GRID SDK FAQ

PG-06183-001| January 2014

Frequently Asked Questions

DOCUMENT CHANGE HISTORY

PG-06183-001

Version	Date	Authors	Description of Change
0.5	5/13/2013	BO	Initial draft
0.8	12/5/2013	EY	Revised and added more FAQ
0.9	12/5/2013	NS/DG	Added 6 new FAQ items
1.0	1/17/2014	EY	First public release

Q1) What is the GRID SDK? How do developers get access to it?

A1) The GRID SDK enables fast capture and compression of the desktop display or render targets from NVIDIA GRID cloud gaming graphics boards. You will need to register in order to download the GRID SDK. Registration is free, and the download page can be found here: https://developer.nvidia.com/grid-app-game-streaming.

Q2) I download the SDK. Now how do I get started?

A2) For recommended hardware, please refer to Question #23 of this FAQ. Please refer to the "Guide to Setup a GRID Server" and the GRID SDK Programming Guide for how to begin using the SDK once you have downloaded the SDK and driver. These samples are simple examples for how to use the GRID APIs on Linux or Windows:

```
{Installation Directory}\Samples
```

Q3) What components are included with GRID SDK? What are they used for?

A3) The GRID SDK consists of two component software APIs: NvFBC and NvIFR.

a) **NvFBC** captures (and optionally H.264 encodes) the entire visible desktop
b) **NvIFR** captures (and optionally H.264 encodes) from a specific render target

Note: Both of these GRID components will interface with the NVENC API for H.264 hardware compression. The NVENC interface is not provided with the GRID SDK, but NvFBC and NvIFR include functions to configure the H.264 hardware encoder.

Q4) What are the interfaces for NvFBC Capture?

A4) There is a NvFBC interface for each type of capture you want to do and where you want the data to go:

a) `NVFBC_CAPTURE_TO_SYS` captures the desktop and copies it to pinned system memory on the host
b) `NVFBC_CAPTURE_SHARED_CUDA` captures the desktop and copies it to CUDA device memory on the GPU
c) `NVFBC_CAPTURE_TO_H264_HW_ENCODER` captures the desktop, compresses it using on-chip hardware video encoder to a H.264 video stream and copies the H.264 encoded elementary bitstream to system memory on the host.

Q5) What are the interfaces for NvIFR Capture?

A5) There is an NvIFR interface for each operation that you want to do when capturing

a) **NvIFRToSys** captures the render target and copies it to pinned system memory on the host
b) **NvIFRH264** captures the render target, compresses it using on-chip hardware video encoder to a H.264 video stream and copies the H.264 encoded elementary bitstream to system memory on the host.

Q6) When should NvFBC and NvIFR be used?

A6) NvIFR is the preferred solution to capture the video output of one specific application. NvFBC is better for remote desktop applications. For a more detailed explanation of the differences, please refer to the "Amazon G2 Instance Getting Started Guide" included with the SDK.

Q7) Can NvIFR capture GDI/2D components of an application window. If not, will it be support in the future?

A7) No, NvIFR is the API for capturing from RenderTargets. If you would like to capture GDI and 2D Components from the system desktop, use NvFBC. There are currently no plans to support this in the future.

Q8) What hardware and drivers are required for the GRID SDK?

A8) Please refer to the release notes included in the GRID SDK package you intend to use for supported hardware and operating systems.

Q9) What Operating Systems does the GRID SDK support?

A9) Microsoft Windows 7, Windows 8, Windows Server 2008 R2, Windows Server 2012, and Linux OS. For servers running Windows, we recommend Windows 8 and Windows Server 2012 because these operating systems are more efficient and can serve more concurrent users.

Q10) Can I distribute any GRID SDK components to my customers?

A10) No, the GRID SDK is subject to NVIDIA's Software License Agreement, which is included with the SDK. Any re-distribution requires express agreement from NVIDIA.

Q11) Whom can I contact on questions integrating my software with the GRID SDK?

A11) Please refer to the Programming Guide and included documentation. If there are other more detailed questions, please contact GRID developer support at NVIDIA GridPublicSupport@nvidia.com .

Q12) What are the differences between the GRID cloud gaming boards?

