



**NVIDIA Framebuffer Capture API for Linux
Reference Manual**

March 24, 2020

Version 7.1.9



Contents

| | | |
|----------|--|-----------|
| 1 | NVIDIA Framebuffer Capture (NvFBC) for Linux. | 1 |
| 2 | Legal Notice | 3 |
| 3 | Deprecated List | 5 |
| 4 | Module Index | 9 |
| 4.1 | Modules | 9 |
| 5 | Class Index | 11 |
| 5.1 | Class List | 11 |
| 6 | File Index | 13 |
| 6.1 | File List | 13 |
| 7 | Module Documentation | 15 |
| 7.1 | Requirements | 15 |
| 7.2 | ChangeLog | 16 |
| 7.3 | Structure Definition | 18 |
| 7.3.1 | Typedef Documentation | 23 |
| 7.3.1.1 | NVFBC_BOX | 23 |
| 7.3.1.2 | NVFBC_RANDR_OUTPUT_INFO | 23 |
| 7.3.1.3 | NVFBCSTATUS | 23 |
| 7.3.2 | Enumeration Type Documentation | 23 |
| 7.3.2.1 | _NVFBC_BOOL | 23 |
| 7.3.2.2 | _NVFBC_BUFFER_FORMAT | 24 |
| 7.3.2.3 | _NVFBC_CAPTURE_TYPE | 24 |
| 7.3.2.4 | _NVFBCSTATUS | 24 |
| 7.3.2.5 | NVFBC_TOCUDA_FLAGS | 25 |
| 7.3.2.6 | NVFBC_TOGL_FLAGS | 26 |
| 7.3.2.7 | NVFBC_TOSYS_GRAB_FLAGS | 26 |

| | | |
|---------|-------------------------------------|----|
| 7.3.2.8 | NVFBC_TRACKING_TYPE | 27 |
| 7.4 | Deprecated Structure Definition | 28 |
| 7.4.1 | Detailed Description | 30 |
| 7.4.2 | Define Documentation | 30 |
| 7.4.2.1 | NVFBC_HWENC_CONFIG_VER | 30 |
| 7.4.2.2 | NVFBC_HWENC_ENCODE_PARAMS_VER | 30 |
| 7.4.2.3 | NVFBC_HWENC_FRAME_INFO_VER | 30 |
| 7.4.2.4 | NVFBC_MAX_REF_FRAMES | 31 |
| 7.4.2.5 | NVFBC_TOHWENC_GET_CAPS_PARAMS_VER | 31 |
| 7.4.2.6 | NVFBC_TOHWENC_GET_HEADER_PARAMS_VER | 31 |
| 7.4.2.7 | NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER | 31 |
| 7.4.2.8 | NVFBC_TOHWENC_SETUP_PARAMS_VER | 31 |
| 7.4.3 | Typedef Documentation | 31 |
| 7.4.3.1 | NVFBC_HWENC_CONFIG | 31 |
| 7.4.3.2 | NVFBC_HWENC_ENCODE_PARAMS | 31 |
| 7.4.3.3 | NVFBC_HWENC_FRAME_INFO | 32 |
| 7.4.3.4 | NVFBC_HWENC_PARAMS_RC_MODE | 32 |
| 7.4.3.5 | NVFBC_TOHWENC_GET_CAPS_PARAMS | 32 |
| 7.4.3.6 | NVFBC_TOHWENC_GET_HEADER_PARAMS | 32 |
| 7.4.3.7 | NVFBC_TOHWENC_GRAB_FRAME_PARAMS | 32 |
| 7.4.3.8 | NVFBC_TOHWENC_SETUP_PARAMS | 32 |
| 7.4.4 | Enumeration Type Documentation | 32 |
| 7.4.4.1 | _NVFBC_HWENC_PARAMS_RC_MODE | 32 |
| 7.4.4.2 | NVFBC_HWENC_CODEEC | 33 |
| 7.4.4.3 | NVFBC_HWENC_PARAM_FLAGS | 33 |
| 7.4.4.4 | NVFBC_HWENC_PRESET | 33 |
| 7.4.4.5 | NVFBC_HWENC_SLICING_MODE | 34 |
| 7.4.4.6 | NVFBC_TOHWENC_GRAB_FLAGS | 34 |
| 7.5 | API Entry Points | 35 |
| 7.5.1 | Detailed Description | 36 |
| 7.5.2 | Function Documentation | 37 |
| 7.5.2.1 | NvFBCBindContext | 37 |
| 7.5.2.2 | NvFBCCreateCaptureSession | 37 |
| 7.5.2.3 | NvFBCCreateHandle | 38 |
| 7.5.2.4 | NvFBCCreateInstance | 38 |
| 7.5.2.5 | NvFBCEndDestroyCaptureSession | 39 |
| 7.5.2.6 | NvFBCEndDestroyHandle | 39 |

| | | |
|----------|---|-----------|
| 7.5.2.7 | NvFBCLastErrorStr | 40 |
| 7.5.2.8 | NvFBCLastStatus | 40 |
| 7.5.2.9 | NvFBCLReleaseContext | 40 |
| 7.5.2.10 | NvFBCToCudaGrabFrame | 41 |
| 7.5.2.11 | NvFBCToCudaSetup | 41 |
| 7.5.2.12 | NvFBCToGLGrabFrame | 42 |
| 7.5.2.13 | NvFBCToGLSetup | 42 |
| 7.5.2.14 | NvFBCToH264GetHeader | 43 |
| 7.5.2.15 | NvFBCToH264GrabFrame | 43 |
| 7.5.2.16 | NvFBCToH264Setup | 44 |
| 7.5.2.17 | NvFBCToHwEncGetCaps | 45 |
| 7.5.2.18 | NvFBCToHwEncGetHeader | 45 |
| 7.5.2.19 | NvFBCToHwEncGrabFrame | 46 |
| 7.5.2.20 | NvFBCToHwEncSetup | 47 |
| 7.5.2.21 | NvFBCToSysGrabFrame | 47 |
| 7.5.2.22 | NvFBCToSysSetup | 48 |
| 8 | Class Documentation | 49 |
| 8.1 | _NVFBC_BIND_CONTEXT_PARAMS Struct Reference | 49 |
| 8.1.1 | Detailed Description | 49 |
| 8.2 | _NVFBC_BOX Struct Reference | 50 |
| 8.2.1 | Detailed Description | 50 |
| 8.3 | _NVFBC_CREATE_CAPTURE_SESSION_PARAMS Struct Reference | 51 |
| 8.3.1 | Detailed Description | 51 |
| 8.3.2 | Member Data Documentation | 52 |
| 8.3.2.1 | bDisableAutoModesetRecovery | 52 |
| 8.3.2.2 | bPushModel | 52 |
| 8.3.2.3 | bRoundFrameSize | 52 |
| 8.3.2.4 | bWithCursor | 52 |
| 8.3.2.5 | captureBox | 53 |
| 8.3.2.6 | dwSamplingRateMs | 53 |
| 8.3.2.7 | eCaptureType | 53 |
| 8.3.2.8 | frameSize | 53 |
| 8.4 | _NVFBC_CREATE_HANDLE_PARAMS Struct Reference | 54 |
| 8.4.1 | Detailed Description | 54 |
| 8.4.2 | Member Data Documentation | 54 |
| 8.4.2.1 | bExternallyManagedContext | 54 |

| | | |
|----------|--|----|
| 8.4.2.2 | glxCtx | 54 |
| 8.4.2.3 | glxFBConfig | 55 |
| 8.5 | _NVFBC_DESTROY_CAPTURE_SESSION_PARAMS Struct Reference | 56 |
| 8.5.1 | Detailed Description | 56 |
| 8.6 | _NVFBC_DESTROY_HANDLE_PARAMS Struct Reference | 57 |
| 8.6.1 | Detailed Description | 57 |
| 8.7 | _NVFBC_FRAME_GRAB_INFO Struct Reference | 58 |
| 8.7.1 | Detailed Description | 58 |
| 8.7.2 | Member Data Documentation | 58 |
| 8.7.2.1 | bIsNewFrame | 58 |
| 8.7.2.2 | dwCurrentFrame | 59 |
| 8.7.2.3 | ulTimestampUs | 59 |
| 8.8 | _NVFBC_GET_STATUS_PARAMS Struct Reference | 60 |
| 8.8.1 | Detailed Description | 60 |
| 8.8.2 | Member Data Documentation | 60 |
| 8.8.2.1 | bXRandRAvailable | 60 |
| 8.8.2.2 | dwOutputNum | 61 |
| 8.8.2.3 | outputs | 61 |
| 8.9 | _NVFBC_HWENC_CONFIG Struct Reference | 62 |
| 8.9.1 | Detailed Description | 63 |
| 8.9.2 | Member Data Documentation | 63 |
| 8.9.2.1 | bEnableAQ | 63 |
| 8.9.2.2 | bEnableIntraRefresh | 63 |
| 8.9.2.3 | bEnableMSE | 64 |
| 8.9.2.4 | bOutBandSPSPPS | 64 |
| 8.9.2.5 | dwAvgBitRate | 64 |
| 8.9.2.6 | dwFrameRateDen | 64 |
| 8.9.2.7 | dwFrameRateNum | 64 |
| 8.9.2.8 | dwGOPLength | 64 |
| 8.9.2.9 | dwMaxNumRefFrames | 64 |
| 8.9.2.10 | dwProfile | 65 |
| 8.9.2.11 | dwVBVBufferSize | 65 |
| 8.9.2.12 | dwVBVInitialDelay | 65 |
| 8.9.2.13 | eInputBufferFormat | 65 |
| 8.10 | _NVFBC_HWENC_ENCODE_PARAMS Struct Reference | 66 |
| 8.10.1 | Detailed Description | 66 |
| 8.10.2 | Member Data Documentation | 67 |

| | | |
|----------|--|----|
| 8.10.2.1 | bInvalidateReferenceFrames | 67 |
| 8.10.2.2 | bReEncodePrevFrame | 67 |
| 8.10.2.3 | dwEncodeParamFlags | 67 |
| 8.10.2.4 | dwNewVBVBufferSize | 67 |
| 8.10.2.5 | dwNewVBVInitialDelay | 67 |
| 8.11 | _NVFBC_HWENC_FRAME_INFO Struct Reference | 68 |
| 8.11.1 | Detailed Description | 68 |
| 8.12 | _NVFBC_OUTPUT Struct Reference | 69 |
| 8.12.1 | Detailed Description | 69 |
| 8.12.2 | Member Data Documentation | 69 |
| 8.12.2.1 | name | 69 |
| 8.13 | _NVFBC_RELEASE_CONTEXT_PARAMS Struct Reference | 70 |
| 8.13.1 | Detailed Description | 70 |
| 8.14 | _NVFBC_SIZE Struct Reference | 71 |
| 8.14.1 | Detailed Description | 71 |
| 8.15 | _NVFBC_TOCUDA_GRAB_FRAME_PARAMS Struct Reference | 72 |
| 8.15.1 | Detailed Description | 72 |
| 8.15.2 | Member Data Documentation | 72 |
| 8.15.2.1 | dwTimeoutMs | 72 |
| 8.15.2.2 | pCUDADeviceBuffer | 73 |
| 8.15.2.3 | pFrameGrabInfo | 73 |
| 8.16 | _NVFBC_TOCUDA_SETUP_PARAMS Struct Reference | 74 |
| 8.16.1 | Detailed Description | 74 |
| 8.17 | _NVFBC_TOGL_GRAB_FRAME_PARAMS Struct Reference | 75 |
| 8.17.1 | Detailed Description | 75 |
| 8.17.2 | Member Data Documentation | 75 |
| 8.17.2.1 | dwTextureIndex | 75 |
| 8.17.2.2 | dwTimeoutMs | 75 |
| 8.17.2.3 | pFrameGrabInfo | 76 |
| 8.18 | _NVFBC_TOGL_SETUP_PARAMS Struct Reference | 77 |
| 8.18.1 | Detailed Description | 77 |
| 8.18.2 | Member Data Documentation | 77 |
| 8.18.2.1 | diffMapSize | 77 |
| 8.18.2.2 | dwDiffMapScalingFactor | 78 |
| 8.18.2.3 | dwTextures | 78 |
| 8.18.2.4 | ppDiffMap | 78 |
| 8.19 | _NVFBC_TOHWENC_GET_CAPS_PARAMS Struct Reference | 79 |

