NVIDIA CAPTURE SDK FAQ

PG-06185-001_v1.04| February 2016

Frequently Asked Questions

DOCUMENT CHANGE HISTORY

PG-06185-001_v1.04

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
1.01	9/16/2014	EY	Added updates on how to assign GPUs to PhysX with GRID baremetal multi-GPU configurations
1.02	11/3/2014	EY	Updated guide on specific GPU products supported.
1.03	7/6/2015	EY	Updated GRID SDK 4.0 guide with Maxwell products supported.
1.04	2/12/2016	EY	Updated for NVIDIA Capture SDK 5.0

TABLE OF CONTENTS

NVIDIA CAPTURE SDK FAQ

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

A1) The NVIDIA Capture SDK, previously called as 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 NVIDIA Capture SDK. Registration is free, and the download page can be found here: https://developer.nvidia.com/grid-app-game-streaming.

Q2) I have downloaded the SDK, how do I get started?

A2) For hardware that is recommend, please refer to Question #23 of this FAQ. Also please refer to the "Guide to Setup a GRID Server" and the NVIDIA Capture 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 NVIDIA Capture SDK? What are they used for?

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

a) **NvFBC** captures (and optionally H.264 encodes) the entire visible desktop to System Memory, DX9, H.264, or CUDA buffers.

b) **NvIFR** captures (and optionally H.264 encodes) from a specific render target to DX9, DX10, DX11, OpenGL, and CUDA buffers.

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 NVIDIA Capture SDK, but NvFBC and NvIFR include functions to configure the H.264 hardware encoder.

Note: GRID SDK 4.0 and NVIDIA Capture SDK 5.0 support H.265 hardware encoding when used with a second generation Maxwell GPU (GM20x).

Q4) What are the interfaces for NvFBC Capture?

A4) There are four different NvFBC interfaces. Please select the appropriate interface, depending on the method of capture that you want, and the destination for the data:

a) `NVFBC_TO_SYS` captures the desktop and copies it to pinned system memory on the host.

b) `NVFBC_TO_CUDA` captures the desktop and copies it to CUDA device memory on the GPU.

c) `NVFBC_TO_DX9VID` captures the desktop and copies it to a DX9 surface. This can then be sent to the HW encoder.

d) `NVFBC_TO_HW_ENCODER` captures the desktop, compresses it using on-chip hardware video encoder to a H.264 or H.265 video stream, and copies the H.264 or H.265 encoded elementary bitstream to system memory on the host.

Note: With NVIDIA Capture SDK 5.0, the NVFBC_TO_H264_HW_ENCODER is replaced by the NVFBC_TO_HW_ENCODER. This is a simple name change to make the interface name video-codec agnostic. NVFBC_TO_DX9VID can be used when running a vGPU profile that supports two or more virtual machines sharing a single GPU. NVFBC_TO_CUDA is not supported for such vGPU profiles.

Q5) What interfaces are available 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) **NvIFRHWENC** captures the render target, compresses it using on-chip hardware video encoder to a H.264 video stream and copies the video encoded elementary bitstream to system memory on the host.

Note: With GRID SDK 4.0 and NVIDIA Capture SDK 5.0, the NvIFRH264 interface is replaced by the NvIFRHWENC interface. This is a simple name change to make the interface name video-codec agnostic.

Q6) When should NvFBC and NvIFR be used?

A6) NvIFR is the preferred solution for capturing 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 NVIDIA Capture SDK?

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

Q9) What Operating Systems are supported by the NVIDIA Capture SDK?

A9) Microsoft Windows 7, Windows 8, Windows Server 2008 R2, Windows Server 2012 R2, Windows 8.1, Windows 10, and Linux OS. For servers running Windows, we recommend Windows 8 and Windows Server 2012 R2 because these operating systems are more efficient and can serve more concurrent users when compared to Windows 7 and Windows Server 2008R2 respectively.

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

A10) No, the NVIDIA Capture SDK is subject to NVIDIA's Software License Agreement. Any re-distribution rights will require explicit permission from NVIDIA.

Q11) I have questions about integrating my software with the NVIDIA Capture SDK. How do I contact support?

A11) Please refer to the Programming Guide and included documentation. For detailed questions, please contact GRID developer support at GridPublicSupport@nvidia.com .

Q12) What are the available GRID products and their features?

A12) The Tesla M6 and M60 are the latest generation of GRID products that enable serving a greater number of users as well as high performance graphical applications running in the data center. Each Tesla M60 board has two Maxwell (GM204) GPUs, while the Tesla M6 board has a single Maxwell (G204) GPU. Tesla M60 is a dual slot board for PCI-e GPU servers and the Tesla M6 has an MXM form factor for blade servers. Tesla M60 also supports twice as many users and video streams as Tesla M6.