A12) GRID K520 and K340 are for use to stream consumer applications games. GRID K520, which includes two Kepler-based GPUs and 8GB of memory, is designed to be a high performance GPU with a powerful 3D engine, but less simultaneous encoding performance as GRID K340. NVIDIA recommends using the GRID K520 with games that require more graphics performance. For more concurrent sessions with applications that require less graphics performance, NVIDIA recommends the GRID K340 which includes four Kepler-based GPUs and 4GB of memory.

The GRID K2 and K1 are for use to enable rich graphics in virtualized environments. GRID K2 is designed for GRID K2 boards, which include two higher end Kepler GPUs and 8GB of memory, deliver maximum density for users of graphics-intensive applications. GRID K1 boards, which include four Kepler-based GPUs and 16GB of memory, are designed to host the maximum number of concurrent users.

	# GPUs / board	3D Performance per GPU	NVENC Encoder per GPU	Memory (GB) per GPU
K1	4	0.5	1	4
K340	4	1	1	1
K2	2	3	1	4
K520	2	3	1	4

Q13) How many simultaneous Capture and HW encode sessions can run on a single GRID K340 or K1 board? How about for a single GRID K520 or K2 board?

A13) Each Kepler GPU has one NVENC hardware encoder that is capable of supporting up to 6 H.264 high-quality 720p at 30 fps. With GRID K2 and K520 boards, there are 2 Kepler-based GPUs per board. Performance will more likely be hardware encoder limited if more sessions are running concurrently. GRID K1 and K340 boards, there are four Kepler-based GPUs per board. Performance will more likely be limited by the 3D graphic engine. It does have the advantage of having twice as many encoders as K2 and K520. The GPUs in the GRID K520 and K2 are more powerful with higher fill rate (pixels/sec), and geometry rates than the K340 and K1, able to generate more frames/second than the NVENC encoder is able to compress. The total number of concurrent sessions for these boards will depend on the games or applications being streamed, the hardware encoder settings being used, and the CPUs in the system.

Q14) The maximal intra refresh count is 16. How does NvIFR determine the number /or size of macroblocks? Will it be different if we encode with different resolutions? Such 800x600, 1280x720?

A14) The count 16 is a limitation of GRID software and does not depend on resolution. The refresh boundaries, however, will depend on the resolution. The refresh regions always begin and end at the macroblock row boundary. The boundaries are determined based on the resolution and the intra-refresh count.

Q15) When intra-refresh capability is enabled, the frame will be divided into several slices. Are there any issues with decoupling the intra-refresh and slice capability?

A15) Each refresh is a slice because the GPU hardware allows changing motion vector search patterns (intra vs. inter) only on slice boundary hence each refresh region has to be a slice.

Q16) Which Remote Desktop solution is recommended to connect with a GRID server or Amazon G2 instance? Can Microsoft Remote Desktop Protocol be used?

A16a) For setup, debugging, and configuration for these servers, it is recommended to use TeamViewer. This software allows a remote client to connect any one desktop window at a time. In comparison, VNC in comparison will also capture and stream with the NVIDIA GPU accelerated driver, but for baremetal systems, it will capture all desktop windows. This adds additional overhead and results in reduced performance when streaming. The overhead for capturing one desktop window in TeamViewer is significantly less vs capturing all windows desktops with the VNC solution.

Note: TeamViewer uses a proprietary compression format for remote streaming. The NVENC H.264 hardware engine is not used by TeamViewer.

For streaming applications and games (not using H.264 for compression), TeamViewer is the recommended solution for streaming the desktop. For the best experience on GRID, use NVENC with H.264 compression for the best remote streaming experience.

A16b) Can I use Microsoft Windows Remote Desktop? While it is more efficient in terms of bitrate and performance of remote graphics in comparison to VNC, Microsoft Remote Desktop uses a proprietary software based graphics driver that does not support all of the NVIDIA GPU accelerated capabilities, and does not enable the NVIDIA GRID driver. Any applications running under Microsoft Remote Desktop will not be using the NVIDIA driver and will not have full benefits of GPU acceleration.

Q17) How do I perform DirectX 9 Capture and Encode using the GRID SDK?

A17) Please refer to the GRID SDK Sample Description document for guidelines for all of the GRID SDK samples. For simple examples of how to use the API, please refer to these four GRID SDK samples to get started.

```
NvFBCH264
DX9IFRSimpleH264HWEncode
DX9IFRAsyncH264HWEncode
DX9IFRSharedSurfaceH264HWEncode
```

Note: there are other things to consider for your DirectX9 application that needs to be properly handled with software.