| | |
|--|----|
| 8.19.1 Detailed Description | 80 |
| 8.19.2 Member Data Documentation | 80 |
| 8.19.2.1 bCodecSupported | 80 |
| 8.20 _NVFBC_TOHWENC_GET_HEADER_PARAMS Struct Reference | 81 |
| 8.20.1 Detailed Description | 81 |
| 8.20.2 Member Data Documentation | 81 |
| 8.20.2.1 pBuffer | 81 |
| 8.21 _NVFBC_TOHWENC_GRAB_FRAME_PARAMS Struct Reference | 82 |
| 8.21.1 Detailed Description | 82 |
| 8.21.2 Member Data Documentation | 82 |
| 8.21.2.1 dwMSE | 82 |
| 8.21.2.2 pFrameGrabInfo | 82 |
| 8.21.2.3 ppBitStreamBuffer | 83 |
| 8.22 _NVFBC_TOHWENC_SETUP_PARAMS Struct Reference | 84 |
| 8.22.1 Detailed Description | 84 |
| 8.23 _NVFBC_TOSYS_GRAB_FRAME_PARAMS Struct Reference | 85 |
| 8.23.1 Detailed Description | 85 |
| 8.23.2 Member Data Documentation | 85 |
| 8.23.2.1 dwTimeoutMs | 85 |
| 8.23.2.2 pFrameGrabInfo | 85 |
| 8.24 _NVFBC_TOSYS_SETUP_PARAMS Struct Reference | 87 |
| 8.24.1 Detailed Description | 87 |
| 8.24.2 Member Data Documentation | 87 |
| 8.24.2.1 diffMapSize | 87 |
| 8.24.2.2 dwDiffMapScalingFactor | 87 |
| 8.24.2.3 ppBuffer | 88 |
| 8.24.2.4 ppDiffMap | 88 |
| 8.25 NVFBC_API_FUNCTION_LIST Struct Reference | 89 |
| 8.25.1 Detailed Description | 90 |
| 8.25.2 Member Data Documentation | 90 |
| 8.25.2.1 dwVersion | 90 |
| 8.25.2.2 nvFBCBindContext | 90 |
| 8.25.2.3 nvFBCCreateCaptureSession | 90 |
| 8.25.2.4 nvFBCCreateHandle | 90 |
| 8.25.2.5 nvFBCDestroyCaptureSession | 91 |
| 8.25.2.6 nvFBCDestroyHandle | 91 |
| 8.25.2.7 nvFBCGetLastErrorStr | 91 |

| | | |
|-----------|--|-----------|
| 8.25.2.8 | nvFBCGetStatus | 91 |
| 8.25.2.9 | nvFBCReleaseContext | 91 |
| 8.25.2.10 | nvFBCToCudaGrabFrame | 91 |
| 8.25.2.11 | nvFBCToCudaSetUp | 91 |
| 8.25.2.12 | nvFBCToGLGrabFrame | 91 |
| 8.25.2.13 | nvFBCToGLSetUp | 91 |
| 8.25.2.14 | nvFBCToH264GetHeader | 91 |
| 8.25.2.15 | nvFBCToH264GrabFrame | 91 |
| 8.25.2.16 | nvFBCToH264SetUp | 92 |
| 8.25.2.17 | nvFBCToHwEncGetCaps | 92 |
| 8.25.2.18 | nvFBCToHwEncGetHeader | 92 |
| 8.25.2.19 | nvFBCToHwEncGrabFrame | 92 |
| 8.25.2.20 | nvFBCToHwEncSetUp | 92 |
| 8.25.2.21 | nvFBCToSysGrabFrame | 92 |
| 8.25.2.22 | nvFBCToSysSetUp | 92 |
| 9 | File Documentation | 93 |
| 9.1 | NvFBC.h File Reference | 93 |
| 9.1.1 | Detailed Description | 102 |

Chapter 1

NVIDIA Framebuffer Capture (NvFBC) for Linux.

NvFBC is a high performance, low latency API to capture and optionally compress the framebuffer of an X server screen. The output from NvFBC captures everything that would be visible if we were directly looking at the monitor. This includes window manager decoration, mouse cursor, overlay, etc.

It is ideally suited to desktop or fullscreen application capture and remoting.

Chapter 2

Legal Notice

Copyright (c) 2011-2018 NVIDIA Corporation.

All rights reserved.

Notice

This source code and/or documentation ("Licensed Deliverables") are subject to NVIDIA intellectual property rights under U.S. and international Copyright laws.

These Licensed Deliverables contained herein is PROPRIETARY and to NVIDIA and is being provided under the terms and conditions of a form of NVIDIA software license agreement by and between NVIDIA and Licensee ("License Agreement") or electronically accepted by Licensee. Notwithstanding any terms or conditions to the contrary in the License Agreement, reproduction or disclosure of the Licensed Deliverables to any third party without the express written consent of NVIDIA is prohibited.

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND. NVIDIA DISCLAIMS ALL WARRANTIES WITH REGARD TO THESE LICENSED DELIVERABLES, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. NOTWITHSTANDING ANY TERMS OR CONDITIONS TO THE CONTRARY IN THE LICENSE AGREEMENT, IN NO EVENT SHALL NVIDIA BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THESE LICENSED DELIVERABLES.

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. This publication supersedes and replaces all other information previously supplied.

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

U.S. Government End Users. These Licensed Deliverables are a "commercial item" as that term is defined at 48 C.F.R. 2.101 (OCT * 1995), consisting of "commercial computer software" and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (SEPT 1995) and is provided to the U.S. Government only as a commercial end item. Consistent with 48 C.F.R.12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (JUNE 1995), all U.S. Government End Users acquire the Licensed Deliverables with only those rights set forth herein.

Any use of the Licensed Deliverables in individual and commercial software must include, in the user documentation and internal comments to the code, the above Disclaimer and U.S. Government End Users Notice.

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.

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.

Chapter 3

Deprecated List

Class [_NVFBC_HWENC_CONFIG](#) Describes HW encoder configuration.

Class [_NVFBC_HWENC_ENCODE_PARAMS](#) Describes encode parameters.

Class [_NVFBC_HWENC_FRAME_INFO](#) Describes an encoded frame.

Class [_NVFBC_TOHWENC_GET_CAPS_PARAMS](#) Defines parameters for the ToHwGetCaps() API call.

Class [_NVFBC_TOHWENC_GET_HEADER_PARAMS](#) Defines parameters for the [NvFBCToHwEncGetHeader\(\)](#) API call.

Class [_NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#) Defines parameters for the [NvFBCToHwEncGrabFrame\(\)](#) API call.

Class [_NVFBC_TOHWENC_SETUP_PARAMS](#) Defines parameters for the ToHwEncSetUp() API call.

Member [NVFBC_CAPTURE_TO_HW_ENCODER](#) Capture HW compressed frames to a buffer in system memory.

Member [_NVFBC_HWENC_PARAMS_RC_MODE](#) Defines encoder rate control modes.

Member [NVFBC_HWENC_CODEC](#) Defines video codecs.

Member [NVFBC_HWENC_PARAM_FLAGS](#) Defines encoder flags.

Member [NVFBC_HWENC_PRESET](#) Defines encoder presets.

Member [NVFBC_HWENC_SLICING_MODE](#) Defines slicing modes.

Member [NVFBC_TOHWENC_GRAB_FLAGS](#) Defines flags that can be used when capturing HW encoded compressed frames to system memory.

Member [NVFBC_HWENC_CONFIG_VER](#) NVFBC_HWENC_CONFIG structure version.

Member [NVFBC_HWENC_ENCODE_PARAMS_VER](#) NVFBC_HWENC_ENCODE_PARAMS structure version.

Member [NVFBC_HWENC_FRAME_INFO_VER](#) NVFBC_HWENC_FRAME_INFO structure version.

Member NVFBC_MAX_REF_FRAMES Maximum number of reference frames.

Member NVFBC_TOHWENC_GET_CAPS_PARAMS_VER NVFBC_TOHWENC_GET_CAPS_PARAMS structure version.

Member NVFBC_TOHWENC_GET_HEADER_PARAMS_VER NVFBC_TOHWENC_GET_HEADER_PARAMS structure version.

Member NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER NVFBC_TOHWENC_GRAB_FRAME_PARAMS structure version.

Member NVFBC_TOHWENC_SETUP_PARAMS_VER NVFBC_TOHWENC_SETUP_PARAMS structure version.

Member NvFBCToH264GetHeader

Member NvFBCToH264GrabFrame

Member NvFBCToH264SetUp

Member NvFBCToHwEncGetCaps Queries HW encoder capabilities for a given codec.

Member NvFBCToHwEncGetHeader Gets SPS/PPS headers

Member NvFBCToHwEncGrabFrame Captures a HW compressed frame to a bitstream in system memory.

Member NvFBCToHwEncSetUp Sets up a capture to HW compressed frames in system memory.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

| | |
|---|----|
| Requirements | 15 |
| ChangeLog | 16 |
| Structure Definition | 18 |
| Deprecated Structure Definition | 28 |
| API Entry Points | 35 |

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

| | |
|---|----|
| _NVFBC_BIND_CONTEXT_PARAMS (Defines parameters for the NvFBCBindContext() API call) . . . | 49 |
| _NVFBC_BOX (Box used to describe an area of the tracked region to capture) | 50 |
| _NVFBC_CREATE_CAPTURE_SESSION_PARAMS (Defines parameters for the NvFBCCreateCaptureSession() API call) | 51 |
| _NVFBC_CREATE_HANDLE_PARAMS (Defines parameters for the CreateHandle() API call) | 54 |
| _NVFBC_DESTROY_CAPTURE_SESSION_PARAMS (Defines parameters for the NvFBCDestroyCaptureSession() API call) | 56 |
| _NVFBC_DESTROY_HANDLE_PARAMS (Defines parameters for the NvFBCDestroyHandle() API call) | 57 |
| _NVFBC_FRAME_GRAB_INFO (Describes information about a captured frame) | 58 |
| _NVFBC_GET_STATUS_PARAMS (Defines parameters for the NvFBCGetStatus() API call) | 60 |
| _NVFBC_HWENC_CONFIG | 62 |
| _NVFBC_HWENC_ENCODE_PARAMS | 66 |
| _NVFBC_HWENC_FRAME_INFO | 68 |
| _NVFBC_OUTPUT (Describes an RandR output) | 69 |
| _NVFBC_RELEASE_CONTEXT_PARAMS (Defines parameters for the NvFBCReleaseContext() API call) | 70 |
| _NVFBC_SIZE (Size used to describe the size of a frame) | 71 |
| _NVFBC_TOCUDA_GRAB_FRAME_PARAMS (Defines parameters for the NvFBCToCudaGrabFrame() API call) | 72 |
| _NVFBC_TOCUDA_SETUP_PARAMS (Defines parameters for the NvFBCToCudaSetup() API call) . . | 74 |
| _NVFBC_TOGL_GRAB_FRAME_PARAMS (Defines parameters for the NvFBCToGLGrabFrame() API call) | 75 |
| _NVFBC_TOGL_SETUP_PARAMS (Defines parameters for the NvFBCToGLSetup() API call) | 77 |
| _NVFBC_TOHWENC_GET_CAPS_PARAMS | 79 |
| _NVFBC_TOHWENC_GET_HEADER_PARAMS | 81 |
| _NVFBC_TOHWENC_GRAB_FRAME_PARAMS | 82 |
| _NVFBC_TOHWENC_SETUP_PARAMS | 84 |
| _NVFBC_TOSYS_GRAB_FRAME_PARAMS (Defines parameters for the NvFBCToSysGrabFrame() API call) | 85 |
| _NVFBC_TOSYS_SETUP_PARAMS (Defines parameters for the NvFBCToSysSetup() API call) | 87 |
| NVFBC_API_FUNCTION_LIST (Structure populated with API function pointers) | 89 |

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

| | |
|---|----|
| NvFBC.h (This file contains the interface constants, structure definitions and function prototypes defining the NvFBC API for Linux) | 93 |
|---|----|

Chapter 7

Module Documentation

7.1 Requirements

The following requirements are provided by the regular NVIDIA Display Driver package:.