GRID K520 and K340 are specifically designed as game streaming servers running in the data center. GRID K520 has two Kepler-based GPUs and 8GB of memory total. It is a high performance GPU with a powerful 3D engine; however, it has fewer encoding hardware, so encoding consists of about ½ the number of video streams compared to a GRID K340. NVIDIA recommends using the GRID K520 with games that require more graphics performance. For more concurrent sessions with lighter graphics, we recommend using the K340 platform.

GRID K2 and K1 are used in enable rich graphics in virtualized environments that target enterprise and workstation applications. GRID K2 includes two higher end Kepler GPUs and 8GB of memory, and deliver the highest density for users of graphically-intensive applications. GRID K1 boards include four Kepler-based GPUs and 16GB of memory. This product is designed to host the maximum number of concurrent users targeting VDI applications.

	# GPUs / board	3D Perf / GPU	NVENC / Board	VMEM / Board
GRID K1	4	0.5	4	16 GB
GRID K340	4	1	4	8 GB
GRID K2	2	3	2	8 GB
GRID K520	2	3	2	8 GB
Tesla M6	1	4.5	2	8 GB
Tesla M60	2	6	4	16 GB

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? How many for a Tesla M6 and M60?

A13) Maxwell GPUs (GM204) have about twice the 3D graphics and compute performance compared to Kepler GPUs (GK104). Each Maxwell GPU also includes two 2nd generation NVENC hardware encoders per GPU. Each Maxwell NVENC is more efficient the Kepler NVENC encoders. Maxwell NVENC also adds H.265 hardware compression. For Tesla M6, it can support up to 40 stream of H.264 encoding high-quality at 720p@30fps. For Tesla M60, it supports up to 80 streams of H.264 encoding high-quality at 720p@30fps.

Each Kepler GPU has one NVENC hardware encoder that is capable of supporting up to six streams of H.264 high-quality 720p@30 fps. With GRID K2 and K520 boards, there are two Kepler-based GPUs per board. Performance will more likely be hardware encoder limited if more sessions are running concurrently. For 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 or 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 begins and ends 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?

A16 (a) 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.

A16 (b) 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 NVIDIA Capture SDK?

A17) Please refer to the NVIDIA Capture SDK Sample Description document for guidelines for all of the NVIDIA Capture SDK samples. For simple examples of how to use the API, please refer to these two NvFBC samples, and three NvIFR samples to get started.

```
NvFBCHWEncode

NvFBCDX9NvEnc
```

Note: NvFBCHWEncode supports vGPU profiles with one virtual machine per GPU, baremetal, and direct attached GPUs/VM and requires a NVIDIA CUDA driver to be present for capture and encode to work.

NVIDIA Capture SDK 5.0 adds a new driver functionality NvFBCDX9NvEnc which supports vGPU profiles with two or more virtual machines sharing a single GPU. These specific profiles do not support the NVIDIA CUDA driver. By using this interface, you can take advantage of NvFBC with NVENC hardware capture.

```
DX9IFRSimpleHWEncode

DX9IFRAsyncHWEncode

DX9IFRSharedSurfaceHWEncode
```

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 is NvFBC enabled?

A18) Here are the NvFBC settings:

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

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

Use the NvFBCHWEncode 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

c) With NVIDIA Capture SDK 5.0, there is a new API that allows you to enable NvFBC through the function NvFBC_Enable(NVFBC_STATE nvFBCState). Refer to the header nvFBC.h and the new NvFBCEnableAPI sample.

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) for GRID K520. 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.

NVIDIA Capture 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 NVIDIA Capture SDK?

A21) NMOS (NVIDIA Multi-OS). One GPU is attached to one Virtual Machine, vGPU profiles that support only one virtual machine per GPU, and vGPU profiles that support two or more virtual machines per GPU.

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. Mixing and matching configurations with Tesla M60 is also not recommended.

Q23) Which NVIDIA GPU products are supported by the NVIDIA Capture SDK?

