisplayMode`
DisplayProfile Member List

This is the complete list of members for DisplayProfile, including all inherited members.

grid DisplayProfile
id DisplayProfile
info() DisplayProfile
isPredefined DisplayProfile
isSupported DisplayProfile
name DisplayProfile
restoreSettings([in]uint32 settingIds[]) DisplayProfile
setBinaryValueById([in]uint32 settingId,[in]uint8 value[]) DisplayProfile
setStringValueById([in]uint32 settingId,[in]string value) DisplayProfile
settings DisplayProfile
setValueById([in]uint32 settingId,[in]uint32 value) DisplayProfile
type DisplayProfile
Ecc Member List

This is the complete list of members for Ecc, including all inherited members.

aggregateDoubleBitErrors Ecc
aggregateSingleBitErrors Ecc
currentDoubleBitErrors Ecc
currentSingleBitErrors Ecc
id Ecc
info() Ecc
isEnabled Ecc
isEnabledByDefault Ecc
isSupported Ecc
isWritable Ecc
option Ecc
resetCounters([in]boolean bResetCurrent,[in]boolean bResetAggregate) Ecc
setConfiguration([in]boolean bEnable,[in]boolean bEnableImmediately)

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
EncoderSessions Member List

This is the complete list of members for EncoderSessions, including all inherited members.

- averageEncodeFps
- averageEncodeLatency
- codecType
- hResolution
- id
- info()
- processId
- sessionId
- sessionsCount
- verClass
- vgpuInstance
- vResolution
This is the complete list of members for Gpu, including all inherited members.

- archId
- archName
- coolers
- coreCount
- count
- createUtilizationEvent([in]sint32 lower,[in]sint32 upper,[in, ValueMap{"-1","0","1","2","3"}, Values{"All","GPU","Frame Buffer","Video Engine","PCIe bus"}] sint32 domain)
- deleteUtilizationEvents([in, ValueMap{"-1","0","1","2","3"}, Values{"All","GPU","Frame Buffer","Video Engine","PCIe bus"}] sint32 domain)
- deviceInfo
- disableLicensedFeature([in]string feature)
ecc
enableLicensedFeature([in]string feature)
encoderSessionsInfo
fakeEDID([in]string filePath,[in,
ValueMap{"-1","0","1","2","3","4","5","6","7","8","9"},
Values{"unknown","uninitialized","VGA","Component","S-
Video","HDMI","DVI","LVDS","DP","Composite","All"}] uint32 output)
fakeEDIDOnPort([in]string filePath,[in]uint32 portIndex,[in,
ValueMap{"-1","0","1","2","3","4","5","6","7","8"},
Values{"unknown","uninitialized","VGA","Component","S-
Video","HDMI","DVI","LVDS","DP","Composite"}] uint32 output)
getAllUtilizationEvents([in, ValueMap{"-1","0","1","2","3"},
Values{"All","GPU","Frame Buffer","Video Engine","PCIe bus"}] sint32 domain)
gpuCoreClockCurrent
handle
id
info()
licensableFeatures
licensableStatus
memoryBusWidth
memoryClockCurrent
memorySizeAvailable
memorySizePhysical
memorySizeVirtual
memoryType
name
nvapild
ordinal
pcieDownstreamWidth
pcieGpu
percentGpuMemoryUsage
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentGpuUsage</td>
<td>Gpu</td>
</tr>
<tr>
<td>power</td>
<td>Gpu</td>
</tr>
<tr>
<td>powerSampleCount</td>
<td>Gpu</td>
</tr>
<tr>
<td>powerSamplingPeriod</td>
<td>Gpu</td>
</tr>
<tr>
<td>processUtilization</td>
<td>Gpu</td>
</tr>
<tr>
<td>productName</td>
<td>Gpu</td>
</tr>
<tr>
<td>productType</td>
<td>Gpu</td>
</tr>
<tr>
<td>thermalProbes</td>
<td>Gpu</td>
</tr>
<tr>
<td>uname</td>
<td>Gpu</td>
</tr>
<tr>
<td>ver</td>
<td>Gpu</td>
</tr>
<tr>
<td>verVBIOS</td>
<td>Gpu</td>
</tr>
<tr>
<td>videoCodec</td>
<td>Gpu</td>
</tr>
</tbody>
</table>
NamedObject Member List

This is the complete list of members for NamedObject, including all inherited members.

- count NamedObject
- id NamedObject
- name NamedObject
- ordinal NamedObject
- uname NamedObject
- ver NamedObject

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
OverlapLimits Member List

This is the complete list of members for **OverlapLimits**, including all inherited members.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>info</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>maxOverlapCol</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>maxOverlapRow</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>maxTotalCol</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>maxTotalRow</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>minOverlapCol</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>minOverlapRow</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>minTotalCol</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>minTotalRow</td>
<td>OverlapLimits</td>
</tr>
<tr>
<td>verClass</td>
<td>OverlapLimits</td>
</tr>
</tbody>
</table>
This is the complete list of members for `PcieLink`, including all inherited members.

- `curGen` `PcieLink`
- `curSpeed` `PcieLink`
- `curWidth` `PcieLink`
- `info()` `PcieLink`
- `maxGen` `PcieLink`
- `maxSpeed` `PcieLink`
- `maxWidth` `PcieLink`
- `verClass` `PcieLink`
ProcessUtilization Member List

This is the complete list of members for ProcessUtilization, including all inherited members.

