

NAME

nvidia-settings – configure the NVIDIA graphics driver

SYNOPSIS

nvidia-settings [*options*]
nvidia-settings [*options*] **--no-config**
nvidia-settings [*options*] **--load-config-only**
nvidia-settings [*options*] {**--query=attr** | **--assign=attr=value**} ...
nvidia-settings [*options*] **--glxinfo**

Options: [**-vh**] [**--config=configfile**] [**-c ctrl-display**]
 [**--verbose={errors | warnings | all}**]
 [**--describe={all | list | attribute_name}**]

attr has the form:

DISPLAY/attribute_name[display_devices]

DESCRIPTION

The **nvidia-settings** utility is a tool for configuring the NVIDIA graphics driver. It operates by communicating with the NVIDIA X driver, querying and updating state as appropriate. This communication is done with the NV-CONTROL X extension.

Values such as brightness and gamma, XVideo attributes, temperature, and OpenGL settings can be queried and configured via **nvidia-settings**.

When **nvidia-settings** starts, it reads the current settings from its configuration file and sends those settings to the X server. Then, it displays a graphical user interface (GUI) for configuring the current settings. When **nvidia-settings** exits, it queries the current settings from the X server and saves them to the configuration file.

OPTIONS

-v, --version

Print the **nvidia-settings** version and exit.

-h, --help

Print usage information and exit.

--config=config

Use the configuration file *config* rather than the default *~/.nvidia-settings-rc*

-c, --ctrl-display=ctrl-display

Control the specified X display. If this option is not given, then **nvidia-settings** will control the display specified by **--display**. If that is not given, then the *\$DISPLAY* environment variable is used.

-n, --no-config

Do not load the configuration file. This mode of operation is useful if **nvidia-settings** has difficulties starting due to problems with applying settings in the configuration file.

-l, --load-config-only

Load the configuration file, send the values specified therein to the X server, and exit. This mode of operation is useful to place in your *.xinitrc* file, for example.

-r, --rewrite-config-file

Write the current X server configuration to the configuration file, and exit without starting a graphical user interface. See Examples section.

-V, --verbose=verbosity

Controls how much information is printed. By default, the verbosity is **errors** and only error messages are printed.

verbosity can be one of the following values:

errors - Print errors.

warnings - Print errors and warnings.

all - Print errors, warnings, and other information.

-a, --assign=assign

The *assign* argument to the **--assign** commandline option is of the form:

```
{DISPLAY}/{attribute name}[[{display devices}]=value]
```

This assigns the attribute {attribute name} to the value {value} on the X Display {DISPLAY}. {DISPLAY} follows the usual {host}:{display}.{screen} syntax of the DISPLAY environment variable and is optional; when it is not specified, then it is implied following the same rule as the **--ctrl-display** option. If the X screen is not specified, then the assignment is made to all X screens. Note that the '/' is only required when {DISPLAY} is present.

{DISPLAY} can additionally include a target specification to direct an assignment to something other than an X screen. A target specification is contained within brackets and consists of a target type name, a colon, and the target id. The target type name can be one of **screen**, **gpu**, or **frame-lock**; the target id is the index into the list of targets (for that target type). The target specification can be used in {DISPLAY} wherever an X screen can be used, following the syntax {host}:{display}[[{target_type}:{target_id}]]. See the output of

```
nvidia-settings --query all
```

for information on which target types can be used with which attributes. See the output of

```
nvidia-settings -q screens -q gpus -q framelocks
```

for lists of targets for each target type.

The [[{display devices}]] portion is also optional; if it is not specified, then the attribute is assigned to all display devices.

Some examples:

```
-a FSAA=5
-a localhost:0.0/DigitalVibrance[CRT-0]=0
--assign="SyncToVBlank=1"
-a [gpu:0]/DigitalVibrance[DFP-1]=63
```

-q, --query=query

The *query* argument to the **--query** commandline option is of the form:

```
{DISPLAY}/{attribute name}[[{display devices}]]
```

This queries the current value of the attribute {attribute name} on the X Display {DISPLAY}. The syntax is the same as that for the **--assign** option, without **=value**. Specify **-q screens**, **-q gpus**, or **-q framelocks** to query a list of X screens, GPUs, or Frame Lock devices, respectively, that are present on the X Display {DISPLAY}. Specify **-q all** to query all attributes.