A23) Kepler GPUs support NVIDIA Capture SDK with capture and H.264 encode. Maxwell second generation GPUs (GM20x) add support for H.265 encode. For capture and video encode, the following GPUs are supported:

GRID	K340	K520	K1	K2
Quadro Desktop	K2000 K2000D K2200	K4000 K4200	K5000 K5200	K6000 M6000
Quadro Mobile	K2000M K2100M K2200M	K3000M K3100M	K4000M K4100M	K5000M K5100M
Tesla	M30 M40	M60		

Fermi GPUs support NVIDIA Capture SDK with capture only:

Quadro Desktop	2000	4000	5000	6000
Quadro Mobile	1000M	3000M		
Tesla	M2070Q			

The following Servers are recommended for use with GRID:

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 GRID Board configurations for 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 for Kepler generation GPUs. For Maxwell generation GPUs, we will provide a revised table in the future

Note: We will soon be providing details on how to set up a system with 5xK340.

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 recommended SBIOS/IPMI Firmware should I use?

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

Q26) What special Server settings do I need to set for the SBIOS and IPMI for use with the NVIDIA Capture SDK?

A26) For the 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) For remote configuration and connection, use either a VNC Server or TeamViewer. Microsoft Remote Desktop uses a software VGA driver, which does not enable the NVIDIA GPU driver upon login.
b) Onboard VGA must be disabled for Windows 7 and Windows Server 2008 R2. 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, GPUs from different vendors cannot be used at the same time. This is due to a restriction in the OS, as only GPU vendor driver can be installed and running at a time. If the onboard VGA is enabled, the PCI-e based boards will not work. If the onboard VGA with the NVIDIA GPUs is required with the GRID server, we recommend to use Windows 8 and Windows Server 2012 R2, as these OS versions remove this restriction.

Note: A benefit for using Windows 8 and Windows Server 2012 R2, 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 NVIDIA Capture 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 NVIDIA Capture SDK are optimized for low-latency 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 low-latency 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_HWENC_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_HWENC_PARAMS_RC_2_PASS_QUALITY` and `NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP`) have lower performance than the single-pass modes (such as `NVFBC_H264_ENC_PARAMS_RC_CBR`).
- If streaming a cloud gaming or desktop application, set the parameters to this:

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

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

Note that the above recommendations for settings are applicable only when prioritizing latency over quality. If best quality is of paramount importance, one must be willing to sacrifice latency by setting higher VBV buffer size. If encode quality has overall higher priority than encode/transmit latency, then start with

```
K = 4;

dwVBVBufferSize  = K * dwAvgBitRate / (dwFrameRateNum/dwFrameRateDen)
```

Then play with some values of 4 to arrive at the best possible latency/quality balance for your use-case/content.

In general, single-frame VBV (`K = 1`) is the worst-possible setting in terms of quality, which should be used only if your use-case is cannot tolerate latency more than 1 frame at all.

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 `lInvalidFrameTimeStamp[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 NVIDIA Capture 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 NVIDIA Capture SDK Programming Guide.

Note: In Linux NvFBC, mouse cursor is always captured.

Q33) Why do I see random crash in multithreaded DX11 NVIFR based application?

A33) One possible reason for such crash could be the multithreading restrictions imposed by DirectX 11. In case of different threads using same ID3D11DeviceContext for calling driver functions, threads should use synchronization objects like critical section, to synchronize access to that ID3DDeviceContext. In NVIDIA Capture SDK sample, DX11IFRAysncHWEncoder sample application different thread uses critical section objects for accessing same ID3D11DeviceContext.

For more info refer to this link:
http://msdn.microsoft.com/en/us/library/windows/desktop/ff476891%28v=vs.85%29.aspx

Q34) How do I assign the appropriate GPU for PhysX on a multiple GPU device system that is running baremetal?

A34) PhysX uses CUDA which has the option to choose to run physics simulations on the GPU for better effects and performance. To specifically set which GPU to use for PhysX, you need to set the following environment flag before launching the game.

- set PHYSXDEVICE_GPU_CUDA_ORDINAL=2

This will use the GPU ordinal 2 for PhysX. You can also refer to the sample under Tools\DXCUDADevices for details on how to enumerate the DirectX and CUDA device ordinals.

Q35) How do I determine when to capture a frame?

We would like to capture a frame only when it has changed from the previous frame. Does a frame grabbed with NVFBC differ from the previous frame?

A35) NVIDIA Capture SDK 5.0 includes a new SDK sample NvFBCDX9DiffMap. This sample will return a 16x16 difference map of the desktop and also capture a downscaled version of the desktop. This is particularily useful for remote desktop environments where the server may decide on a different encoding method depending on what has changed on the screen.

Q36) Does NvFBC support capture on Mosaic?

A36) No. NvFBC does not support capture on mosaic configuration.

Notice

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

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

HDMI

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

OpenCL

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

Trademarks

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

Copyright