When the game loading is complete, the D3DDevice becomes invalid, resulting in a game hang. `IDirect3DDevice9::TestCooperativeLevel()` function will return `D3DERR_DEVICENOTRESET`.

The Shim layer should add check before calling `NvIFRTransferRenderTargetToH264HWEncoder` whether D3D9Device is alive. The application can call `IDirect3DDevice9::TestCooperativeLevel()` function and if it returns `D3DERR_DEVICENOTRESET`, try calling `IDirect3DDevice9::Reset()` function.

If a game instance creates a new device, the shim layer will need to destroy the previous NvIFR context. After that, it can create new NvIFR context with the newly created DirectX device.

Q18) How do I enable NvFBC?

A18) Here are the NvFBC settings

a) ***Windows***: NvFBC needs to be enabled after a clean installation.

{Installation Directory} \bin> NvFBCEnable.exe -enable

Use the NvFBCH264 SDK sample to capture the desktop to a H.264 file.

b) ***Linux***: For the NvFBC GRID API functions, please refer to the flag `NVFBC_CREATE_CAPTURE_SESSION_PARAMS.bWithCursor`

Q19) What are the optimal encoder settings for desktop applications streaming?

A19) With the preset setting `LOW_LATENCY_HP`, a single GPU can encode 6-8 streams (exact number depends on other settings such as RC mode). `LOW_LATENCY_HQ` preset will give 4-6 streams (exact number depends on other settings such as RC mode).

The HQ preset will give you slightly better quality, but the performance will be lower than that of HP. To get the ideal performance for each preset, you can use PerfMSNEC sample application in the SDK.

GRID SDK exposes 3 video encoder presets (`LOW_LATENCY_HQ`, `LOW_LATENCY_HP` and `LOW_LATENCY_DEFAULT`). As the names suggest, these presets are meant for encoding for low-latency applications such as remote streaming, cloud gaming etc. Please refer to the API reference for more details on each parameter.

Q20) What are the recommended hypervisors if I am using virtualized environment?

A20) We recommend XenServer 6.2 (with SP1) for a NMOS for stability and performance. Other hypervisors which are supported are: Citrix, VMware, and KVM. XCP 1.5 is now deprecated.

Note: When using XenServer 6.2 with a K340 in a virtualized environment, the Xen distribution needs to be patched to enable device class configuration space matching. For K520 based servers, this step is not required.

You can find the patch under the XenPatch folder in the SDK.

In the public SDK, it includes the path for Xen. For Xen, to apply the patch:

- `rpm –Uvh xen-device-model-1.8.0-89.7554.i686.rpm`

This is a required step, to ensure that the Windows VM running on Xen can boot with GPU pass through.

Q21) Which Virtual Environments are supported by the GRID SDK?

A21) NMOS (NVIDIA Multi-OS). One GPU is attached to one Virtual Machine.

Q22) Can hybrid card combinations of GRID boards be used in GRID servers

A22) Using hybrid combinations is not recommended. For example, mixing K340 and K520 GPUs in the same system may work fine, but is not the recommended way of setting up a baremetal (or virtualized) system.

Q23) Do I need a specific hardware/system/server to use GRID SDK?

A23) Following Servers and GPUs are recommended for use with the GRID SDK:

GRID GPUs: *NVIDIA GRID K340, K520, K1, and K2*
Desktop GPUs: *NVIDIA Quadro K4000 and higher*
Mobile GPUs: *NVIDIA Quadro K2000M and higher*
GRID Servers:

Supermicro 1027 GRTRF (Intel Platform):
http://www.supermicro.com/products/system/2u/2027/sys-2027gr-trf.cfm

Supermicro 2027 GRTRF (Intel Platform):
http://www.supermicro.com/products/system/1u/1027/sys-1027gr-trf.cfm

Supermicro 1022 GG-TF (AMD Platform):
http://www.supermicro.com/aplus/system/1u/1022/as-1022gg-tf.cfm

Q24) What are the recommended Board configurations in a Baremetal system?

A24) Servers from SuperMicro and Asus are recommended. The actual number and Boards that should be used depends on the server type and use case. The following tables give information about the maximum number of GPUs that can be supported.

Note: For details on how to setup a system with 5xK340, we will be providing this soon.