- decUtilization
- encUtilization
- id
- info()
- memUtilization
- pid
- processCount
- smUtilization
- timeStamp
- verClass
Profile Member List

This is the complete list of members for **Profile**, including all inherited members.

<table>
<thead>
<tr>
<th></th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Profile</td>
</tr>
<tr>
<td>info()</td>
<td>Profile</td>
</tr>
<tr>
<td>isPredefined</td>
<td>Profile</td>
</tr>
<tr>
<td>isSupported</td>
<td>Profile</td>
</tr>
<tr>
<td>name</td>
<td>Profile</td>
</tr>
<tr>
<td>restoreSettings([in]uint32 settingIds[])</td>
<td>Profile</td>
</tr>
<tr>
<td>setBinaryValueById([in]uint32 settingId,[in]uint8 value[])</td>
<td>Profile</td>
</tr>
<tr>
<td>setStringValueById([in]uint32 settingId,[in]string value)</td>
<td>Profile</td>
</tr>
<tr>
<td>settings</td>
<td>Profile</td>
</tr>
<tr>
<td>setValueById([in]uint32 settingId,[in]uint32 value)</td>
<td>Profile</td>
</tr>
<tr>
<td>type</td>
<td>Profile</td>
</tr>
<tr>
<td>verClass</td>
<td>Profile</td>
</tr>
</tbody>
</table>
ProfileManager Member List

This is the complete list of members for ProfileManager, including all inherited members.

- applyDisplayProfiles([in]string prefix) ProfileManager
- applySystemProfile([in]string name) ProfileManager
- createProfile([in]string name,[in]uint32 type,[in]string params) ProfileManager
- currentProfile3D ProfileManager
- currentProfileDesktop ProfileManager
- currentSystemProfile ProfileManager
- defaultProfile3D ProfileManager
- defaultProfileDesktop ProfileManager
- defaultSystemProfile ProfileManager
- deleteDesktopProfile([in]string name) ProfileManager
- getAllProfiles([in, ValueMap{"0","1","3","4","5"}, Values{"3D Application Profile","3D Global ProfileManager"}]} ProfileManager
Profile","nView Profile","Display Profile","System Profile"] uint32 type)

info() ProfileManager
loadDesktopProfile([in]string name) ProfileManager
lockDesktopProfile([in]string name,[in]uint32 lock) ProfileManager
restoreDefaults3D() ProfileManager
restoreProfile([in]string name) ProfileManager
saveDesktopProfile([in]string name) ProfileManager
saveDisplayProfiles([in]string prefix) ProfileManager
saveSystemProfile([in]string name) ProfileManager
setCurrentProfile3D([in]string name) ProfileManager
setVSync([in, ValueMap{"0","1","2","3","4"}, Values{"Passive","Off","On","Adaptive","Adaptive half"}] ProfileManager
uint32 vsyncMode)
verClass ProfileManager

---
NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Setting Member List

This is the complete list of members for Setting, including all inherited members.

getStringValue() Setting
id Setting
info() Setting
type Setting
value Setting
verClass Setting
SettingInfo Member List

This is the complete list of members for SettingInfo, including all inherited members.

- description SettingInfo
- id SettingInfo
- info() SettingInfo
- name SettingInfo
- verClass SettingInfo
SettingTable Member List

This is the complete list of members for `SettingTable`, including all inherited members.

- `getIdFromName([in]string name)`
- `getNameFromId([in]uint32 id)`
- `id`
- `info()`
- `infoById([in]uint32 id)`
- `infoByName([in]string name)`
- `settingIds` (in `string name`)
- `settingNames`
- `settings`
- `type`
- `verClass`
Sync Member List

This is the complete list of members for Sync, including all inherited members.

count
flStatus
id
info()
interlaceMode
interval
isHouseSync
isStereoSynced
isSynced
name
nvapild
ordinal
polarity
setInterlaceMode([in, ValueMap{"0","1"}, Values{"Disabled","Enabled"}] uint32 interlaceMode)

setInterval([in]uint32 interval)

setPolarity([in, ValueMap{"0","1","2"}, Values{"Rising Edge","Falling Edge","Both Edges"}] uint32 polarity)

setStartupDelay([in]uint32 numOfPixels,[in]uint32 numOfLines)

setSyncSkew([in]uint32 numOfPixels,[in]uint32 numOfLines)

setSyncStateById([in]uint32 syncDisplayIds[],[in]uint32 syncState[])

setSyncStateByName([in]string syncDisplayNames,[in]uint32 syncState[])

setVmode([in, ValueMap{"0","1","2","3","4"}, Values{"none","TTL","NTSC-PAL-SECAM","HDTV","composite"}] uint32 vmode)

source

startupDelay

syncDisplays

syncSignalRate

syncSkew

toggleSource()

uname

ver

verSyncFirmware

vmode

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
SyncDelay Member List

This is the complete list of members for `SyncDelay`, including all inherited members.