The following requirements are provided by the regular NVIDIA Display Driver package:.

- OpenGL core ≥ 4.2 : Mandatory. NvFBC relies on OpenGL to perform frame capture and post-processing.
- `libcuda.so.1` ≥ 5.5 : Optional. Used for capture to video memory with CUDA interop and capture to HW compressed frames.
- `libnvidia-encode.so.1` ≥ 5.0 : Optional. Used for capture to HW compressed frames.

The following requirements must be installed separately depending on the Linux distribution being used:

- XRandR extension ≥ 1.2 : Optional. Used for RandR output tracking.
- `libX11-xcb.so.1` ≥ 1.2 : Mandatory. NvFBC uses a mix of Xlib and XCB. Xlib is needed to use GLX, XCB is needed to make NvFBC more resilient against X server terminations while a capture session is active.
- `libxcb.so.1` ≥ 1.3 : Mandatory. See above.
- `xorg-server` ≥ 1.3 : Optional. Required for push model to work properly.

Note that all optional dependencies are `dlopen()`'d at runtime. Failure to load an optional library is not fatal.

7.2 ChangeLog

NvFBC Linux API version 0.1

- Initial BETA release.

NvFBC Linux API version 0.1

- Initial BETA release.

NvFBC Linux API version 0.2

- Added 'bEnableMSE' field to NVFBC_H264_HW_ENC_CONFIG.
- Added 'dwMSE' field to NVFBC_TOH264_GRAB_FRAME_PARAMS.
- Added 'bEnableAQ' field to NVFBC_H264_HW_ENC_CONFIG.
- Added 'NVFBC_H264_PRESET_LOSSLESS_HP' enum to NVFBC_H264_PRESET.
- Added 'NVFBC_BUFFER_FORMAT_YUV444P' enum to NVFBC_BUFFER_FORMAT.
- Added 'eInputBufferFormat' field to NVFBC_H264_HW_ENC_CONFIG.
- Added '0' and '244' values for NVFBC_H264_HW_ENC_CONFIG::dwProfile.

NvFBC Linux API version 0.3

- Improved multi-threaded support by implementing an API locking mechanism.
- Added 'nvFBCBindContext' API entry point.
- Added 'nvFBCReleaseContext' API entry point.

NvFBC Linux API version 1.0

- Added codec agnostic interface for HW encoding.
- Deprecated H.264 interface.
- Added support for H.265/HEVC HW encoding.

NvFBC Linux API version 1.1

- Added 'nvFBCToHwGetCaps' API entry point.
- Added 'dwDiffMapScalingFactor' field to NVFBC_TOSYS_SETUP_PARAMS.

NvFBC Linux API version 1.2

- Deprecated ToHwEnc interface.
- Added ToGL interface that captures frames to an OpenGL texture in video memory.
- Added 'bDisableAutoModesetRecovery' field to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.
- Added 'bExternallyManagedContext' field to NVFBC_CREATE_HANDLE_PARAMS.

NvFBC Linux API version 1.3

- Added NVFBC_BUFFER_FORMAT_RGBA
- Added 'dwTimeoutMs' field to NVFBC_TOSYS_GRAB_FRAME_PARAMS, NVFBC_TOCUDA_GRAB_FRAME_PARAMS, and NVFBC_TOGL_GRAB_FRAME_PARAMS.

NvFBC Linux API version 1.4

- Clarified that NVFBC_BUFFER_FORMAT_{ARGB,RGB,RGBA} are byte-order formats.
- Renamed NVFBC_BUFFER_FORMAT_YUV420P to NVFBC_BUFFER_FORMAT_NV12.
- Added new requirements.
- Made NvFBC more resilient against the X server terminating during an active capture session. See new comments for [NVFBC_ERR_X](#).
- Relaxed requirement that 'frameSize' must have a width being a multiple of 4 and a height being a multiple of 2.
- Added 'bRoundFrameSize' field to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.
- Relaxed requirement that the scaling factor for differential maps must be a multiple of the size of the frame.
- Added 'diffMapSize' field to NVFBC_TOSYS_SETUP_PARAMS and NVFBC_TOGL_SETUP_PARAMS.

NvFBC Linux API version 1.5

- Added NVFBC_BUFFER_FORMAT_BGRA

NvFBC Linux API version 1.6

- Added the 'NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY', 'NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY', and 'NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY' capture flags.
- Exposed debug and performance logs through the NVFBC_LOG_LEVEL environment variable. Setting it to "1" enables performance logs, setting it to "2" enables debugging logs, setting it to "3" enables both.
- Logs are printed to stdout or to the file pointed by the NVFBC_LOG_FILE environment variable.
- Added 'dwTimestampUs' to NVFBC_FRAME_GRAB_INFO.
- Added 'dwSamplingRateMs' to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.
- Added 'bPushModel' to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.

7.3 Structure Definition

Classes

- struct [_NVFBC_BOX](#)
Box used to describe an area of the tracked region to capture.
- struct [_NVFBC_SIZE](#)
Size used to describe the size of a frame.
- struct [_NVFBC_FRAME_GRAB_INFO](#)
Describes information about a captured frame.
- struct [_NVFBC_CREATE_HANDLE_PARAMS](#)
Defines parameters for the `CreateHandle()` API call.
- struct [_NVFBC_DESTROY_HANDLE_PARAMS](#)
Defines parameters for the `NvFBCDestroyHandle()` API call.
- struct [_NVFBC_OUTPUT](#)
Describes an RandR output.
- struct [_NVFBC_GET_STATUS_PARAMS](#)
Defines parameters for the `NvFBCGetStatus()` API call.
- struct [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS](#)
Defines parameters for the `NvFBCCreateCaptureSession()` API call.
- struct [_NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#)
Defines parameters for the `NvFBCDestroyCaptureSession()` API call.
- struct [_NVFBC_BIND_CONTEXT_PARAMS](#)
Defines parameters for the `NvFBCBindContext()` API call.
- struct [_NVFBC_RELEASE_CONTEXT_PARAMS](#)
Defines parameters for the `NvFBCReleaseContext()` API call.
- struct [_NVFBC_TOSYS_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToSysSetUp()` API call.
- struct [_NVFBC_TOSYS_GRAB_FRAME_PARAMS](#)
Defines parameters for the `NvFBCToSysGrabFrame()` API call.
- struct [_NVFBC_TOCUDA_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToCudaSetUp()` API call.
- struct [_NVFBC_TOCUDA_GRAB_FRAME_PARAMS](#)
Defines parameters for the `NvFBCToCudaGrabFrame()` API call.
- struct [_NVFBC_TOGL_SETUP_PARAMS](#)

Defines parameters for the `NvFBCToGLSetUp()` API call.

- struct `_NVFBC_TOGL_GRAB_FRAME_PARAMS`
Defines parameters for the `NvFBCToGLGrabFrame()` API call.
- struct `NVFBC_API_FUNCTION_LIST`
Structure populated with API function pointers.

Defines

- #define `NVFBCAPI`
Calling convention.
- #define `NVFBC_VERSION_MAJOR` 1
NvFBC API major version.
- #define `NVFBC_VERSION_MINOR` 6
NvFBC API minor version.
- #define `NVFBC_VERSION` (uint32_t) (NVFBC_VERSION_MINOR | (NVFBC_VERSION_MAJOR << 8))

NvFBC API version.
- #define `NVFBC_STRUCT_VERSION`(typeName, ver) (uint32_t) (sizeof(typeName) | ((ver) << 16) | (NVFBC_VERSION << 24))
Creates a version number for structure parameters.
- #define `NVFBC_ERR_STR_LEN` 512
Maximum size in bytes of an error string.
- #define `NVFBC_CREATE_HANDLE_PARAMS_VER` NVFBC_STRUCT_VERSION(NVFBC_CREATE_HANDLE_PARAMS, 2)
NVFBC_CREATE_HANDLE_PARAMS structure version.
- #define `NVFBC_DESTROY_HANDLE_PARAMS_VER` NVFBC_STRUCT_VERSION(NVFBC_DESTROY_HANDLE_PARAMS, 1)
NVFBC_DESTROY_HANDLE_PARAMS structure version.
- #define `NVFBC_OUTPUT_MAX` 5
Maximum number of connected RandR outputs to an X screen.
- #define `NVFBC_OUTPUT_NAME_LEN` 128
Maximum size in bytes of an RandR output name.
- #define `NVFBC_GET_STATUS_PARAMS_VER` NVFBC_STRUCT_VERSION(NVFBC_GET_STATUS_PARAMS, 1)
NVFBC_GET_STATUS_PARAMS structure version.
- #define `NVFBC_CREATE_CAPTURE_SESSION_PARAMS_VER` NVFBC_STRUCT_VERSION(NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 5)

NVFBC_CREATE_CAPTURE_SESSION_PARAMS structure version.

- #define [NVFBC_DESTROY_CAPTURE_SESSION_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#), 1)
NVFBC_DESTROY_CAPTURE_SESSION_PARAMS structure version.
- #define [NVFBC_BIND_CONTEXT_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_BIND_CONTEXT_PARAMS](#), 1)
NVFBC_BIND_CONTEXT_PARAMS structure version.
- #define [NVFBC_RELEASE_CONTEXT_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_RELEASE_CONTEXT_PARAMS](#), 1)
NVFBC_RELEASE_CONTEXT_PARAMS structure version.
- #define [NVFBC_TOSYS_SETUP_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_TOSYS_SETUP_PARAMS](#), 3)
NVFBC_TOSYS_SETUP_PARAMS structure version.
- #define [NVFBC_TOSYS_GRAB_FRAME_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_TOSYS_GRAB_FRAME_PARAMS](#), 2)
NVFBC_TOSYS_GRAB_FRAME_PARAMS structure version.
- #define [NVFBC_TOCUDA_SETUP_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_TOCUDA_SETUP_PARAMS](#), 1)
NVFBC_TOCUDA_SETUP_PARAMS structure version.
- #define [NVFBC_TOCUDA_GRAB_FRAME_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_TOCUDA_GRAB_FRAME_PARAMS](#), 2)
NVFBC_TOCUDA_GRAB_FRAME_PARAMS structure version.
- #define [NVFBC_TOGL_TEXTURES_MAX](#) 2
Maximum number of GL textures that can be used to store frames.
- #define [NVFBC_TOGL_SETUP_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_TOGL_SETUP_PARAMS](#), 2)
NVFBC_TOGL_SETUP_PARAMS structure version.
- #define [NVFBC_TOGL_GRAB_FRAME_PARAMS_VER](#) NVFBC_STRUCT_VERSION([NVFBC_TOGL_GRAB_FRAME_PARAMS](#), 2)
NVFBC_TOGL_GRAB_FRAME_PARAMS structure version.

Typedefs

- typedef enum [_NVFBCSTATUS](#) NVFBCSTATUS
Defines error codes.
- typedef enum [_NVFBC_BOOL](#) NVFBC_BOOL
Defines boolean values.
- typedef enum [_NVFBC_CAPTURE_TYPE](#) NVFBC_CAPTURE_TYPE

Capture type.