-g, --glxinfo

Print GLX Information for the X display and exit.

-e, --describe

Prints information about a particular attribute. Specify 'all' to list the descriptions of all attributes. Specify 'list' to list the attribute names without a descriptions.

USER GUIDE

Contents

1. Layout of the nvidia-settings GUI
2. How OpenGL Interacts with nvidia-settings
3. Loading Settings Automatically
4. Commandline Interface
5. X Display Names in the Config File
6. Connecting to Remote X Servers
7. Licensing
8. TODO

1. Layout of the nvidia-settings GUI

The **nvidia-settings** GUI is organized with a list of different categories on the left side. Only one entry in the list can be selected at once, and the selected category controls which "page" is displayed on the right side of the **nvidia-settings** GUI.

The category list is organized in a tree: each X screen contains the relevant subcategories beneath it. Similarly, the Display Devices category for a screen contains all the enabled display devices beneath it. Besides each X screen, the other top level category is "nvidia-settings Configuration", which configures behavior of the **nvidia-settings** application itself.

Along the bottom of the **nvidia-settings** GUI, from left to right, is:

- 1) a status bar which indicates the most recently altered option;
- 2) a Help button that toggles the display of a help window which provides a detailed explanation of the available options in the current page; and
- 3) a Quit button to exit **nvidia-settings**.

Most options throughout **nvidia-settings** are applied immediately. Notable exceptions are OpenGL options which are only read by OpenGL when an OpenGL application starts.

Details about the options on each page of **nvidia-settings** are available in the help window.

2. How OpenGL Interacts with nvidia-settings

When an OpenGL application starts, it downloads the current values from the X driver, and then reads the environment (see *APPENDIX E: OPENGL ENVIRONMENT VARIABLE SETTINGS* in the README). Settings from the X server override OpenGL's default values, and settings from the environment override values from the X server.

For example, by default OpenGL uses the FSAA setting requested by the application (normally, applications do not request any FSAA). An FSAA setting specified in **nvidia-settings** would override the OpenGL application's request. Similarly, the `__GL_FSAA_MODE` environment variable will override the application's FSAA setting, as well as any FSAA setting specified in **nvidia-settings**.

Note that an OpenGL application only retrieves settings from the X server when it starts, so if you make a change to an OpenGL value in **nvidia-settings**, it will only apply to OpenGL applications which are started after that point in time.

3. Loading Settings Automatically

The NVIDIA X driver does not preserve values set with **nvidia-settings** between runs of the X server (or even between logging in and logging out of X, with **xdm(1)**, **gdm**, or **kdm**). This is intentional, because different users may have different preferences, thus these settings are stored on a per-user basis in a configuration file stored in the user's home directory.

The configuration file is named `~/.nvidia-settings-rc`. You can specify a different configuration file name with the `--config` commandline option.

After you have run **nvidia-settings** once and have generated a configuration file, you can then run:

```
nvidia-settings --load-config-only
```

at any time in the future to upload these settings to the X server again. For example, you might place the above command in your `~/.xinitrc` file so that your settings are applied automatically when you log in to X.

Your `.xinitrc` file, which controls what X applications should be started when you log into X (or startx), might look something like this:

```
nvidia-settings --load-config-only &
xterm &
evilwm
```

or:

```
nvidia-settings --load-config-only &
gnome-session
```

If you do not already have an `~/.xinitrc` file, then chances are that **xinit(1)** is using a system-wide `xinitrc` file. This system wide file is typically here:

```
/etc/X11/xinit/xinitrc
```

To use it, but also have **nvidia-settings** upload your settings, you could create an `~/.xinitrc` with the contents:

```
nvidia-settings --load-config-only &
./etc/X11/xinit/xinitrc
```

System administrators may choose to place the **nvidia-settings** load command directly in the system `xinitrc` script.

Please see the **xinit(1)** man page for further details of configuring your `~/.xinitrc` file.

4. Commandline Interface

nvidia-settings has a rich commandline interface: all attributes that can be manipulated with the GUI can also be queried and set from the command line. The commandline syntax for querying and assigning attributes matches that of the `.nvidia-settings-rc` configuration file.