<table>
<thead>
<tr>
<th>Member</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>delayType</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>id</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>info()</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>maxLines</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>minPixels</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>numOfLines</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>numOfPixels</td>
<td><code>SyncDelay</code></td>
</tr>
<tr>
<td>verClass</td>
<td><code>SyncDelay</code></td>
</tr>
</tbody>
</table>
SyncEvent Member List

This is the complete list of members for `SyncEvent`, including all inherited members.

- `eventType`  `SyncEvent`
- `syncDeviceId`  `SyncEvent`
- `verClass`  `SyncEvent`
SyncTopology Member List

This is the complete list of members for `SyncTopology`, including all inherited members.

```csharp
count SyncTopology
displaySyncState SyncTopology
id SyncTopology
info() SyncTopology
isDisplayMasterable SyncTopology
isGPUSynced() SyncTopology
name SyncTopology
ordinal SyncTopology
uname SyncTopology
ver SyncTopology
```
System Member List

This is the complete list of members for System, including all inherited members.

- disableEncoderSessionsPerfCounter()
- disableProcessUtilizationPerfCounter()
- enableEncoderSessionsPerfCounter()
- enableProcessUtilizationPerfCounter()
- info()
- licensingPort
- licensingServer
- logFilter
- logOptions
- logTypes
- machineType
- nViewState
- setLicensingServer([in]string address,[in]uint16 port)
**setLogState**([in, BitMap{"0","1","2"}],
BitValues{"debugger","file","eventLog"}] uint32 types,[in]uint32 filter,[in]uint32 options)

**setnViewState**([in, ValueMap{"0","1"},
Values{"Disable","Enable"}] uint32 state)

**unsetLicensingServer**()

**verClass**

**verDisplayDriver**

**vernViewDesktopManager**

**verNVWMI**

**verSDICaptureFirmware**

**verSDIOutputFirmware**

**verVBIOS**

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
ThermalEvent Member List

This is the complete list of members for ThermalEvent, including all inherited members.

handleGpu  ThermalEvent
probeIndex  ThermalEvent
thermalLevel ThermalEvent
This is the complete list of members for **ThermalProbe**, including all inherited members.

```cpp
defaultMaxTemperature ThermalProbe
defaultMinTemperature ThermalProbe
handle ThermalProbe
id ThermalProbe
info() ThermalProbe
temperature ThermalProbe
thermalLevel ThermalProbe
type ThermalProbe
verClass ThermalProbe
```
Version Member List

This is the complete list of members for Version, including all inherited members.

orderedValue Version
strValue Version
value Version

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
VideoCodec Member List

This is the complete list of members for VideoCodec, including all inherited members.

- `averageFps` VideoCodec
- `averageLatency` VideoCodec
- `decoderSamplingPeriod` VideoCodec
- `encoderSamplingPeriod` VideoCodec
- `encoderSessionsCount` VideoCodec
- `info()` VideoCodec
- `percentDecoderUsage` VideoCodec
- `percentEncoderUsage` VideoCodec
- `verClass` VideoCodec
- a -
- addApplications() : ApplicationProfile
- applyDisplayProfiles() : ProfileManager
- applySystemProfile() : ProfileManager

- c -

- createClone() : DisplayManager
- createCustomTiming() : DisplayManager
- createDesktop() : DesktopManager
- createDisplayGridById() : DisplayManager
- createDisplayGridByName() : DisplayManager
- createDisplayGridByRef() : DisplayManager
- createDisplayGrids() : DisplayManager
- createModeFilter() : DisplayManager
- createProfile() : ProfileManager
- createUtilizationEvent() : Gpu

- d -

- deleteCustomTiming() : DisplayManager
- deleteDesktop() : DesktopManager
- deleteDesktopProfile() : ProfileManager
- deleteModeFilter() : DisplayManager
- deleteUtilizationEvents() : Gpu
- disableEncoderSessionsPerfCounter() : System
- disableLicensedFeature() : Gpu
- disableProcessUtilizationPerfCounter() : System

- e -

- editCustomTiming() : DisplayManager
- editDesktop() : DesktopManager
- enableEncoderSessionsPerfCounter() : System
- enableLicensedFeature() : Gpu
- enableProcessUtilizationPerfCounter() : System
- enumCustomTimings() : DisplayManager
- enumModeFilters() : DisplayManager
- f -
  - fakeEDID() : Gpu
  - fakeEDIDAII() : DisplayManager
  - fakeEDIDOOnPort() : Gpu

- g -
  - getAllDesktops() : DesktopManager
  - getAllProfiles() : ProfileManager
  - getAllUtilizationEvents() : Gpu
  - getCurrentTiming() : Display
  - getIdFromName() : SettingTable
  - getNameFromId() : SettingTable
  - getStringValue() : Setting

- i -
  - info() : PcieLink, VideoCodec, System, DisplayGrid, SettingTable, SettingInfo, DisplayProfile, ApplicationProfile, Profile, DisplayGridInfo, Application, Setting, ProfileManager, Sync, EncoderSessions, SyncDelay, SyncTopology, ProcessUtilization, Ecc, ThermalProbe, Cooler, DisplayMode, Display, Board, OverlapLimits, DisplayManager, Gpu
  - infoById() : SettingTable
  - infoByName() : SettingTable
  - isGPUSynced() : SyncTopology