- typedef enum `_NVFBC_BUFFER_FORMAT NVFBC_BUFFER_FORMAT`
Buffer format.
- typedef uint64_t `NVFBC_SESSION_HANDLE`
Handle used to identify an NvFBC session.
- typedef struct `_NVFBC_BOX NVFBC_BOX`
Box used to describe an area of the tracked region to capture.
- typedef struct `_NVFBC_SIZE NVFBC_SIZE`
Size used to describe the size of a frame.
- typedef struct `_NVFBC_FRAME_GRAB_INFO NVFBC_FRAME_GRAB_INFO`
Describes information about a captured frame.
- typedef struct `_NVFBC_CREATE_HANDLE_PARAMS NVFBC_CREATE_HANDLE_PARAMS`
Defines parameters for the `CreateHandle()` API call.
- typedef struct `_NVFBC_DESTROY_HANDLE_PARAMS NVFBC_DESTROY_HANDLE_PARAMS`
Defines parameters for the `NvFBCDestroyHandle()` API call.
- typedef struct `_NVFBC_OUTPUT NVFBC_RANDR_OUTPUT_INFO`
Describes an RandR output.
- typedef struct `_NVFBC_GET_STATUS_PARAMS NVFBC_GET_STATUS_PARAMS`
Defines parameters for the `NvFBCGetStatus()` API call.
- typedef struct `_NVFBC_CREATE_CAPTURE_SESSION_PARAMS NVFBC_CREATE_CAPTURE_SESSION_PARAMS`
Defines parameters for the `NvFBCCreateCaptureSession()` API call.
- typedef struct `_NVFBC_DESTROY_CAPTURE_SESSION_PARAMS NVFBC_DESTROY_CAPTURE_SESSION_PARAMS`
Defines parameters for the `NvFBCDestroyCaptureSession()` API call.
- typedef struct `_NVFBC_BIND_CONTEXT_PARAMS NVFBC_BIND_CONTEXT_PARAMS`
Defines parameters for the `NvFBCBindContext()` API call.
- typedef struct `_NVFBC_RELEASE_CONTEXT_PARAMS NVFBC_RELEASE_CONTEXT_PARAMS`
Defines parameters for the `NvFBCReleaseContext()` API call.
- typedef struct `_NVFBC_TOSYS_SETUP_PARAMS NVFBC_TOSYS_SETUP_PARAMS`
Defines parameters for the `NvFBCToSysSetUp()` API call.
- typedef struct `_NVFBC_TOSYS_GRAB_FRAME_PARAMS NVFBC_TOSYS_GRAB_FRAME_PARAMS`
Defines parameters for the `NvFBCToSysGrabFrame()` API call.
- typedef struct `_NVFBC_TOCUDA_SETUP_PARAMS NVFBC_TOCUDA_SETUP_PARAMS`

Defines parameters for the `NvFBCToCudaSetUp()` API call.

- typedef struct `_NVFBC_TOCUDA_GRAB_FRAME_PARAMS` `NVFBC_TOCUDA_GRAB_FRAME_PARAMS`

Defines parameters for the `NvFBCToCudaGrabFrame()` API call.

- typedef struct `_NVFBC_TOGL_SETUP_PARAMS` `NVFBC_TOGL_SETUP_PARAMS`

Defines parameters for the `NvFBCToGLSetUp()` API call.

- typedef struct `_NVFBC_TOGL_GRAB_FRAME_PARAMS` `NVFBC_TOGL_GRAB_FRAME_PARAMS`

Defines parameters for the `NvFBCToGLGrabFrame()` API call.

Enumerations

- enum `_NVFBCSTATUS` {
`NVFBC_SUCCESS` = 0, `NVFBC_ERR_API_VERSION` = 1, `NVFBC_ERR_INTERNAL` = 2, `NVFBC_ERR_INVALID_PARAM` = 3,
`NVFBC_ERR_INVALID_PTR` = 4, `NVFBC_ERR_INVALID_HANDLE` = 5, `NVFBC_ERR_MAX_CLIENTS` = 6, `NVFBC_ERR_UNSUPPORTED` = 7,
`NVFBC_ERR_OUT_OF_MEMORY` = 8, `NVFBC_ERR_BAD_REQUEST` = 9, `NVFBC_ERR_X` = 10, `NVFBC_ERR_GLX` = 11,
`NVFBC_ERR_GL` = 12, `NVFBC_ERR_CUDA` = 13, `NVFBC_ERR_ENCODER` = 14, `NVFBC_ERR_CONTEXT` = 15,
`NVFBC_ERR_MUST_RECREATE` = 16 }

Defines error codes.

- enum `_NVFBC_BOOL` { `NVFBC_FALSE` = 0, `NVFBC_TRUE` }

Defines boolean values.

- enum `_NVFBC_CAPTURE_TYPE` { `NVFBC_CAPTURE_TO_SYS` = 0, `NVFBC_CAPTURE_SHARED_CUDA`, `NVFBC_CAPTURE_TO_HW_ENCODER`, `NVFBC_CAPTURE_TO_GL` }

Capture type.

- enum `NVFBC_TRACKING_TYPE` { `NVFBC_TRACKING_DEFAULT` = 0, `NVFBC_TRACKING_OUTPUT`, `NVFBC_TRACKING_SCREEN` }

Tracking type.

- enum `_NVFBC_BUFFER_FORMAT` {
`NVFBC_BUFFER_FORMAT_ARGB` = 0, `NVFBC_BUFFER_FORMAT_RGB`, `NVFBC_BUFFER_FORMAT_NV12`, `NVFBC_BUFFER_FORMAT_YUV444P`,
`NVFBC_BUFFER_FORMAT_RGBA`, `NVFBC_BUFFER_FORMAT_BGRA` }

Buffer format.

- enum `NVFBC_TOSYS_GRAB_FLAGS` { `NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS` = 0, `NVFBC_TOSYS_GRAB_FLAGS_NOWAIT` = (1 << 0), `NVFBC_TOSYS_GRAB_FLAGS_FORCE_REFRESH` = (1 << 1), `NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY` = (1 << 2) }

Defines flags that can be used when capturing to system memory.

- enum `NVFBC_TOCUDA_FLAGS` { `NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS` = 0, `NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT` = (1 << 0), `NVFBC_TOCUDA_GRAB_FLAGS_FORCE_REFRESH` = (1 << 1), `NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY` = (1 << 2) }

Defines flags that can be used when capturing to a CUDA buffer in video memory.

- enum `NVFBC_TOGL_FLAGS` { `NVFBC_TOGL_GRAB_FLAGS_NOFLAGS` = 0, `NVFBC_TOGL_GRAB_FLAGS_NOWAIT` = (1 << 0), `NVFBC_TOGL_GRAB_FLAGS_FORCE_REFRESH` = (1 << 1), `NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY` = (1 << 2) }

Defines flags that can be used when capturing to an OpenGL buffer in video memory.

7.3.1 Typedef Documentation

7.3.1.1 typedef struct `_NVFBC_BOX` `NVFBC_BOX`

Box used to describe an area of the tracked region to capture.

The coordinates are relative to the tracked region.

E.g., if the size of the X screen is 3520x1200 and the tracked RandR output scans a region of 1600x1200+1920+0, then setting a capture box of 800x600+100+50 effectively captures a region of 800x600+2020+50 relative to the X screen.

7.3.1.2 typedef struct `_NVFBC_OUTPUT` `NVFBC_RANDR_OUTPUT_INFO`

Describes an RandR output.

Filling this structure relies on the XRandR extension. This feature cannot be used if the extension is missing or its version is below the requirements.

See also:

Requirements

7.3.1.3 typedef enum `_NVFBCSTATUS` `NVFBCSTATUS`

Defines error codes.

See also:

[NvFBCGetLastErrorStr](#)

7.3.2 Enumeration Type Documentation

7.3.2.1 enum `_NVFBC_BOOL`

Defines boolean values.

Enumerator:

`NVFBC_FALSE` False value.

`NVFBC_TRUE` True value.

7.3.2.2 enum _NVFBC_BUFFER_FORMAT

Buffer format.

Enumerator:

- NVFBC_BUFFER_FORMAT_ARGB*** Data will be converted to ARGB8888 byte-order format.
32 bpp.
- NVFBC_BUFFER_FORMAT_RGB*** Data will be converted to RGB888 byte-order format.
24 bpp.
- NVFBC_BUFFER_FORMAT_NV12*** Data will be converted to NV12 format using HDTV weights according to ITU-R BT.709.
12 bpp.
- NVFBC_BUFFER_FORMAT_YUV444P*** Data will be converted to YUV 444 planar format using HDTV weights according to ITU-R BT.709.
24 bpp
- NVFBC_BUFFER_FORMAT_RGBA*** Data will be converted to RGBA8888 byte-order format.
32 bpp.
- NVFBC_BUFFER_FORMAT_BGRA*** Data will be converted to BGRA8888 byte-order format.
32 bpp.

7.3.2.3 enum _NVFBC_CAPTURE_TYPE

Capture type.

Enumerator:

- NVFBC_CAPTURE_TO_SYS*** Capture frames to a buffer in system memory.
- NVFBC_CAPTURE_SHARED_CUDA*** Capture frames to a CUDA device in video memory.
Specifying this will dlopen() libcuda.so.1 and fail if not available.
Deprecated
- NVFBC_CAPTURE_TO_HW_ENCODER*** Capture HW compressed frames to a buffer in system memory.
Using the HW encoder relies on the CUDA interop. Therefore, CUDA must be installed on the system.
- NVFBC_CAPTURE_TO_GL*** Capture frames to an OpenGL buffer in video memory.

7.3.2.4 enum _NVFBCSTATUS

Defines error codes.

See also:

[NvFBCGetLastErrorStr](#)

Enumerator:

- NVFBC_SUCCESS*** This indicates that the API call returned with no errors.
- NVFBC_ERR_API_VERSION*** This indicates that the API version between the client and the library is not compatible.

NVFBC_ERR_INTERNAL An internal error occurred.

NVFBC_ERR_INVALID_PARAM This indicates that one or more of the parameter passed to the API call is invalid.

NVFBC_ERR_INVALID_PTR This indicates that one or more of the pointers passed to the API call is invalid.

NVFBC_ERR_INVALID_HANDLE This indicates that the handle passed to the API call to identify the client is invalid.

NVFBC_ERR_MAX_CLIENTS This indicates that the maximum number of threaded clients of the same process has been reached.

The limit is 10 threads per process. There is no limit on the number of process.

NVFBC_ERR_UNSUPPORTED This indicates that the requested feature is not currently supported by the library.

NVFBC_ERR_OUT_OF_MEMORY This indicates that the API call failed because it was unable to allocate enough memory to perform the requested operation.

NVFBC_ERR_BAD_REQUEST This indicates that the API call was not expected.

This happens when API calls are performed in a wrong order, such as trying to capture a frame prior to creating a new capture session; or trying to set up a capture to video memory although a capture session to system memory was created.

NVFBC_ERR_X This indicates an X error, most likely meaning that the X server has been terminated.

When this error is returned, the only resort is to create another FBC handle using [NvFBCCreateHandle\(\)](#).

The previous handle should still be freed with [NvFBCDestroyHandle\(\)](#), but it might leak resources, in particular X, GLX, and GL resources since it is no longer possible to communicate with an X server to free them through the driver.

The best course of action to eliminate this potential leak is to close the OpenGL driver, close the forked process running the capture, or restart the application.

NVFBC_ERR_GLX This indicates a GLX error.

NVFBC_ERR_GL This indicates an OpenGL error.

NVFBC_ERR_CUDA This indicates a CUDA error.

NVFBC_ERR_ENCODER This indicates a HW encoder error.

NVFBC_ERR_CONTEXT This indicates an NvFBC context error.

NVFBC_ERR_MUST_RECREATE This indicates that the application must recreate the capture session.

This error can be returned if a modeset event occurred while capturing frames, and `NVFBC_CREATE_HANDLE_PARAMS::bDisableAutoModesetRecovery` was set to `NVFBC_TRUE`.

7.3.2.5 enum NVFBC_TOCUDA_FLAGS

Defines flags that can be used when capturing to a CUDA buffer in video memory.

Enumerator:

NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS Default, capturing waits for a new frame or mouse move.

The default behavior of blocking grabs is to wait for a new frame until after the call was made. But it's possible that there is a frame already ready that the client hasn't seen.

See also:

[NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY](#)

NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT Capturing does not wait for a new frame nor a mouse move.

It is therefore possible to capture the same frame multiple times. When this occurs, the `dwCurrentFrame` parameter of the `NVFBC_FRAME_GRAB_INFO` structure is not incremented.

NVFBC_TOCUDA_GRAB_FLAGS_FORCE_REFRESH [in] Forces the destination buffer to be refreshed even if the frame has not changed since previous capture.

By default, if the captured frame is identical to the previous one, NvFBC will omit one copy and not update the destination buffer.

Setting that flag will prevent this behavior. This can be useful e.g., if the application has modified the buffer in the meantime.

NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY Similar to `NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS`, except that the capture will not wait if there is already a frame available that the client has never seen yet.

