# NVIDIA CAPTURE SDK SAMPLE DESCRIPTIONS

DA-06818-001_v03 | February 2016

## Basic Guide to the NVIDIA Capture SDK Samples

# TABLE OF CONTENTS

# NVIDIA CAPTURE SDK OVERVIEW

The NVIDIA Capture SDK includes a collection of samples that demonstrate how to use the SDK APIs to capture either the desktop or a window and then send the contents to the NVIDIA H.264 Encoder. This document explains the purpose of each NVIDIA Capture SDK sample as well as the APIs being used.

## Requirements:

- NVIDIA Capture SDK RC driver, or latest driver including NVFBC and NVIFR
- Visual Studio 2008, Visual Studio 2010, or Visual Studio 2013, to build samples
- Microsoft DirectX SDK (June 2010)
  - NVIFR DirectX 9 samples require DirectX 9, d3d9.dll (included with Windows 7)
  - NVIFR DirectX 10 samples require DirectX SDK (June 2010), d3dx10_43.dll
  - NVIFR DirectX 11 samples require DirectX SDK (June 2010, d3dx11_43.dll, and also dxgi.lib, and the d3dxeffects.lib (included with the samples)
  - NVFBC samples require DirectX 9 and NVIDIA CUDA.
- Visual Studio 2008 solutions require NVIDIA CUDA 5.0
- Visual Studio 2013 solutions require NVIDIA CUDA 6.5
- H.264 encoding requires NVIDIA GPUs based on Kepler architecture or later.
- HEVC encoding requires NVIDIA GPUs based on 2nd Generation Maxwell architecture or newer.

# Sample Applications:

## 1. NvFBCEnableAPI

This sample demonstrates how to use NvFBC_Enable API exported by NVFBC library to programmatically enable or disable NVFBC feature. The API needs to be called from an application that is running with administrator privileges.

## 2. MultiHead

1This sample demonstrates how to grab the frame buffers from multiple displays using NVFBC. This is done by creating multiple NVFBC objects (one for each adapter) and incrementing the NVFBC_TARGET_ADAPTER environment variable before each NvFBC_CreateFunction call - up to the maximum number of adapters in the system. This is accomplished with a helper class in NvFBCLibrary.h (in the Util folder) that initializes the NVFBC dll, sets pointers to the dll functions, and provides methods to set the TargetAdapter and create the NVFBC device.

## 3. NvFBCToSys

This sample demonstrates how to use the NvFBCToSys interface to copy the desktop into a system memory buffer and save it as a file.

## 4. NvFBCHWEncode

This sample demonstrates how to use the INVFBCToHWEncoder interface to grab the desktop and encode to H.264 or HEVC using NVENC HW encoder in a single API call. The INVFBCToHWEncoder interface will be deprecated after NVIDIA Capture SDK 5.0.

## 5. NvFBCCudaNvENC

This sample demonstrates how to use the NvFBCToCuda interface to copy the desktop into a CUDA buffer. From the CUDA buffer, this is then mapped directly to NVENC, where the NVENC hardware video encoder can encode the stream. We recommend that developers use the NVIDIA Video Codec to utilize the NVENC hardware video Encoder. This sample provides a useful wrapper class around the NVENC API, in Encoder.h

## 6. NvFBCCudaSimple

This sample demonstrates how to use the NvFBCToCuda interface to copy the desktop into a CUDA buffer. It covers loading the NvFBC.dll, loading the NVFBC function pointers, creating an instance of NvFBCCuda, and using NvFBCCuda to copy the frame buffer into a CUDA device pointer.

## 7. NvFBCCursorCapture

This sample demonstrates how to use NVFBC's cursor capture feature to copy the desktop and cursor to system memory, where the cursor and desktop are grabbed using separate threads.

## 8. NvFBCDX9DiffMap

This sample demonstrates how to use the NvFBCToDX9 to capture diff maps, and also to capture the desktop to DX9 video memory surfaces.

## 9. NvFBCDX9NvEnc

This sample demonstrates how to use the NvFBCToDX9 to capture to a DirectX 9 surface, and then send it to the NVENC encoder. This sample provides a useful wrapper class around the NVENC API, in Encoder.h

## 10. NvFBCDX9NvEncSharedSurface

The sample demonstrates how to use NVFBC to grab the desktop to a DX9 shared surface, and then send it to the hardware encoder to encode as H.264 or HEVC using a separate DX9 context.

## 11. PerfNVHWENC

This sample acts as a benchmark to measure the maximum performance of the NVIFR encoder.