- l -
  - loadCustomTimings() : DisplayManager
  - loadDesktopProfile() : ProfileManager
  - lockDesktopProfile() : ProfileManager

- r -
  - removeApplications() : ApplicationProfile
• resetCounters() : Ecc
• restoreDefaults3D() : ProfileManager
• restoreNativeDisplayMode() : Display
• restoreProfile() : ProfileManager
• restoreSettings() : DisplayProfile, ApplicationProfile, Profile

- S -

• saveCSC() : DisplayGrid, Display
• saveCustomTimings() : DisplayManager
• saveDesktopProfile() : ProfileManager
• saveDisplayProfiles() : ProfileManager
• saveEDID() : Display
• saveGammaRamp() : Display, DisplayGrid
• saveSystemProfile() : ProfileManager
• setBinaryValueById() : Profile, ApplicationProfile, DisplayProfile
• setConfiguration() : Ecc
• setCSC() : DisplayGrid, Display
• setCurrentProfile3D() : ProfileManager
• setDisplayMode() : DisplayGrid, Display
• setDisplayModeById() : DisplayGrid, Display
• setDisplayModeByRef() : DisplayGrid, Display
• setDither() : Display
• setEDID() : Display
• setGammaRamp() : DisplayGrid, Display
• setGammaRampBasic() : DisplayGrid, Display
• setGridPositions() : DisplayManager
• setInterlaceMode() : Sync
• setInterval() : Sync
• setLicensingServer() : System
• setnViewState() : System
• setOverlapCol() : DisplayGrid
• setOverlapRow() : DisplayGrid
• setOverlaps() : DisplayGrid
• setPolarity() : Sync
• setRotation() : DisplayGrid, Display
- t -

- u -

- v -

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- a -

- aggregateDoubleBitErrors : Ecc
- aggregateSingleBitErrors : Ecc
- applications : ApplicationProfile
- archId : Gpu
- archName : Gpu
- averageEncodeFps : EncoderSessions
- averageEncodeLatency : EncoderSessions
- averageFps : VideoCodec
- averageLatency : VideoCodec

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
NVWMI v2.31 API Reference Documentation
NVIDIA

- Main Page
- Related Pages
- Classes

- Class List
- Class Index
- Class Members

- All
- Functions
- Variables

- a
- c
- d
- e
- f
- g
- h
- i
- j
- l
- m
- n
- o
- p
- r
- s
- t
- u
Here is a list of all class members with links to the classes they belong to:

- C -

- chipSKU : Board
- chipSKUMod : Board
- codecType : EncoderSessions
- colorDepth : DisplayMode
- cols : DisplayGrid, DisplayGridInfo
- coolerIndex : CoolerEvent
- coolerLevel : Cooler, CoolerEvent
- coolers : Gpu, Board
- coolerType : Cooler
- coreCount : Gpu
- count : Board, DisplayGrid, Display, SyncTopology, Sync, NamedObject, Gpu
- createClone() : DisplayManager
- createCustomTiming() : DisplayManager
- createDesktop() : DesktopManager
- createDisplayGridById() : DisplayManager
- createDisplayGridByName() : DisplayManager
- createDisplayGridByRef() : DisplayManager
- createDisplayGrids() : DisplayManager
- createModeFilter() : DisplayManager
- createProfile() : ProfileManager
- createUtilizationEvent() : Gpu
- cscFilePath : DisplayGridInfo
- curGen : PcieLink
- currentDoubleBitErrors : Ecc
- currentProfile3D : ProfileManager
- currentProfileDesktop : ProfileManager
- currentSingleBitErrors : Ecc
- currentSystemProfile : ProfileManager
- curSpeed : PcieLink
Main Page
Related Pages
Classes

Class List
Class Index
Class Members

All
Functions
Variables

a
c
d
e
f
g
h
i
j
l
m
n
o
p
q
r
s
t
u
Here is a list of all class members with links to the classes they belong to:

- d -

- decoderSamplingPeriod : VideoCodec
- decUtilization : ProcessUtilization
- defaultMaxTemperature : ThermalProbe
- defaultMinTemperature : ThermalProbe
- defaultProfile3D : ProfileManager
- defaultProfileDesktop : ProfileManager
- defaultSystemProfile : ProfileManager
- delayType : SyncDelay
- deleteCustomTiming() : DisplayManager
- deleteDesktop() : DesktopManager
- deleteDesktopProfile() : ProfileManager
- deleteModeFilter() : DisplayManager
- deleteUtilizationEvents() : Gpu
- description : SettingInfo
- deviceInfo : Gpu
- disableEncoderSessionsPerfCounter() : System
- disableLicensedFeature() : Gpu
- disableProcessUtilizationPerfCounter() : System
- displayConnectorType : Display
- displayIds : DisplayGridInfo
- displayModeNative : Display
- displayModePhysical : DisplayGrid
- displayModes : DisplayGrid, Display
- displayModeVirtual : DisplayGrid
- displays : DisplayGrid
- displaySyncState : SyncTopology
- ditherBits : Display
- ditherMode : Display
- ditherState : Display
Here is a list of all class members with links to the classes they belong to:

- ecc : **Gpu**
- EDID : **Display**
- EDIDSize : **Display**
- editCustomTiming() : **DisplayManager**
- editDesktop() : **DesktopManager**
- enableEncoderSessionsPerfCounter() : **System**
- enableLicensedFeature() : **Gpu**
- enableProcessUtilizationPerfCounter() : **System**
- encoderSamplingPeriod : **VideoCodec**
- encoderSessionsCount : **VideoCodec**
- encoderSessionsInfo : **Gpu**
- encUtilization : **ProcessUtilization**
- enumCustomTimings() : **DisplayManager**
- enumModeFilters() : **DisplayManager**
- eventType : **SyncEvent**

---

**NVIDIA**

*Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.*
- Main Page
- Related Pages
- Classes

- Class List
- Class Index
- Class Members

- All
- Functions
- Variables

- a
- c
- d
- e
- f
- g
- h
- i
- j
- l
- m
- n
- o
- p
- r
- s
- t
- u
Here is a list of all class members with links to the classes they belong to:

- f -

  - fakeEDID() : **Gpu**
  - fakeEDIDAll() : **DisplayManager**
  - fakeEDIDOnPort() : **Gpu**
  - fanSpeed : **Cooler**
  - flStatus : **Sync**
• Main Page
• Related Pages
• Classes

• Class List
• Class Index
• Class Members

• All
• Functions
• Variables

• a
• c
• d
• e
• f
• g
• h
• i
• j
• l
• m
• n
• o
• p
• r
• s
• t
• u
Here is a list of all class members with links to the classes they belong to:

- g -

- gammaRampFilePath : DisplayGridInfo
- getAllDesktops() : DesktopManager
- getAllProfiles() : ProfileManager
- getAllUtilizationEvents() : Gpu
- getCurrentTiming() : Display
- getIdFromName() : SettingTable
- getNameFromId() : SettingTable
- getStringValue() : Setting
- gpuConnectorType : Display
- gpuCoreClockCurrent : Gpu
- gpus : Board, DisplayGrid
- grid : DisplayProfile

———

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Here is a list of all class members with links to the classes they belong to:

- handle : **Gpu**, **Cooler**, **ThermalProbe**
- handleGpu : **CoolerEvent**, **ThermalEvent**
- height : **DisplayMode**
- hResolution : **EncoderSessions**
Here is a list of all class members with links to the classes they belong to:

- i -

- id: NamedObject, Gpu, OverlapLimits, Cooler, Setting, Application, ThermalProbe, DisplayGridInfo, Profile, DisplayGrid, Ecc, ApplicationProfile, DisplayProfile, ProcessUtilization, SettingInfo, SettingTable, Board, Display, SyncTopology, EncoderSessions, DisplayMode, SyncDelay, Sync
- info(): SettingTable, Display, DisplayMode, ApplicationProfile, Cooler, ThermalProbe, Ecc, Setting, ProcessUtilization, SyncTopology, EncoderSessions, SyncDelay, Sync, ProfileManager, Application, DisplayGridInfo, Profile, PcieLink, DisplayProfile, DisplayGrid, SettingInfo, VideoCodec, Gpu, System, Board, DisplayManager, OverlapLimits
- infoById(): SettingTable
- infoByName(): SettingTable
- interlaceMode: Sync
- interval: Sync
- isActive: Display
- isDisplayMasterable: SyncTopology
- isEnabled: Ecc
- isEnabledByDefault: Ecc
- isGPUSynced(): SyncTopology
- isHouseSync: Sync
- isPredefined: Profile, DisplayProfile, ApplicationProfile, Application
- isStereoSynced: Sync
- isSupported: ApplicationProfile, DisplayProfile, Profile, Application, Ecc
- isSynced: Sync
- isWritable: Ecc
Here is a list of all class members with links to the classes they belong to:

- | -

- launcher : **Application**
- licensableFeatures : **Gpu**
- licensableStatus : **Gpu**
- licensingPort : **System**
- licensingServer : **System**
- loadCustomTimings() : **DisplayManager**
- loadDesktopProfile() : **ProfileManager**
- lockDesktopProfile() : **ProfileManager**
- locus : **Display**
- logFilter : **System**
- logOptions : **System**
- logTypes : **System**
Here is a list of all class members with links to the classes they belong to:

- m -

- machineType : System
- make : Display
- maxGen : PcieLink
- maxLines : SyncDelay
- maxOverlapCol : OverlapLimits
- maxOverlapRow : OverlapLimits
- maxSpeed : Cooler, PcieLink
- maxTotalCol : OverlapLimits
- maxTotalRow : OverlapLimits
- maxWidth : PcieLink
- memoryBusWidth : Gpu
- memoryClockCurrent : Gpu
- memorySizeAvailable : Gpu
- memorySizePhysical : Gpu
- memorySizeVirtual : Gpu
- memoryType : Gpu
- memUtilization : ProcessUtilization
- minOverlapCol : OverlapLimits
- minOverlapRow : OverlapLimits
- minPixels : SyncDelay
- minSpeed : Cooler
- minTotalCol : OverlapLimits
- minTotalRow : OverlapLimits
- model : Display
Here is a list of all class members with links to the classes they belong to:

- n -

- name : NamedObject, Gpu, DisplayGrid, Application, Profile, Display, ApplicationProfile, DisplayProfile, SettingInfo, Board, SyncTopology, Sync
- numOfLines : SyncDelay
- numOfPixels : SyncDelay
- nvapild : Gpu, Board, Display, Sync
- nViewState : System