7.3.2.6 enum NVFBC_TOGL_FLAGS

Defines flags that can be used when capturing to an OpenGL buffer in video memory.

Enumerator:

NVFBC_TOGL_GRAB_FLAGS_NOFLAGS Default, capturing waits for a new frame or mouse move.

The default behavior of blocking grabs is to wait for a new frame until after the call was made. But it's possible that there is a frame already ready that the client hasn't seen.

See also:

[NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY](#)

NVFBC_TOGL_GRAB_FLAGS_NOWAIT Capturing does not wait for a new frame nor a mouse move.

It is therefore possible to capture the same frame multiple times. When this occurs, the `dwCurrentFrame` parameter of the `NVFBC_FRAME_GRAB_INFO` structure is not incremented.

NVFBC_TOGL_GRAB_FLAGS_FORCE_REFRESH [in] Forces the destination buffer to be refreshed even if the frame has not changed since previous capture.

By default, if the captured frame is identical to the previous one, NvFBC will omit one copy and not update the destination buffer.

Setting that flag will prevent this behavior. This can be useful e.g., if the application has modified the buffer in the meantime.

NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY Similar to `NVFBC_TOGL_GRAB_FLAGS_NOFLAGS`, except that the capture will not wait if there is already a frame available that the client has never seen yet.

7.3.2.7 enum NVFBC_TOSYS_GRAB_FLAGS

Defines flags that can be used when capturing to system memory.

Enumerator:

NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS Default, capturing waits for a new frame or mouse move.

The default behavior of blocking grabs is to wait for a new frame until after the call was made. But it's possible that there is a frame already ready that the client hasn't seen.

See also:

[NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY](#)

NVFBC_TOSYS_GRAB_FLAGS_NOWAIT Capturing does not wait for a new frame nor a mouse move.

It is therefore possible to capture the same frame multiple times. When this occurs, the `dwCurrentFrame` parameter of the `NVFBC_FRAME_GRAB_INFO` structure is not incremented.

NVFBC_TOSYS_GRAB_FLAGS_FORCE_REFRESH Forces the destination buffer to be refreshed even if the frame has not changed since previous capture.

By default, if the captured frame is identical to the previous one, NvFBC will omit one copy and not update the destination buffer.

Setting that flag will prevent this behavior. This can be useful e.g., if the application has modified the buffer in the meantime.

NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY Similar to `NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS`, except that the capture will not wait if there is already a frame available that the client has never seen yet.

7.3.2.8 enum NVFBC_TRACKING_TYPE

Tracking type.

NvFBC can track a specific region of the framebuffer to capture.

An X screen corresponds to the entire framebuffer.

An RandR CRTC is a component of the GPU that reads pixels from a region of the X screen and sends them through a pipeline to an RandR output. A physical monitor can be connected to an RandR output. Tracking an RandR output captures the region of the X screen that the RandR CRTC is sending to the RandR output.

Enumerator:

NVFBC_TRACKING_DEFAULT By default, NvFBC tries to track a connected primary output.

If none is found, then it tries to track the first connected output. If none is found then it tracks the entire X screen.

If the XRandR extension is not available, this option has the same effect as `NVFBC_TRACKING_SCREEN`.

This default behavior might be subject to changes in the future.

NVFBC_TRACKING_OUTPUT Track an RandR output specified by its ID in the appropriate field.

The list of connected outputs can be queried via `NvFBCGetStatus()`. This list can also be obtained using e.g., `xrandr(1)`.

If the XRandR extension is not available, setting this option returns an error.

NVFBC_TRACKING_SCREEN Track the entire X screen.

7.4 Deprecated Structure Definition

These definitions are deprecated and should not be used anymore.

Classes

- struct [_NVFBC_HWENC_CONFIG](#)
- struct [_NVFBC_HWENC_ENCODE_PARAMS](#)
- struct [_NVFBC_HWENC_FRAME_INFO](#)
- struct [_NVFBC_TOHWENC_GET_CAPS_PARAMS](#)
- struct [_NVFBC_TOHWENC_SETUP_PARAMS](#)
- struct [_NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#)
- struct [_NVFBC_TOHWENC_GET_HEADER_PARAMS](#)

Defines

- #define [NVFBC_MAX_REF_FRAMES](#) 0x10
- #define [NVFBC_HWENC_CONFIG_VER](#) NVFBC_STRUCT_VERSION(NVFBC_HWENC_CONFIG, 5)
- #define [NVFBC_HWENC_ENCODE_PARAMS_VER](#) NVFBC_STRUCT_VERSION(NVFBC_HWENC_ENCODE_PARAMS, 1)
- #define [NVFBC_HWENC_FRAME_INFO_VER](#) NVFBC_STRUCT_VERSION(NVFBC_HWENC_FRAME_INFO, 1)
- #define [NVFBC_TOHWENC_GET_CAPS_PARAMS_VER](#) NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GET_CAPS_PARAMS, 1)
- #define [NVFBC_TOHWENC_SETUP_PARAMS_VER](#) NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_SETUP_PARAMS, 1)
- #define [NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER](#) NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 2)
- #define [NVFBC_TOHWENC_GET_HEADER_PARAMS_VER](#) NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GET_HEADER_PARAMS, 1)
- #define [NVFBC_CAPTURE_TO_H264_HW_ENCODER](#) NVFBC_CAPTURE_TO_HW_ENCODER
- #define [NVFBC_TOH264_GRAB_FLAGS_NOFLAGS](#) NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS
- #define [NVFBC_TOH264_GRAB_FLAGS_NOWAIT](#) NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT
- #define [NVFBC_TOH264_GRAB_FLAGS](#) [NVFBC_TOHWENC_GRAB_FLAGS](#)
- #define [NVFBC_H264_PRESET_LOW_LATENCY_HP](#) NVFBC_HWENC_PRESET_LOW_LATENCY_HP
- #define [NVFBC_H264_PRESET_LOW_LATENCY_HQ](#) NVFBC_HWENC_PRESET_LOW_LATENCY_HQ
- #define [NVFBC_H264_PRESET_LOW_LATENCY_DEFAULT](#) NVFBC_HWENC_PRESET_LOW_LATENCY_DEFAULT
- #define [NVFBC_H264_PRESET_LOSSLESS_HP](#) NVFBC_HWENC_PRESET_LOSSLESS_HP
- #define [NVFBC_H264_PRESET](#) [NVFBC_HWENC_PRESET](#)
- #define [NVFBC_H264_ENC_PARAMS_RC_CONSTQP](#) NVFBC_HWENC_PARAMS_RC_CONSTQP
- #define [NVFBC_H264_ENC_PARAMS_RC_VBR](#) NVFBC_HWENC_PARAMS_RC_VBR
- #define [NVFBC_H264_ENC_PARAMS_RC_CBR](#) NVFBC_HWENC_PARAMS_RC_CBR
- #define [NVFBC_H264_ENC_PARAMS_RC_2_PASS_QUALITY](#) NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY
- #define [NVFBC_H264_ENC_PARAMS_RC_2_PASS_FRAME_SIZE_CAP](#) NVFBC_HWENC_PARAMS_RC_2_PASS_FRAME_SIZE_CAP

- `#define NVFBC_H264_RATE_CONTROL_CBR_IFRAME_2_PASS NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_PASS`
- `#define NVFBC_H264_ENC_PARAMS_RC_MODE NVFBC_HWENC_PARAMS_RC_MODE`
- `#define NVFBC_H264_ENC_PARAM_FLAG_FORCEIDR NVFBC_HWENC_PARAM_FLAG_FORCEIDR`
- `#define NVFBC_H264_ENC_PARAM_FLAG_DYN_BITRATE_CHANGE NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE`
- `#define NVFBC_H264_ENC_PARAM_FLAGS NVFBC_HWENC_PARAM_FLAGS`
- `#define NVFBC_H264_ENC_SLICING_MODE_DISABLED NVFBC_HWENC_SLICING_MODE_DISABLED`
- `#define NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_MBS NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MBS`
- `#define NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_BYTES NVFBC_HWENC_SLICING_MODE_FIXED_NUM_BYTES`
- `#define NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_MB_ROWS NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MB_ROWS`
- `#define NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_SLICES NVFBC_HWENC_SLICING_MODE_FIXED_NUM_SLICES`
- `#define NVFBC_H264_ENC_SLICING_MODE NVFBC_HWENC_SLICING_MODE`
- `#define NVFBC_H264_HW_ENC_CONFIG NVFBC_HWENC_CONFIG`
- `#define NVFBC_H264_HW_ENC_CONFIG_VER NVFBC_STRUCT_VERSION(NVFBC_H264_HW_ENC_CONFIG, 4)`
- `#define NVFBC_H264_HW_ENC_ENCODE_PARAMS NVFBC_HWENC_ENCODE_PARAMS`
- `#define NVFBC_H264_HW_ENC_ENCODE_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_H264_HW_ENC_ENCODE_PARAMS, 1)`
- `#define NVFBC_H264_HW_ENC_FRAME_INFO NVFBC_HWENC_FRAME_INFO`
- `#define NVFBC_H264_HW_ENC_FRAME_INFO_VER NVFBC_STRUCT_VERSION(NVFBC_H264_HW_ENC_FRAME_INFO, 1)`
- `#define NVFBC_TOH264_SETUP_PARAMS NVFBC_TOHWENC_SETUP_PARAMS`
- `#define NVFBC_TOH264_SETUP_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOH264_SETUP_PARAMS, 1)`
- `#define NVFBC_TOH264_GRAB_FRAME_PARAMS NVFBC_TOHWENC_GRAB_FRAME_PARAMS`
- `#define NVFBC_TOH264_GRAB_FRAME_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOH264_GRAB_FRAME_PARAMS, 2)`
- `#define NVFBC_TOH264_GET_HEADER_PARAMS NVFBC_TOHWENC_GET_HEADER_PARAMS`
- `#define NVFBC_TOH264_GET_HEADER_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOH264_GET_HEADER_PARAMS, 1)`
- `#define NVFBC_BUFFER_FORMAT_YUV420P NVFBC_BUFFER_FORMAT_NV12`

Typedefs

- `typedef enum _NVFBC_HWENC_PARAMS_RC_MODE NVFBC_HWENC_PARAMS_RC_MODE`
- `typedef struct _NVFBC_HWENC_CONFIG NVFBC_HWENC_CONFIG`
- `typedef struct _NVFBC_HWENC_ENCODE_PARAMS NVFBC_HWENC_ENCODE_PARAMS`
- `typedef struct _NVFBC_HWENC_FRAME_INFO NVFBC_HWENC_FRAME_INFO`
- `typedef struct _NVFBC_TOHWENC_GET_CAPS_PARAMS NVFBC_TOHWENC_GET_CAPS_PARAMS`
- `typedef struct _NVFBC_TOHWENC_SETUP_PARAMS NVFBC_TOHWENC_SETUP_PARAMS`
- `typedef struct _NVFBC_TOHWENC_GRAB_FRAME_PARAMS NVFBC_TOHWENC_GRAB_FRAME_PARAMS`
- `typedef struct _NVFBC_TOHWENC_GET_HEADER_PARAMS NVFBC_TOHWENC_GET_HEADER_PARAMS`

Enumerations

- enum NVFBC_TOHWENC_GRAB_FLAGS { NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS = 0, NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT = (1 << 0) }
- enum NVFBC_HWENC_PRESET { NVFBC_HWENC_PRESET_LOW_LATENCY_HP = 0, NVFBC_HWENC_PRESET_LOW_LATENCY_HQ, NVFBC_HWENC_PRESET_LOW_LATENCY_DEFAULT, NVFBC_HWENC_PRESET_LOSSLESS_HP }
- enum _NVFBC_HWENC_PARAMS_RC_MODE { NVFBC_HWENC_PARAMS_RC_CONSTQP = 0, NVFBC_HWENC_PARAMS_RC_VBR, NVFBC_HWENC_PARAMS_RC_CBR, NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY, NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP, NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_PASS }
- enum NVFBC_HWENC_PARAM_FLAGS { NVFBC_HWENC_PARAM_FLAG_FORCEIDR = (1 << 0), NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE = (1 << 1) }
- enum NVFBC_HWENC_SLICING_MODE { NVFBC_HWENC_SLICING_MODE_DISABLED = 0, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MBS, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_BYTES, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MB_ROWS, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_SLICES }
- enum NVFBC_HWENC_CODEC { NVFBC_HWENC_CODEC_H264 = 0, NVFBC_HWENC_CODEC_HEVC }

7.4.1 Detailed Description

These definitions are deprecated and should not be used anymore.