Supermicro SMCServer 2027 (Intel)	**Max. K340 boards**	**Max. GK107 GPUs**	**Max. K520 boards**	**Max. GK104 GPUs**
Windows Baremetal	3xK340	12	5xK520	10
Linux Baremetal	5xK340	20	5xK520	10
Windows/Linux VMs using XenServer with NMOS	5xK340	20	5xK520	10

Supermicro SMC Server 1027 (Intel)	Max. K340 boards	Max. GK107 GPUs	Max. K520 boards	Max. GK104 GPUs
Windows Baremetal	3xK340	12	3xK520	6
Linux Baremetal	3xK340	12	3xK520	6
Windows/Linux VMs using XenServer with NMOS	3xK340	12	3xK520	6

Supermicro SMC 1022 (AMD)	Max. K340 boards	Max. GK107 GPUs	Max. K520 boards	Max. GK104 GPUs
Windows Baremetal	2xK340	8	2xK520	4
Linux Baremetal	2xK340	8	2xK520	4
Windows/Linux VMs using XenServer with NMOS	2xK340	8	2xK520	4

Q25) What is the recommended SBIOS/IPMI Firmware I should use?

A25) Please contact your NVIDIA GRID support person for the recommended SBIOS/IPMI firmware.

Q26) Are there any special setting I need to do in SBIOS and IPMI for using these Servers for GRID SDK usage?

A26) Yes. For SBIOS: Setting "**VT-d**" for Intel platforms and '**IOMMU**' for AMD platforms has to be enabled for virtualization environment.

Note: For IPMI the FAN speed must be set to optimal for Virtualization and Baremetal environments.

Q27) For AMD servers (i.e. Super Micro 1022GG), when obtaining the system topology information, I am unable to determine which NUMA domain the GRID device is closest to. This system is a NUMA system and the System BIOS for NUMA is enabled. However it fails when making the function call to retrieve the NUMA information. How do I resolve the problem?

A27) Make sure you apply the motherboard has the latest SBIOS "H8DGG.926" or newer for the SuperMicro SM-1022 server. Update it to this SBIOS if it is older than this one.

In order to verify that the GPU and CPU are on the same NUMA node, use a bandwidth test application (Nvidia CUDA SDK contains a test application named ***'bandwidthTest.exe'*** that is suitable for this purpose) to measure the PCI-e bandwidth between the CPU and GPU. You can compare other NUMA nodes to make sure you are getting the best I/O throughput using the bandwidthTest.exe tool from the CUDA Toolkit.

Q28) When setting up a new Server in a Baremetal environment, there are problems installing and using the GRID devices. The Server OS is Windows 7 or Windows 2008 R2. The NVIDIA driver installation succeeds, but there are problems recognizing the GRID GPUs when I launch my application. What is the solution?

A28) Please refer the GRID Game Server Configuration document that is included with the SDK. Sections 1.2 explain how to configure the server and setup the driver. Before installing the NVIDIA driver, please make sure:

a) You are connected through VNC or TeamViewer. Microsoft Remote Desktop uses a software VGA driver, which does not enable the NVIDIA GPU driver.
b) Onboard VGA needs to be disabled. Prior to installing the NVIDIA drivers, the onboard VGA must be disabled. Refer to the SuperMicro server manual or the GRID Game Server Configuration Guide) for information how to disable VGA jumper (**JPG1**).

For Windows 7 and Windows Server 2008 R2, two GPUs from different vendors cannot be used as this is a limitation of the OS, and one GPU vendor driver can be installed and running at a time. If the onboard VGA with the NVIDIA GPUs is required with the GRID server, Windows 8 and Windows Server 2012 both support this.

Note: A benefit for using Windows 8 and Windows Server 2012, is that both allow IPMI management tools to run. If the onboard VGA is disabled, IPMI will is not available.

Q29) For a Baremetal server configuration, when there are very many applications or game sessions using NvIFR are being launched on multiple GPUs, the performance does not scale as expected.

A29) Included with the GRID SDK package, there is a registry key modification. Under the following folder, and click on the **MemoryManagerListSize_96MB.reg** file to install into your registry on the server. This is required in order to achieve proper scaling across all of the GPUs.

{ Installation Directory } \ Tools \ DXG_Kernel_Memory_Limit

Q30) The quality of the H.264 bitstream obtained from the encoder does not appear to be good. Are there settings that should be checked?