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
• Main Page
• Related Pages
• Classes

• Class List
• Class Index
• Class Members

• All
• Functions
• Variables

• a
• c
• d
• e
• f
• g
• h
• i
• j
• k
• m
• n
• o
• p
• q
• r
• s
• t
• u
Here is a list of all class members with links to the classes they belong to:

- Option: Ecc
- orderedValue: Version
- ordinal: Gpu, SyncTopology, Sync, Board, Display, DisplayGrid, NamedObject
- overlapCols: DisplayGrid
- overlapLimits: DisplayGrid
- overlapRows: DisplayGrid
Here is a list of all class members with links to the classes they belong to:

- p -

- pcieDownstreamWidth : Gpu
- pcieGpu : Gpu
- percentCoolerRate : Cooler
- percentDecoderUsage : VideoCodec
- percentEncoderUsage : VideoCodec
- percentGpuMemoryUsage : Gpu
- percentGpuUsage : Gpu
- pid : ProcessUtilization
- polarity : Sync
- positionCol : DisplayGrid
- positionRow : DisplayGrid
- power : Gpu
- powerSampleCount : Gpu
- powerSamplingPeriod : Gpu
- primaryId : DisplayGridInfo
- probeIndex : ThermalEvent
- processCount : ProcessUtilization
- processId : EncoderSessions
- processUtilization : Gpu
- productName : Gpu
- productType : Gpu
- project : Board
- projectSKU : Board

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Here is a list of all class members with links to the classes they belong to:

- refreshRate : DisplayMode
- removeApplications() : ApplicationProfile
- resetCounters() : Ecc
- restoreDefaults3D() : ProfileManager
- restoreNativeDisplayMode() : Display
- restoreProfile() : ProfileManager
- restoreSettings() : ApplicationProfile, DisplayProfile, Profile
- rotation : Display, DisplayGrid
- rows : DisplayGrid, DisplayGridInfo

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
• Main Page
• Related Pages
• Classes

• Class List
• Class Index
• Class Members

• All
• Functions
• Variables

• a
• c
• d
• e
• f
• g
• h
• i
• j
• l
• m
• n
• o
• p
• r
• s
• t
• u
Here is a list of all class members with links to the classes they belong to:

- S -

• saveCSC() : DisplayGrid, Display
• saveCustomTimings() : DisplayManager
• saveDesktopProfile() : ProfileManager
• saveDisplayProfiles() : ProfileManager
• saveEDID() : Display
• saveGammaRamp() : Display, DisplayGrid
• saveSystemProfile() : ProfileManager
• scaling : Display
• serialNumber : Board
• sessionId : EncoderSessions
• sessionsCount : EncoderSessions
• setBinaryValueById() : DisplayProfile, Profile, ApplicationProfile
• setConfiguration() : Ecc
• setCSC() : DisplayGrid, Display
• setCurrentProfile3D() : ProfileManager
• setDisplayMode() : DisplayGrid, Display
• setDisplayModeById() : DisplayGrid, Display
• setDisplayModeByRef() : DisplayGrid, Display
• setDither() : Display
• setEDID() : Display
• setGammaRamp() : DisplayGrid, Display
• setGammaRampBasic() : Display, DisplayGrid
• setGridPositions() : DisplayManager
• setInterlaceMode() : Sync
• setInterval() : Sync
• setLicensingServer() : System
• setLogState() : System
• setnViewState() : System
• setOverlapCol() : DisplayGrid
• setOverlapRow() : DisplayGrid
- `setOverlaps()` : `DisplayGrid`
- `setPolarity()` : `Sync`
- `setRotation()` : `DisplayGrid`, `Display`
- `setScaling()` : `Display`
- `setScalingAll()` : `DisplayManager`
- `setStartupDelay()` : `Sync`
- `setStringValueById()` : `Profile`, `ApplicationProfile`, `DisplayProfile`
- `setSyncSkew()` : `Sync`
- `setSyncStateById()` : `Sync`
- `setSyncStateByName()` : `Sync`
- `settingIds` : `SettingTable`
- `settingNames` : `SettingTable`
- `settings` : `SettingTable`, `ApplicationProfile`, `Profile`, `DisplayProfile`
- `setValueById()` : `ApplicationProfile`, `DisplayProfile`, `Profile`
- `setVmode()` : `Sync`
- `setVSync()` : `ProfileManager`
- `smUtilization` : `ProcessUtilization`
- `source` : `Sync`
- `startTime` : `ApplicationProfile`
- `startupDelay` : `Sync`
- `strValue` : `Version`
- `subPaths` : `Application`
- `syncDeviceId` : `SyncEvent`
- `syncDisplays` : `Sync`
- `syncSignalRate` : `Sync`
- `syncSkew` : `Sync`
Here is a list of all class members with links to the classes they belong to:

- `t`

- `v`
- `w`

- temperature : ThermalProbe
- thermalLevel : ThermalProbe, ThermalEvent
- thermalProbes : Gpu, Board
- timeStamp : ProcessUtilization
- toggleSource() : Sync
- tryCustomTiming() : DisplayManager
- type : Setting, DisplayProfile, ApplicationProfile, Profile, ThermalProbe, SettingTable