## 12. DX9/DX10/DX11/GL IFR SimpleSample

This set of simple samples show how to use NVIFR with DirectX 9, 10, 11, or OpenGL to capture a render target to a file.

# 13. DX9/DX10/DX11 AsyncHWEncode

This is a sample shows how to use the NVIFR INVIFRToHWEncoder Interface with DirectX 9, 10, or 11 to capture a render target, compress it to H.264 or HEVC, and write it to a video file. It demonstrates using asynchronous threads to perform rendering, encoding, and write to file.

# 14. DX9IFRSimpleHWEncode

This DirectX 9 sample demonstrates how to capture a render target, compress it, and write it to a video file in a simple single threaded application using the NVIFR INVIFRToHWEncoder Interface.

# 15. DX9IFRSharedSurfaceHWEncode

This DirectX 9 sample demonstrates how to grab and encode a frame using a shared surface with asynchronous render and encode using different DX9 devices for rendering and encoding.

# 16. DX9IFRHWEncMultiGPUStress

This DirectX 9 sample demonstrates how to grab and encode a frame using a shared surface with asynchronous render, and encode DX9 devices running in separate threads. This example shows a method to use all of the available GPUs in a system. In addition, however, writing to a shared surface also allows rendering to occur asynchronously, so that both rendering DX9 device and encoding DX9 device can run at their maximum rate.

This sample uses NvIFRHWEncoder interface, which can generate H.264 as well as HEVC output.

## 17. DXIFR_Shim

This DirectX 9/9Ex/10.0/10.1/11.0 sample demonstrates how to implement a shim layer to intercept D3D API calls, and perform NVIFR encoding on the rendered frames.

It employs a side D3D device dedicated for NVIFR encoding, and the rendered frames are copied by *NVIFR extension* (please refer to NVIDIA Capture SDK Programming Guide for details) for D3D9 and *D3D surface sharing* (please refer to MSDN for details) for D3D9Ex, D3D10 and above.

This sample can't run as a standalone, but should run with a D3D application or game. The shim DLLs, instead of the original DLLs, must be loaded by the D3D application. You can achieve that by placing the shim DLLs into the application directory. The original DLL must be renamed by adding an underscore prefix. For example, d3d9.dll should be renamed as _d3d9.dll. After running, a file named NvIFR.h264 will be generated which contains the encoded H264 stream.

## 18. GLIFR_Shim

This sample shows how to use NvIFROGL to encode the contents of a FBO framebuffer with the hardware H.264 encoder without modifying the OpenGL application. This library intercepts application's glX* calls and introduces NvIFROpenGL APIs in-between the application call and actual glX* call in GL.

## 19. GLIFRAsync

This OpenGL sample shows how to asynchronously transfer an OpenGL FBO Rendertarget while rendering the next frame.

## 20. GLIFRMultiHwEnc

This OpenGL sample shows how to use NvIFROGL to encode the contents of a FBO RenderTarget with the H.264 hardware encoder using multiple GPUs. One thread is created for each stream on a GPU.

## 21. GLIFRPbufferHwEnc

This sample shows how to use NvIFROGL to encode the contents of a pbuffer with the H.264 hardware encoder.

## 22. GLIFRPerfHwEnc

This sample is a performance test for OpenGL NVIFR capture and H.264 hardware encoding. Settings used are 2-pass quality with low latency HP at 720p resolution, 5 mbps, and 30 fps.

## 23. GLIFRSimpleHwEnc

This sample shows how to use NvIFROGL to encode the contents of a FBO framebuffer with the hardware H.264 encoder. This is a simpler example showing how to use the SDK.

## 24. GLIFRSimplePBuffer

This sample shows how to initialize NvIFROGL and transfer the contents of a pbuffer to the host and write to a TGA bitmap file. A color 3D cube is drawn to the pbuffer.

## 25. GLIFRThreadedHwEnc

This sample shows how to use NvIFROGL to encode the contents of a FBO frame buffer with the hardware H.264 encoder with multiple threads. One thread triggers the encoding while the other thread reads back the encoded bitstream.

## 26. GLIFRNVENCDynRes

This sample shows how to handle dynamic resolution changes while using NvIFROGL to encode the contents of a FBO frame buffer with the hardware H.264 encoder.

## 27. GLIFRNVENCGetCaps

This sample shows how to use NvIFROGL API to check for capabilities and supported features of the HW Encoder.

## Notice



## Trademarks



## Copyright



**DA-06818-001_v03**