The `--query` option can be used to query the current value of attributes. This will also report the valid values for the attribute. You can run **nvidia-settings --query all** for a complete list of available attributes, what the current value is, what values are valid for the attribute, and through which target types (e.g., X screens, GPUs) the attributes can be addressed. Additionally, individual attributes may be specified like this:

```
nvidia-settings --query CursorShadow
```

Attributes that may differ per display device (for example, DigitalVibrance can be set independently on each display device when in TwinView) can be appended with a "display device name" within brackets; e.g.:

```
nvidia-settings --query DigitalVibrance[CRT-0]
```

If an attribute is display device specific, but the query does not specify a display device, then the attribute value for all display devices will be queried.

An attribute name may be prepended with an X Display name and a forward slash to indicate a different X Display; e.g.:

```
nvidia-settings --query localhost:0.0/DigitalVibrance[DFP-1]
```

An attribute name may also just be prepended with the screen number and a forward slash:

```
nvidia-settings --query 0/DigitalVibrance[DFP-1]
```

in which case the default X Display will be used, but you can indicate to which X screen to direct the query (if your X server has multiple X screens). If no X screen is specified, then the attribute value will be queried for all X screens.

Attributes can be addressed through "target types". A target type indicates the object that is queried when you query an attribute. The default target type is an X screen, but other possible target types are GPUs and Frame Lock devices.

Target types give you different granularities with which to perform queries and assignments. Since X screens can span multiple GPUs (in the case of Xinerama, or SLI), and multiple X screens can exist on the same GPU, it is sometimes useful to address attributes by GPU rather than X screen.

A target specification is contained within brackets and consists of a target type name, a colon, and the target id. The target type name can be one of **screen**, **gpu**, or **framelock**; the target id is the index into the list of targets (for that target type). Target specifications can be used wherever an X screen is used in query and assignment commands; the target specification can be used either by itself on the left side of the forward slash, or as part of an X Display name.

For example, the following queries address X screen 0 on the localhost:

```
nvidia-settings --query 0/VideoRam
nvidia-settings --query localhost:0.0/VideoRam
nvidia-settings --query [screen:0]/VideoRam
nvidia-settings --query localhost:0[screen:0]/VideoRam
```

To address GPU 0 instead, you can use either of:

```
nvidia-settings --query [gpu:0]/VideoRam
nvidia-settings --query localhost:0[gpu:0]/VideoRam
```

See the output of

```
nvidia-settings --query all
```

for what target types can be used with each attribute. See the output of

```
nvidia-settings --query screens --query gpus --query framelocks
```

for lists of targets for each target type.

The **--assign** option can be used to assign a new value to an attribute. The valid values for an attribute are reported when the attribute is queried. The syntax for **--assign** is the same as **--query**, with the additional requirement that assignments also have an equal sign and the new value. For example:

```
nvidia-settings --assign FSAA=2
nvidia-settings --assign 0/DigitalVibrance[CRT-1]=9
nvidia-settings --assign [gpu:0]/DigitalVibrance=0
```

Multiple queries and assignments may be specified on the commandline for a single invocation of **nvidia-settings**.

If either the **--query** or **--assign** options are passed to **nvidia-settings**, the GUI will not be presented, and **nvidia-settings** will exit after processing the assignments and/or queries.

5. X Display Names in the Config File

In the Commandline Interface section above, it was noted that you can specify an attribute without any X Display qualifiers, with only an X screen qualifier, or with a full X Display name. For example:

```
nvidia-settings --query FSAA
nvidia-settings --query 0/FSAA
nvidia-settings --query stravinsky.nvidia.com:0/FSAA
```

In the first two cases, the default X Display will be used, in the second case, the screen from the default X Display can be overridden, and in the third case, the entire default X Display can be overridden.

The same possibilities are available in the `~/.nvidia-settings-rc` configuration file.

For example, in a computer lab environment, you might log into any of multiple workstations, and your home directory is NFS mounted to each workstation. In such a situation, you might want your `~/.nvidia-settings-rc` file to be applicable to all the workstations. Therefore, you would not want your config file to qualify each attribute with an X Display Name. Leave the "Include X Display Names in the Config File" option unchecked on the **nvidia-settings** Configuration page (this is the default).