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- Main Page
- Related Pages
- Classes

- Class List
- Class Index
- Class Members

- All
- Functions
- Variables

- a
- c
- d
- e
- f
- g
- h
- i
- j
- l
- m
- n
- o
- p
- r
- s
- t
- u
Here is a list of all class members with links to the classes they belong to:

- **uname**: NamedObject, Gpu, DisplayGrid, Sync, SyncTopology, Display, Board
- **unames**: DisplayGridInfo
- **unsetLicensingServer()**: System

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Here is a list of all class members with links to the classes they belong to:

- V -

- validateDisplayGridById() : DisplayManager
- validateDisplayGrids() : DisplayManager
- value : Setting, Version
- ver : DisplayGrid, Display, NamedObject, SyncTopology, Sync, Gpu, Board
- verClass : DisplayManager, SettingInfo, SettingTable, OverlapLimits, SyncEvent, DisplayMode, DisplayProfile, ApplicationProfile, Profile, Cooler, Application, Setting, ThermalProbe, Ecc, ProcessUtilization, EncoderSessions, PcieLink, SyncDelay, ProfileManager, VideoCodec, System, DisplayGridInfo, DesktopManager
- verDisplayDriver : System
- verFirmware : Display
- vernViewDesktopManager : System
- verNVWMI : System
- verSDICaptureFirmware : System
- verSDIOutputFirmware : System
- verSyncFirmware : Sync
- verVBIOS : Gpu, System
- vgpuInstance : EncoderSessions
- videoCodec : Gpu
- vmode : Sync
- vResolution : EncoderSessions

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Here is a list of all class members with links to the classes they belong to:

- **width** : `DisplayMode`
- C -

- v
- w

- chipSKU : Board
- chipSKUMod : Board
- codecType : EncoderSessions
- colorDepth : DisplayMode
- cols : DisplayGrid, DisplayGridInfo
- coolerIndex : CoolerEvent
- coolerLevel : Cooler, CoolerEvent
- coolers : Gpu, Board
- coolerType : Cooler
- coreCount : Gpu
- count : Board, DisplayGrid, Gpu, Display, Sync, SyncTopology, NamedObject
- cscFilePath : DisplayGridInfo
- curGen : PcieLink
- currentDoubleBitErrors : Ecc
- currentProfile3D : ProfileManager
- currentProfileDesktop : ProfileManager
- currentSingleBitErrors : Ecc
- currentSystemProfile : ProfileManager
- curSpeed : PcieLink
- curWidth : PcieLink

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- d -

- decoderSamplingPeriod : VideoCodec
- decUtilization : ProcessUtilization
- defaultMaxTemperature : ThermalProbe
- defaultMinTemperature : ThermalProbe
- defaultProfile3D : ProfileManager
- defaultProfileDesktop : ProfileManager
- defaultSystemProfile : ProfileManager
- delayType : SyncDelay
- description : SettingInfo
- deviceInfo : Gpu
- displayConnectorType : Display
- displayIds : DisplayGridInfo
- displayModeNative : Display
- displayModePhysical : DisplayGrid
- displayModes : Display, DisplayGrid
- displayModeVirtual : DisplayGrid
- displays : DisplayGrid
- displaySyncState : SyncTopology
- ditherBits : Display
- ditherMode : Display
- ditherState : Display
NVWMI v2.31 API Reference Documentation
NVIDIA

Feb
2018

- Main Page
- Related Pages
- Classes

- Class List
- Class Index
- Class Members

- All
- Functions
- Variables

- a
- c
- d
- e
- f
- g
- h
- i
- j
- l
- m
- n
- o
- p
- r
- s
- t
- u
- e -

- ecc : **Gpu**
- EDID : **Display**
- EDIDSize : **Display**
- encoderSamplingPeriod : **VideoCodec**
- encoderSessionsCount : **VideoCodec**
- encoderSessionsInfo : **Gpu**
- encUtilization : **ProcessUtilization**
- eventType : **SyncEvent**
- f -

- fanSpeed : **Cooler**
- flStatus : **Sync**
- g -

- gammaRampFilePath : DisplayGridInfo
- gpuConnectorType : Display
- gpuCoreClockCurrent : Gpu
- gpus : Board, DisplayGrid
- grid : DisplayProfile

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
• v
• w

- h -

• handle : Gpu, Cooler, ThermalProbe
• handleGpu : CoolerEvent, ThermalEvent
• height : DisplayMode
• hResolution : EncoderSessions
Main Page
Related Pages
Classes

Class List
Class Index
Class Members

All
Functions
Variables

a
c
d
e
f
g
h
j
l
m
n
o
p
r
s
t
u
- i -

- v
- w

- i -

- id : NamedObject, Gpu, OverlapLimits, Cooler, Setting, Application, ThermalProbe, DisplayGridInfo, Profile, DisplayGrid, Ecc, ApplicationProfile, DisplayProfile, ProcessUtilization, SettingInfo, SettingTable, Board, Display, SyncTopology, EncoderSessions, DisplayMode, SyncDelay, Sync
- interlaceMode : Sync
- interval : Sync
- isActive : Display
- isDisplayMasterable : SyncTopology
- isEnabled : Ecc
- isEnabledByDefault : Ecc
- isHouseSync : Sync
- isPredefined : Profile, ApplicationProfile, DisplayProfile, Application
- isStereoSynced : Sync
- isSupported : Application, Ecc, DisplayProfile, ApplicationProfile, Profile
- isSynced : Sync
- isWritable : Ecc