However, NvFBC ensures backward compatibility.

Deprecated structures point to the new ones at a specific structure version.

7.4.2 Define Documentation

7.4.2.1 #define NVFBC_HWENC_CONFIG_VER NVFBC_STRUCT_VERSION(NVFBC_HWENC_CONFIG, 5)

Deprecated

NVFBC_HWENC_CONFIG structure version.

7.4.2.2 #define NVFBC_HWENC_ENCODE_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_HWENC_ENCODE_PARAMS, 1)

Deprecated

NVFBC_HWENC_ENCODE_PARAMS structure version.

7.4.2.3 #define NVFBC_HWENC_FRAME_INFO_VER NVFBC_STRUCT_VERSION(NVFBC_HWENC_FRAME_INFO, 1)

Deprecated

NVFBC_HWENC_FRAME_INFO structure version.

7.4.2.4 #define NVFBC_MAX_REF_FRAMES 0x10**Deprecated**

Maximum number of reference frames.

7.4.2.5 #define NVFBC_TOHWENC_GET_CAPS_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GET_CAPS_PARAMS, 1)**Deprecated**

NVFBC_TOHWENC_GET_CAPS_PARAMS structure version.

7.4.2.6 #define NVFBC_TOHWENC_GET_HEADER_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GET_HEADER_PARAMS, 1)**Deprecated**

NVFBC_TOHWENC_GET_HEADER_PARAMS structure version.

7.4.2.7 #define NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 2)**Deprecated**

NVFBC_TOHWENC_GRAB_FRAME_PARAMS structure version.

7.4.2.8 #define NVFBC_TOHWENC_SETUP_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_SETUP_PARAMS, 1)**Deprecated**

NVFBC_TOHWENC_SETUP_PARAMS structure version.

7.4.3 Typedef Documentation**7.4.3.1 typedef struct _NVFBC_HWENC_CONFIG NVFBC_HWENC_CONFIG****Deprecated**

Describes HW encoder configuration.

7.4.3.2 typedef struct _NVFBC_HWENC_ENCODE_PARAMS NVFBC_HWENC_ENCODE_PARAMS**Deprecated**

Describes encode parameters.

7.4.3.3 `typedef struct _NVFBC_HWENC_FRAME_INFO NVFBC_HWENC_FRAME_INFO`

Deprecated

Describes an encoded frame.

7.4.3.4 `typedef enum _NVFBC_HWENC_PARAMS_RC_MODE NVFBC_HWENC_PARAMS_RC_MODE`

Deprecated

Defines encoder rate control modes.

7.4.3.5 `typedef struct _NVFBC_TOHWENC_GET_CAPS_PARAMS NVFBC_TOHWENC_GET_CAPS_PARAMS`

Deprecated

Defines parameters for the `ToHwGetCaps()` API call.

7.4.3.6 `typedef struct _NVFBC_TOHWENC_GET_HEADER_PARAMS NVFBC_TOHWENC_GET_HEADER_PARAMS`

Deprecated

Defines parameters for the [NvFBCToHwEncGetHeader\(\)](#) API call.

7.4.3.7 `typedef struct _NVFBC_TOHWENC_GRAB_FRAME_PARAMS NVFBC_TOHWENC_GRAB_FRAME_PARAMS`

Deprecated

Defines parameters for the [NvFBCToHwEncGrabFrame\(\)](#) API call.

7.4.3.8 `typedef struct _NVFBC_TOHWENC_SETUP_PARAMS NVFBC_TOHWENC_SETUP_PARAMS`

Deprecated

Defines parameters for the `ToHwEncSetUp()` API call.

7.4.4 Enumeration Type Documentation

7.4.4.1 `enum _NVFBC_HWENC_PARAMS_RC_MODE`

Deprecated

Defines encoder rate control modes.

Enumerator:

`NVFBC_HWENC_PARAMS_RC_CONSTQP` Constant QP mode.

NVFBC_HWENC_PARAMS_RC_VBR Variable bitrate mode.

NVFBC_HWENC_PARAMS_RC_CBR Constant bitrate mode.

NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY Multi pass encoding optimized for image quality and works best with single frame VBV buffer size.

NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP Multi pass encoding optimized for maintaining frame size and works best with single frame VBV buffer size.

NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_PASS Deprecated.

Use *NVFBC_HWENC_PARAMS_RC_CBR* instead.

7.4.4.2 enum NVFBC_HWENC_CODEC

Deprecated

Defines video codecs.

Enumerator:

NVFBC_HWENC_CODEC_H264 H.264 codec.

NVFBC_HWENC_CODEC_HEVC H.265/HEVC codec.

7.4.4.3 enum NVFBC_HWENC_PARAM_FLAGS

Deprecated

Defines encoder flags.

Enumerator:

NVFBC_HWENC_PARAM_FLAG_FORCEIDR Encodes the current frame as an IDR picture.

NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE Indicates change in bitrate from current frame onwards.

7.4.4.4 enum NVFBC_HWENC_PRESET

Deprecated

Defines encoder presets.

Enumerator:

NVFBC_HWENC_PRESET_LOW_LATENCY_HP Use for fastest encoding, with suboptimal quality.

NVFBC_HWENC_PRESET_LOW_LATENCY_HQ Use for better overall quality, compromising on encoding speed.

NVFBC_HWENC_PRESET_LOW_LATENCY_DEFAULT Use for better quality than *NVFBC_HWENC_PRESET_LOW_LATENCY_HP* and higher speed than *NVFBC_HWENC_PRESET_LOW_LATENCY_HQ*.

NVFBC_HWENC_PRESET_LOSSLESS_HP Use for lossless encoding at higher performance.

Currently supported only when [NVFBC_HWENC_CONFIG::eRateControl](#) is set to *NVFBC_HWENC_PARAMS_RC_CONSTQP*. If this preset is used, [NVFBC_HWENC_CONFIG::dwProfile](#) is forced to 244. Available on Maxwell GPUs onwards and only with the H.264 codec.

7.4.4.5 enum NVFBC_HWENC_SLICING_MODE

Deprecated

Defines slicing modes.

Enumerator:

NVFBC_HWENC_SLICING_MODE_DISABLED Disable slicing mode.

NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MBS Picture will be divided into slices of n MBs, where n = dwSlicingModeParam.

NVFBC_HWENC_SLICING_MODE_FIXED_NUM_BYTES Picture will be divided into slices of n Bytes, where n = dwSlicingModeParam.

NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MB_ROWS Picture will be divided into slices of n rows of MBs, where n = dwSlicingModeParam.

NVFBC_HWENC_SLICING_MODE_FIXED_NUM_SLICES Picture will be divided into n+1 slices, where n = dwSlicingModeParam.

7.4.4.6 enum NVFBC_TOHWENC_GRAB_FLAGS

Deprecated

Defines flags that can be used when capturing HW encoded compressed frames to system memory.

Enumerator:

NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS Default, capturing waits for a new frame or mouse move.

NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT Capturing does not wait for a new frame nor a mouse move.

It is therefore possible to capture the same frame multiple times. When this occurs, the dwCurrentFrame parameter of the NVFBC_FRAME_GRAB_INFO structure is not incremented.

7.5 API Entry Points

Entry points are thread-safe and can be called concurrently.

Typedefs

- typedef `NVFBCSTATUS(NVFBCAPI * PNVFBCCREATEINSTANCE)(NVFBC_API_FUNCTION_LIST *pFunctionList)`
Defines function pointer for the `NvFBCCreateInstance()` API call.

Functions

- `const char *NVFBCAPI NvFBCGetLastErrorStr (const NVFBC_SESSION_HANDLE sessionHandle)`
Gets the last error message that got recorded for a client.
- `NVFBCSTATUS NVFBCAPI NvFBCCreateHandle (NVFBC_SESSION_HANDLE *pSessionHandle, NVFBC_CREATE_HANDLE_PARAMS *pParams)`
Allocates a new handle for an NvFBC client.
- `NVFBCSTATUS NVFBCAPI NvFBCDestroyHandle (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_DESTROY_HANDLE_PARAMS *pParams)`
Destroys the handle of an NvFBC client.
- `NVFBCSTATUS NVFBCAPI NvFBCGetStatus (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_GET_STATUS_PARAMS *pParams)`
Gets the current status of the display driver.
- `NVFBCSTATUS NVFBCAPI NvFBCBindContext (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_BIND_CONTEXT_PARAMS *pParams)`
Binds the FBC context to the calling thread.
- `NVFBCSTATUS NVFBCAPI NvFBCReleaseContext (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_RELEASE_CONTEXT_PARAMS *pParams)`
Releases the FBC context from the calling thread.
- `NVFBCSTATUS NVFBCAPI NvFBCCreateCaptureSession (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_CREATE_CAPTURE_SESSION_PARAMS *pParams)`
Creates a capture session for an FBC client.
- `NVFBCSTATUS NVFBCAPI NvFBCDestroyCaptureSession (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_DESTROY_CAPTURE_SESSION_PARAMS *pParams)`
Destroys a capture session for an FBC client.
- `NVFBCSTATUS NVFBCAPI NvFBCToSysSetUp (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_TOSYS_SETUP_PARAMS *pParams)`
Sets up a capture to system memory session.
- `NVFBCSTATUS NVFBCAPI NvFBCToSysGrabFrame (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_TOSYS_GRAB_FRAME_PARAMS *pParams)`

Captures a frame to a buffer in system memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToCudaSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOCUDA_SETUP_PARAMS](#) *pParams)

Sets up a capture to video memory session.

- [NVFBCSTATUS NVFBCAPI NvFBCToCudaGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOCUDA_GRAB_FRAME_PARAMS](#) *pParams)

Captures a frame to a CUDA device in video memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToGLSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOGL_SETUP_PARAMS](#) *pParams)

Sets up a capture to OpenGL buffer in video memory session.

- [NVFBCSTATUS NVFBCAPI NvFBCToGLGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOGL_GRAB_FRAME_PARAMS](#) *pParams)

Captures a frame to an OpenGL buffer in video memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToH264SetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOH264_SETUP_PARAMS](#) *pParams)

Sets up a capture to H.264 compressed frames in system memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToH264GrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOH264_GRAB_FRAME_PARAMS](#) *pParams)

Captures a H.264 compressed frame to a bitstream in system memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToH264GetHeader](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOH264_GET_HEADER_PARAMS](#) *pParams)

Gets SPS/PPS headers.

- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncGetCaps](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_GET_CAPS_PARAMS](#) *pParams)

- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_SETUP_PARAMS](#) *pParams)

- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#) *pParams)

- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncGetHeader](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_GET_HEADER_PARAMS](#) *pParams)

- [NVFBCSTATUS NVFBCAPI NvFBCCreateInstance](#) ([NVFBC_API_FUNCTION_LIST](#) *pFunctionList)

Entry Points to the NvFBC interface.

7.5.1 Detailed Description

Entry points are thread-safe and can be called concurrently.

The locking model includes a global lock that protects session handle management (

See also:

[NvFBCCreateHandle](#),
[NvFBCDestroyHandle](#)).

Each NvFBC session uses a local lock to protect other entry points. Note that in certain cases, a thread can hold the local lock for an undefined amount of time, such as grabbing a frame using a blocking call.

Note that a context is associated with each session. NvFBC clients wishing to share a session between different threads are expected to release and bind the context appropriately (

See also:

[NvFBCBindContext](#),
[NvFBCReleaseContext](#)). This is not required when each thread uses its own NvFBC session.

7.5.2 Function Documentation

7.5.2.1 NVFBCSTATUS NVFBCAPI NvFBCBindContext (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_BIND_CONTEXT_PARAMS * *pParams*)

Binds the FBC context to the calling thread.

The NvFBC library internally relies on objects that must be bound to a thread. Such objects are OpenGL contexts and CUDA contexts.