A30) The video encoder presets and RC modes exposed in the GRID SDK are optimized for use-cases such as remote streaming, cloud gaming, remote desktop of applications, etc. There is no one set of parameters which will result in good quality under all conditions. For general use cases, please follow these guidelines:

- Try setting the GOP length to -1 (infinite). If you set GOP length too small, frequent I–frames will be generated. If the encoder is set up in low-latency mode, quality of I-frames is low (bit budget allocated for I-frames is low), resulting in frequent flickering effect.
- If you must use periodic I-frames (i.e. if you cannot set GOP length = -1), then try using the rate control mode `eRateControl = NVFBC_H264_ENC_PARAMS_RC_2_PASS_QUALITY`. This rate control mode allows higher bit-budget for I-frames and other high-complexity frames (e.g. scene changes), resulting in better quality I-frames. Note, however, that both 2-pass modes (`NVFBC_H264_ENC_PARAMS_RC_2_PASS_QUALITY` and `NVFBC_H264_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP`) have lower performance than the single-pass modes (such as `NVFBC_H264_ENC_PARAMS_RC_CBR`).
- If running in a streaming/cloud gaming application, set

  ```
  dwVBVBufferSize = dwAvgBitRate/(dwFrameRateNum/ dwFrameRateDen;
  dwVBVInitialDelay = dwVBVBufferSize;
  ```

  The above setting enables a special quality-optimized low-latency mode inside the hardware encoder.

Q31) When streaming an H.264 bitstream, what are some of the methods I can use to recover from channel errors and lost frames on the decoder?

A31) If the decoder loses an entire frame, you can use the following error recovery mechanisms:

- Force an IDR-frame to be generated by setting `bForceIDRFrame = 1` for the next frame on the encoder. This is the easiest and brute-force method for error recovery. However, this method is prone to problems because the IDR frame generated may have a lower quality/larger size (depending upon which RC mode is in use) and since the channel is already bad, there are higher chances of the IDR frame itself getting lost, resulting in cyclically worsening problem.
- Force an intra-refresh wave. This is the recommended method for error recovery. To use this method, set (`bEnableIntraRefresh = 1`) during encoder setup, and then set (`bStartIntraRefresh = 1`) and (`dwIntraRefreshCnt = n`) where *n* is the number of slices in which intra-refresh should be split. In other words, the

next *n* frames will be split in (approximately) equal number of sections (on an MB row boundary) and starting from top, each section will be refreshed with intra MB's. At the end of *n* frames, the entire picture will be refreshed. This method has the advantage of lower channel overhead (since only part of the frame needs to intra-coded), but the disadvantage that it takes slightly longer to recover from the error.

- The third method for error-recovery is to use reference-picture-invalidation. This method works only if there is an upstream communication channel from the decoder to the encoder. To use this method, the decoder signals a lost frame to the encoder and the encoder then invalidates the reference frame by setting (`bInvalidateRefrenceFrames = 1)` and indicate the frames to be invalidated by setting `dwNumRefFramesToInvalidate` and corresponding timestamps in for frames to be invalidated in `ulInvalidFrameTimeStamp[NVFBC_MAX_REF_FRAMES]`. For this to work, the encoder must provide a monotonically increasing timestamp to each frame being encoded through the parameter `ulCaptureTimeStamp`. The reference picture invalidation logic simply matches this timestamp with those passed in `ulInvalidFrameTimeStamp[]` and invalidates those reference frames. This prevents the invalidated frames from being used for subsequently encoded frames as references and prevents error propagation. When encoder exhausts all reference frames (set via `dwMaxNumRefFrames`), it automatically generates an IDR frame.

  Note: In addition to the above, you can also use the error concealment on the decoder so as to avoid discarding an entire frame on the decoder. This works if the error impact is not large.

Q32) How do I capture the mouse input while using NvFBC?

A32) The choice of having HW rendered mouse cursor blended on to the image grabbed by NvFBC is available only with the NvFBCToSys interface. The other two interfaces always blend the HW rendered mouse cursor on the desktop at server-end.

For NvFBCToSys, to enable capturing desktop images with the HW cursor blended on to the grabbed image, please refer to the documentation for the parameter NVFBC_TOSYS_SETUP_PARAMS::bWithHWCursor in the GRID SDK Programming Guide.

In order to enable capturing HW mouse images separately, so that they can be blended at client-side during streaming, please refer to the documentation for the API NvFBCtoSysGrabMouse in the GRID SDK Programming Guide.

Notice

HDMI

OpenCL

Trademarks

Copyright