There may be cases when you do want attributes in the config file to be qualified with the X Display name. If you know what you are doing and want config file attributes to be qualified with an X Display, check the "Include X Display Names in the Config File" option on the **nvidia-settings** Configuration page.

In the typical home user environment where your home directory is local to one computer and you are only configuring one X Display, then it does not matter whether each attribute setting is qualified with an X Display Name.

6. Connecting to Remote X Servers

nvidia-settings is an X client, but uses two separate X connections: one to display the GUI, and another to communicate the NV-CONTROL requests. These two X connections do not need to be to the same X server. For example, you might run **nvidia-settings** on the computer `stravinsky.nvidia.com`, export the display to the computer `bartok.nvidia.com`, but be configuring the X server on the computer `schoenberg.nvidia.com`:

```
nvidia-settings --display=bartok.nvidia.com:0 \
  --ctrl-display=schoenberg.nvidia.com:0
```

If `--ctrl-display` is not specified, then the X Display to control is what `--display` indicates. If `--display` is also not specified, then the `$DISPLAY` environment variable is used.

Note, however, that you will need to have X permissions configured such that you can establish an X connection from the computer on which you are running **nvidia-settings** (`stravinsky.nvidia.com`) to the computer where you are displaying the GUI (`bartok.nvidia.com`) and the computer whose X Display you are configuring (`schoenberg.nvidia.com`).

The simplest, most common, and least secure mechanism to do this is to use 'xhost' to allow access from the computer on which you are running **nvidia-settings**.

```
(issued from bartok.nvidia.com)
xhost +stravinsky.nvidia.com
```

```
(issued from schoenberg.nvidia.com)
xhost +stravinsky.nvidia.com
```

This will allow all X clients run on `stravinsky.nvidia.com` to connect and display on `bartok.nvidia.com`'s X server and configure `schoenberg.nvidia.com`'s X server.

Please see the **xauth**(1) and **xhost**(1) man pages, or refer to your system documentation on remote X applications and security. You might also Google for terms such as "remote X security" or "remote X

Windows", and see documents such as the Remote X Apps mini-HOWTO:

`<http://www.tldp.org/HOWTO/Remote-X-Apps.html>`

Please also note that the remote X server to be controlled must be using the NVIDIA X driver.

7. Licensing

The source code to **nvidia-settings** is released as GPL. The most recent official version of the source code is available here:

`<ftp://download.nvidia.com/XFree86/nvidia-settings/>`

Note that **nvidia-settings** is simply an NV-CONTROL client. It uses the NV-CONTROL X extension to communicate with the NVIDIA X server to query current settings and make changes to settings.

You can make additions directly to **nvidia-settings**, or write your own NV-CONTROL client, using **nvidia-settings** as an example.

Documentation on the NV-CONTROL extension and additional sample clients are available in the **nvidia-settings** source tarball. Patches can be submitted to `linux-bugs@nvidia.com`.

8. TODO

There are many things still to be added to **nvidia-settings**, some of which include:

- different toolkits? The GUI for **nvidia-settings** is cleanly abstracted from the backend of **nvidia-settings** that parses the configuration file and commandline, communicates with the X server, etc. If someone were so inclined, a different frontend GUI could be implemented.
- write a design document explaining how **nvidia-settings** is architected; presumably this would make it easier for people to become familiar with the code base.

If there are other things you would like to see added (or better yet, would like to add yourself), please contact `linux-bugs@nvidia.com`.

FILES

`~/nvidia-settings-rc`

EXAMPLES

nvidia-settings

Starts the **nvidia-settings** graphical interface.

nvidia-settings --load-config-only

Loads the settings stored in `~/nvidia-settings-rc` and exits.

nvidia-settings --rewrite-config-file

Writes the current X server configuration to `~/nvidia-settings-rc` file and exits.

nvidia-settings --query FSAA

Query the value of the full-screen antialiasing setting.

nvidia-settings --assign RedGamma=2.0 --assign BlueGamma=2.0 --assign GreenGamma=2.0

Set the gamma of the screen to 2.0.

AUTHOR

Aaron Plattner
NVIDIA Corporation

SEE ALSO

nvidia-xconfig(1), **nvidia-installer(1)**

COPYRIGHT

Copyright © 2006 NVIDIA Corporation.