This function binds these objects to the calling thread.

The FBC context must be bound to the calling thread for most NvFBC entry points, otherwise [NVFBC_ERR_CONTEXT](#) is returned.

If the FBC context is already bound to a different thread, [NVFBC_ERR_CONTEXT](#) is returned. The other thread must release the context first by calling the [ReleaseContext\(\)](#) entry point.

If the FBC context is already bound to the current thread, this function has no effects.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* [NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#)

Returns:

[NVFBC_SUCCESS](#)
[NVFBC_ERR_INVALID_HANDLE](#)
[NVFBC_ERR_API_VERSION](#)
[NVFBC_ERR_BAD_REQUEST](#)
[NVFBC_ERR_CONTEXT](#)
[NVFBC_ERR_INTERNAL](#)
[NVFBC_ERR_X](#)

7.5.2.2 NVFBCSTATUS NVFBCAPI NvFBCCreateCaptureSession (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_CREATE_CAPTURE_SESSION_PARAMS * *pParams*)

Creates a capture session for an FBC client.

This function starts a capture session of the desired type (system memory, video memory with CUDA interop, or H.264 compressed frames in system memory).

Not all types are supported on all systems. Also, it is possible to use NvFBC without having the CUDA library or HW encoder library. In this case, requesting a capture session of the concerned type will return an error.

After this function returns, the display driver will start generating frames that can be captured using the corresponding API call.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_CREATE_CAPTURE_SESSION_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PARAM
 NVFBC_ERR_OUT_OF_MEMORY
 NVFBC_ERR_X
 NVFBC_ERR_GLX
 NVFBC_ERR_GL
 NVFBC_ERR_CUDA
 NVFBC_ERR_ENCODER
 NVFBC_ERR_INTERNAL

7.5.2.3 NVFBCSTATUS NVFBCAPI NvFBCCreateHandle (NVFBC_SESSION_HANDLE * *pSessionHandle*, NVFBC_CREATE_HANDLE_PARAMS **pParams*)

Allocates a new handle for an NvFBC client.

This function allocates a session handle used to identify an FBC client.

This function implicitly calls [NvFBCBindContext\(\)](#).

Parameters:

- *pSessionHandle* Pointer that will hold the allocated session handle.
- ← *pParams* NVFBC_CREATE_HANDLE_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_PTR
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_OUT_OF_MEMORY
 NVFBC_ERR_MAX_CLIENTS
 NVFBC_ERR_X
 NVFBC_ERR_GLX
 NVFBC_ERR_GL

7.5.2.4 NVFBCSTATUS NVFBCAPI NvFBCCreateInstance (NVFBC_API_FUNCTION_LIST * *pFunctionList*)

Entry Points to the NvFBC interface.

Creates an instance of the NvFBC interface, and populates the pFunctionList with function pointers to the API routines implemented by the NvFBC interface.

Parameters:

→ *pFunctionList*

Returns:

NVFBC_SUCCESS
NVFBC_ERR_INVALID_PTR
NVFBC_ERR_API_VERSION

7.5.2.5 NVFBCSTATUS NVFBCAPI NvFBCDestroyCaptureSession (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_DESTROY_CAPTURE_SESSION_PARAMS *pParams)

Destroys a capture session for an FBC client.

This function stops a capture session and frees allocated objects.

After this function returns, it is possible to create another capture session using the corresponding API call.

Parameters:

← *sessionHandle* FBC session handle.

← *pParams* NVFBC_DESTROY_CAPTURE_SESSION_PARAMS

Returns:

NVFBC_SUCCESS
NVFBC_ERR_INVALID_HANDLE
NVFBC_ERR_API_VERSION
NVFBC_ERR_BAD_REQUEST
NVFBC_ERR_CONTEXT
NVFBC_ERR_INTERNAL
NVFBC_ERR_X

7.5.2.6 NVFBCSTATUS NVFBCAPI NvFBCDestroyHandle (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_DESTROY_HANDLE_PARAMS *pParams)

Destroys the handle of an NvFBC client.

This function uninitialized an FBC client.

This function implicitly calls [NvFBCReleaseContext\(\)](#).

After this function returns, it is not possible to use this session handle for any further API call.

Parameters:

← *sessionHandle* FBC session handle.

← *pParams* NVFBC_DESTROY_HANDLE_PARAMS

Returns:

NVFBC_SUCCESS
NVFBC_ERR_INVALID_HANDLE
NVFBC_ERR_API_VERSION
NVFBC_ERR_BAD_REQUEST

NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_X

7.5.2.7 **const char* NVFBCAPI NvFBCGetLastErrorStr (const NVFBC_SESSION_HANDLE sessionHandle)**

Gets the last error message that got recorded for a client.

When NvFBC returns an error, it will save an error message that can be queried through this API call. Only the last message is saved. The message and the return code should give enough information about what went wrong.

Parameters:

← *sessionHandle* Handle to the NvFBC client.

Returns:

A NULL terminated error message, or an empty string. Its maximum length is NVFBC_ERROR_STR_LEN.

7.5.2.8 **NVFBCSTATUS NVFBCAPI NvFBCGetStatus (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_GET_STATUS_PARAMS * pParams)**

Gets the current status of the display driver.

This function queries the display driver for various information.

Parameters:

← *sessionHandle* FBC session handle.

← *pParams* NVFBC_GET_STATUS_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_X

7.5.2.9 **NVFBCSTATUS NVFBCAPI NvFBCReleaseContext (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_RELEASE_CONTEXT_PARAMS * pParams)**

Releases the FBC context from the calling thread.

If the FBC context is bound to a different thread, NVFBC_ERR_CONTEXT is returned.

If the FBC context is already released, this function has no effects.

Parameters:

← *sessionHandle* FBC session handle.

← *pParams* NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_X

7.5.2.10 NVFBCSTATUS NVFBCAPI NvFBCToCudaGrabFrame (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_TOCUDA_GRAB_FRAME_PARAMS *pParams)

Captures a frame to a CUDA device in video memory.

This function triggers a frame capture to a CUDA device in video memory.

Note about changes of resolution:

See also:

[NvFBCToSysGrabFrame](#)

Parameters:

← *sessionHandle* FBC session handle.
 ← *pParams* NVFBC_TOCUDA_GRAB_FRAME_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PTR
 NVFBC_ERR_CUDA
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_X

See also:

[NvFBCCreateCaptureSession](#)
[NvFBCToCudaSetUp](#)

7.5.2.11 NVFBCSTATUS NVFBCAPI NvFBCToCudaSetUp (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_TOCUDA_SETUP_PARAMS *pParams)

Sets up a capture to video memory session.

This function configures how the capture to video memory with CUDA interop should behave. It can be called anytime and several times after the capture session has been created. However, it must be called at least once prior to start capturing frames.

Parameters:

← *sessionHandle* FBC session handle.

← *pParams* NVFBC_TOCUDA_SETUP_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_UNSUPPORTED
 NVFBC_ERR_GL
 NVFBC_ERR_MUST_RECREATE
 NVFBC_ERR_X

7.5.2.12 NVFBCSTATUS NVFBCAPI NvFBCToGLGrabFrame (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_TOGL_GRAB_FRAME_PARAMS * pParams)

Captures a frame to an OpenGL buffer in video memory.

This function triggers a frame capture to a selected resource in video memory.

Note about changes of resolution:

See also:

[NvFBCToSysGrabFrame](#)

Parameters:

← *sessionHandle* FBC session handle.
 ← *pParams* NVFBC_TOGL_GRAB_FRAME_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PTR
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_X

See also:

[NvFBCCreateCaptureSession](#)
[NvFBCToCudaSetUp](#)

7.5.2.13 NVFBCSTATUS NVFBCAPI NvFBCToGLSetUp (const NVFBC_SESSION_HANDLE sessionHandle, NVFBC_TOGL_SETUP_PARAMS * pParams)

Sets up a capture to OpenGL buffer in video memory session.

This function configures how the capture to video memory should behave. It can be called anytime and several times after the capture session has been created. However, it must be called at least once prior to start capturing frames.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOGL_SETUP_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_UNSUPPORTED
 NVFBC_ERR_GL
 NVFBC_ERR_MUST_RECREATE
 NVFBC_ERR_X

7.5.2.14 NVFBCSTATUS NVFBCAPI NvFBCToH264GetHeader (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOH264_GET_HEADER_PARAMS * *pParams*)

Gets SPS/PPS headers.

Deprecated**See also:**

[NvFBCToHwEncGetHeader](#)

This function returns the Sequence Parameter Set and Picture Parameter Sets for the current frame.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOH264_GET_HEADER_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PTR
 NVFBC_ERR_ENCODER
 NVFBC_ERR_MUST_RECREATE

7.5.2.15 NVFBCSTATUS NVFBCAPI NvFBCToH264GrabFrame (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOH264_GRAB_FRAME_PARAMS * *pParams*)

Captures a H.264 compressed frame to a bitstream in system memory.

Deprecated**See also:**

[NvFBCToHwEncGrabFrame](#)

This function triggers a H.264 compressed frame capture to a bitstream in system memory.

Note about changes of resolution:

See also:

[NvFBCToSysGrabFrame](#)

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOH264_GRAB_FRAME_PARAMS

Returns:

[NVFBC_SUCCESS](#)
[NVFBC_ERR_INVALID_HANDLE](#)
[NVFBC_ERR_API_VERSION](#)
[NVFBC_ERR_BAD_REQUEST](#)
[NVFBC_ERR_CONTEXT](#)
[NVFBC_ERR_INVALID_PTR](#)
[NVFBC_ERR_CUDA](#)
[NVFBC_ERR_ENCODER](#)
[NVFBC_ERR_INTERNAL](#)
[NVFBC_ERR_X](#)

See also:

[NvFBCCreateCaptureSession](#)
[NvFBCToH264SetUp](#)

7.5.2.16 NVFBCSTATUS NVFBCAPI NvFBCToH264SetUp (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOH264_SETUP_PARAMS * *pParams*)

Sets up a capture to H.264 compressed frames in system memory.

Deprecated**See also:**

[NvFBCToHwEncSetUp](#)

This function configures how the capture to H.264 compressed frames in system memory should behave. It can be called anytime and several times after the capture session has been created. However, it must be called at least once prior to start capturing frames.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOH264_SETUP_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_UNSUPPORTED
 NVFBC_ERR_GL
 NVFBC_ERR_MUST_RECREATE
 NVFBC_ERR_X

7.5.2.17 NVFBCSTATUS NVFBCAPI NvFBCToHwEncGetCaps (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOHWENC_GET_CAPS_PARAMS * *pParams*)

Deprecated

Queries HW encoder capabilities for a given codec.

This function can be used to figure out how to configure the HW encoder.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOHWENC_GET_CAPS_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PARAM
 NVFBC_ERR_ENCODER

7.5.2.18 NVFBCSTATUS NVFBCAPI NvFBCToHwEncGetHeader (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOHWENC_GET_HEADER_PARAMS * *pParams*)

Deprecated

Gets SPS/PPS headers

This function returns the Sequence Parameter Set and Picture Parameter Sets for the current frame.

Parameters:

- ← *sessionHandle* FBC session handle.

← *pParams* NVFBC_TOHWENC_GET_HEADER_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PTR
 NVFBC_ERR_ENCODER
 NVFBC_ERR_MUST_RECREATE
 NVFBC_ERR_X

7.5.2.19 NVFBCSTATUS NVFBCAPI NvFBCToHwEncGrabFrame (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOHWENC_GRAB_FRAME_PARAMS * *pParams*)

Deprecated

Captures a HW compressed frame to a bitstream in system memory.

This function triggers a compressed frame capture to a bitstream in system memory.

Note about changes of resolution:

See also:

[NvFBCToSysGrabFrame](#)

Parameters:

← *sessionHandle* FBC session handle.
 ← *pParams* NVFBC_TOHWENC_GRAB_FRAME_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_INVALID_PTR
 NVFBC_ERR_CUDA
 NVFBC_ERR_ENCODER
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_X

See also:

[NvFBCCreateCaptureSession](#)
[NvFBCToHwEncSetUp](#)

7.5.2.20 NVFBCSTATUS NVFBCAPI NvFBCToHwEncSetup (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOHWENC_SETUP_PARAMS * *pParams*)

Deprecated

Sets up a capture to HW compressed frames in system memory.