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- | -

- launcher: **Application**
- licensableFeatures: **Gpu**
- licensableStatus: **Gpu**
- licensingPort: **System**
- licensingServer: **System**
- locus: **Display**
- logFilter: **System**
- logOptions: **System**
- logTypes: **System**

---

**NVIDIA**

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- m -

- v
- w

- m -

- machineType : System
  - make : Display
  - maxGen : PcieLink
  - maxLines : SyncDelay
  - maxOverlapCol : OverlapLimits
  - maxOverlapRow : OverlapLimits
  - maxSpeed : Cooler, PcieLink
  - maxTotalCol : OverlapLimits
  - maxTotalRow : OverlapLimits
  - maxWidth : PcieLink
  - memoryBusWidth : Gpu
  - memoryClockCurrent : Gpu
  - memorySizeAvailable : Gpu
  - memorySizePhysical : Gpu
  - memorySizeVirtual : Gpu
  - memoryType : Gpu
  - memUtilization : ProcessUtilization
  - minOverlapCol : OverlapLimits
  - minOverlapRow : OverlapLimits
  - minPixels : SyncDelay
  - minSpeed : Cooler
  - minTotalCol : OverlapLimits
  - minTotalRow : OverlapLimits
  - model : Display

---

NVIDIA

Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Main Page
Related Pages
Classes

Class List
Class Index
Class Members

All
Functions
Variables

a
c
d
e
f
g
h
j
l
m
n
o
p
r
s
t
u
- n -

- name : **NamedObject**, **Gpu**, **DisplayGrid**, **Application**, **Profile**, **Display**, **ApplicationProfile**, **DisplayProfile**, **SettingInfo**, **Board**, **SyncTopology**, **Sync**
- numOfLines : **SyncDelay**
- numOfPixels : **SyncDelay**
- nvapild : **Gpu**, **Board**, **Display**, **Sync**
- nViewState : **System**
- O -

- option : Ecc
- orderedValue : Version
- ordinal : Gpu, SyncTopology, Sync, Board, Display, DisplayGrid, NamedObject
- overlapCols : DisplayGrid
- overlapLimits : DisplayGrid
- overlapRows : DisplayGrid
- Main Page
- Related Pages
- Classes

- Class List
- Class Index
- Class Members

- All
- Functions
- Variables

- a
- c
- d
- e
- f
- g
- h
- i
- j
- l
- m
- n
- o
- p
- r
- s
- t
- u
- p -

- v
- w

- p -

- v
- w
- r -

- refreshRate : DisplayMode
- rotation : DisplayGrid, Display
- rows : DisplayGrid, DisplayGridInfo

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- S -

- scaling: Display
- serialNumber: Board
- sessionId: EncoderSessions
- sessionsCount: EncoderSessions
- settingIds: SettingTable
- settingNames: SettingTable
- settings: ApplicationProfile, DisplayProfile, SettingTable, Profile
- smUtilization: ProcessUtilization
- source: Sync
- startTime: ApplicationProfile
- startupDelay: Sync
- strValue: Version
- subPaths: Application
- syncDeviceId: SyncEvent
- syncDisplays: Sync
- syncSignalRate: Sync
- syncSkew: Sync

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- t -

- temperature : ThermalProbe
- thermalLevel : ThermalProbe, ThermalEvent
- thermalProbes : Gpu, Board
- timeStamp : ProcessUtilization
- type : ThermalProbe, DisplayProfile, ApplicationProfile, SettingTable, Setting, Profile

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
- U -

- uname : NamedObject, Gpu, Sync, SyncTopology, Display, DisplayGrid, Board
- unames : DisplayGridInfo

NVIDIA
Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.
Main Page
Related Pages
Classes

Class List
Class Index
Class Members

All
Functions
Variables

a
c
d
e
f
g
h
i
j
l
m
n
o
p
r
s
t
u
• value: Version, Setting
• ver: Gpu, SyncTopology, Sync, Board, NamedObject, DisplayGrid, Display
• verClass: DisplayMode, SyncEvent, Cooler, ThermalProbe, Ecc, PcieLink, SettingTable, SettingInfo, DisplayProfile, ProcessUtilization, Profile, DisplayGridInfo, EncoderSessions, VideoCodec, SyncDelay, ProfileManager, System, Setting, Application, DesktopManager, DisplayManager, ApplicationProfile, OverlapLimits
• verDisplayDriver: System
• verFirmware: Display
• vernViewDesktopManager: System
• verNVWMI: System
• verSDICaptureFirmware: System
• verSDIOutputFirmware: System
• verSyncFirmware: Sync
• verVBIOS: Gpu, System
• vgpuInstance: EncoderSessions
• videoCodec: Gpu
• vmode: Sync
• vResolution: EncoderSessions
- \mathbf{w} -

- width : \textbf{DisplayMode}

\textbf{NVIDIA}

\textit{Copyright (c) 2010-2018 NVIDIA Corporation. All rights reserved.}