This function configures how the capture to compressed frames in system memory should behave. It can be called anytime and several times after the capture session has been created. However, it must be called at least once prior to start capturing frames.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOHWENC_SETUP_PARAMS

Returns:

NVFBC_SUCCESS
 NVFBC_ERR_INVALID_HANDLE
 NVFBC_ERR_API_VERSION
 NVFBC_ERR_BAD_REQUEST
 NVFBC_ERR_INTERNAL
 NVFBC_ERR_CONTEXT
 NVFBC_ERR_UNSUPPORTED
 NVFBC_ERR_GL
 NVFBC_ERR_CUDA
 NVFBC_ERR_INVALID_PARAM
 NVFBC_ERR_ENCODER
 NVFBC_ERR_X

7.5.2.21 NVFBCSTATUS NVFBCAPI NvFBCToSysGrabFrame (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOSYS_GRAB_FRAME_PARAMS * *pParams*)

Captures a frame to a buffer in system memory.

This function triggers a frame capture to a buffer in system memory that was registered with the ToSysSetUp() API call.

Note that it is possible that the resolution of the desktop changes while capturing frames. This should be transparent for the application.

When the resolution changes, the capture session is recreated using the same parameters, and necessary buffers are re-allocated. The frame counter is not reset.

An application can detect that the resolution changed by comparing the dwByteSize member of the NVFBC_FRAME_GRAB_INFO against a previous frame and/or dwWidth and dwHeight.

During a change of resolution the capture is paused even in asynchronous mode.

Parameters:

- ← *sessionHandle* FBC session handle.
- ← *pParams* NVFBC_TOSYS_GRAB_FRAME_PARAMS

Returns:

[NVFBC_SUCCESS](#)
[NVFBC_ERR_INVALID_HANDLE](#)
[NVFBC_ERR_API_VERSION](#)
[NVFBC_ERR_BAD_REQUEST](#)
[NVFBC_ERR_CONTEXT](#)
[NVFBC_ERR_INVALID_PTR](#)
[NVFBC_ERR_INTERNAL](#)
[NVFBC_ERR_X](#)

See also:

[NvFBCCreateCaptureSession](#)
[NvFBCToSysSetUp](#)

7.5.2.22 NVFBCSTATUS NVFBCAPI NvFBCToSysSetUp (const NVFBC_SESSION_HANDLE *sessionHandle*, NVFBC_TOSYS_SETUP_PARAMS **pParams*)

Sets up a capture to system memory session.

This function configures how the capture to system memory should behave. It can be called anytime and several times after the capture session has been created. However, it must be called at least once prior to start capturing frames.

This function allocates the buffer that will contain the captured frame. The application does not need to free this buffer. The size of this buffer is returned in the [NVFBC_FRAME_GRAB_INFO](#) structure.

Parameters:

← *sessionHandle* FBC session handle.
 ← *pParams* [NVFBC_TOSYS_SETUP_PARAMS](#)

Returns:

[NVFBC_SUCCESS](#)
[NVFBC_ERR_INVALID_HANDLE](#)
[NVFBC_ERR_API_VERSION](#)
[NVFBC_ERR_BAD_REQUEST](#)
[NVFBC_ERR_INTERNAL](#)
[NVFBC_ERR_CONTEXT](#)
[NVFBC_ERR_UNSUPPORTED](#)
[NVFBC_ERR_INVALID_PTR](#)
[NVFBC_ERR_INVALID_PARAM](#)
[NVFBC_ERR_OUT_OF_MEMORY](#)
[NVFBC_ERR_X](#)

Chapter 8

Class Documentation

8.1 `_NVFBC_BIND_CONTEXT_PARAMS` Struct Reference

Defines parameters for the [NvFBCBindContext\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to NVFBC_BIND_CONTEXT_PARAMS_VER

8.1.1 Detailed Description

Defines parameters for the [NvFBCBindContext\(\)](#) API call.

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

- [NvFBC.h](#)

8.2 `_NVFBC_BOX` Struct Reference

Box used to describe an area of the tracked region to capture.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t x`
[in] X offset of the box.
- `uint32_t y`
[in] Y offset of the box.
- `uint32_t w`
[in] Width of the box.
- `uint32_t h`
[in] Height of the box.

8.2.1 Detailed Description

Box used to describe an area of the tracked region to capture.

The coordinates are relative to the tracked region.

E.g., if the size of the X screen is 3520x1200 and the tracked RandR output scans a region of 1600x1200+1920+0, then setting a capture box of 800x600+100+50 effectively captures a region of 800x600+2020+50 relative to the X screen.

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

- [NvFBC.h](#)

8.3 _NVFBC_CREATE_CAPTURE_SESSION_PARAMS Struct Reference

Defines parameters for the [NvFBCCreateCaptureSession\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to `NVFBC_CREATE_CAPTURE_SESSION_PARAMS_VER`
- [NVFBC_CAPTURE_TYPE eCaptureType](#)
[in] Desired capture type.
- [NVFBC_TRACKING_TYPE eTrackingType](#)
[in] What region of the framebuffer should be tracked.
- [uint32_t dwOutputId](#)
[in] ID of the output to track if `eTrackingType` is set to `NVFBC_TRACKING_OUTPUT`.
- [NVFBC_BOX captureBox](#)
[in] Crop the tracked region.
- [NVFBC_SIZE frameSize](#)
[in] Desired size of the captured frame.
- [NVFBC_BOOL bWithCursor](#)
[in] Whether the mouse cursor should be composited to the frame.
- [NVFBC_BOOL bDisableAutoModesetRecovery](#)
[in] Whether `NvFBC` should not attempt to recover from modesets.
- [NVFBC_BOOL bRoundFrameSize](#)
[in] Whether `NvFBC` should round the requested `frameSize`.
- [uint32_t dwSamplingRateMs](#)
[in] Rate in `ms` at which the display server generates new frames
- [NVFBC_BOOL bPushModel](#)
[in] Enable push model for frame capture

8.3.1 Detailed Description

Defines parameters for the [NvFBCCreateCaptureSession\(\)](#) API call.

8.3.2 Member Data Documentation

8.3.2.1 NVFBC_BOOL_NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bDisableAutoModesetRecovery

[in] Whether NvFBC should not attempt to recover from modesets.

NvFBC is able to detect when a modeset event occurred and can automatically re-create a capture session with the same settings as before, then resume its frame capture seamlessly.

This option allows to disable this behavior. NVFBC_ERR_MUST_RECREATE will be returned in that case.

It can be useful in the cases when an application needs to do some work between setting up a capture and grabbing the first frame.

For example: an application using the ToGL interface needs to register resources with EncodeAPI prior to encoding frames.

8.3.2.2 NVFBC_BOOL_NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bPushModel

[in] Enable push model for frame capture

When set to NVFBC_TRUE, the display server will generate frames whenever it receives a damage event from applications.

Setting this to NVFBC_TRUE will ignore dwSamplingRateMs.

Using push model with the NVFBC_*_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY capture flag should guarantee the shortest amount of time between an application rendering a frame and an NvFBC client capturing it, provided that the NvFBC client is able to process the frames quickly enough.

Note that applications running at high frame rates will increase CPU and GPU loads.

8.3.2.3 NVFBC_BOOL_NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bRoundFrameSize

[in] Whether NvFBC should round the requested frameSize.

When disabled, NvFBC will not attempt to round the requested resolution.

However, some pixel formats have resolution requirements. E.g., YUV/NV formats must have a width being a multiple of 4, and a height being a multiple of 2. RGB formats don't have such requirements.

If the resolution doesn't meet the requirements of the format, then NvFBC will fail at setup time.

When enabled, NvFBC will round the requested width to the next multiple of 4 and the requested height to the next multiple of 2.

In this case, requesting any resolution will always work with every format. However, an NvFBC client must be prepared to handle the case where the requested resolution is different than the captured resolution.

[NVFBC_FRAME_GRAB_INFO::dwWidth](#) and [NVFBC_FRAME_GRAB_INFO::dwHeight](#) should always be used for getting information about captured frames.

8.3.2.4 NVFBC_BOOL_NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bWithCursor

[in] Whether the mouse cursor should be composited to the frame.

Disabling the cursor will not generate new frames when only the cursor is moved.

8.3.2.5 `NVFB_BOX_NVFB_CREATE_CAPTURE_SESSION_PARAMS::captureBox`

[in] Crop the tracked region.

The coordinates are relative to the tracked region.

It can be set to 0 to capture the entire tracked region.

8.3.2.6 `uint32_t_NVFB_CREATE_CAPTURE_SESSION_PARAMS::dwSamplingRateMs`

[in] Rate in ms at which the display server generates new frames

This controls the frequency at which the display server will generate new frames if new content is available. This effectively controls the capture rate when using blocking calls.

Note that lower values will increase the CPU and GPU loads.

The default value is 16ms (~ 60 Hz).

8.3.2.7 `NVFB_CAPTURE_TYPE_NVFB_CREATE_CAPTURE_SESSION_PARAMS::eCaptureType`

[in] Desired capture type.

Note that when specifying `NVFB_CAPTURE_SHARED_CUDA` or `NVFB_CAPTURE_TO_HW_ENCODER` NvFBC will try to `dlopen()` the corresponding libraries. This means that NvFBC can run on a system without CUDA or HW encoder libraries since it does not link against them.

8.3.2.8 `NVFB_SIZE_NVFB_CREATE_CAPTURE_SESSION_PARAMS::frameSize`

[in] Desired size of the captured frame.

This parameter allow to scale the captured frame.

It can be set to 0 to disable frame resizing.

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

- [NvFBC.h](#)

8.4 `_NVFBC_CREATE_HANDLE_PARAMS` Struct Reference

Defines parameters for the `CreateHandle()` API call.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to `NVFBC_CREATE_HANDLE_PARAMS_VER`
- `const void * privateData`
[in] Application specific private information passed to the NvFBC session.
- `uint32_t privateDataSize`
[in] Size of the application specific private information passed to the NvFBC session.
- `NVFBC_BOOL bExternallyManagedContext`
[in] Whether NvFBC should not create and manage its own graphics context
- `void * glxCtx`
[in] GLX context
- `void * glxFBConfig`
[in] GLX framebuffer configuration

8.4.1 Detailed Description

Defines parameters for the `CreateHandle()` API call.

8.4.2 Member Data Documentation

8.4.2.1 `NVFBC_BOOL _NVFBC_CREATE_HANDLE_PARAMS::bExternallyManagedContext`

[in] Whether NvFBC should not create and manage its own graphics context

NvFBC internally uses OpenGL to perform graphics operations on the captured frames. By default, NvFBC will create and manage (e.g., make current, detect new threads, etc.) its own OpenGL context.

If set to `NVFBC_TRUE`, NvFBC will use the application's context. It will be the application's responsibility to make sure that a context is current on the thread calling into the NvFBC API.

8.4.2.2 `void* _NVFBC_CREATE_HANDLE_PARAMS::glxCtx`

[in] GLX context

GLX context that NvFBC should use internally to create pixmaps and make them current when creating a new capture session.

Note: NvFBC expects a context created against a `GLX_RGBA_TYPE` render type.

8.4.2.3 `void* _NVFBC_CREATE_HANDLE_PARAMS::glxFBConfig`

[in] GLX framebuffer configuration

Framebuffer configuration that was used to create the GLX context, and that will be used to create pixmaps internally.

Note: NvFBC expects a configuration having at least the following attributes: `GLX_DRAWABLE_TYPE`, `GLX_PIXMAP_BIT`, `GLX_BIND_TO_TEXTURE_RGBA_EXT`, `1`, `GLX_BIND_TO_TEXTURE_TARGETS_EXT`, `GLX_TEXTURE_2D_BIT_EXT`

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

- [NvFBC.h](#)

8.5 `_NVFBC_DESTROY_CAPTURE_SESSION_PARAMS` Struct Reference

Defines parameters for the [NvFBCDestroyCaptureSession\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`

[in] Must be set to `NVFBC_DESTROY_CAPTURE_SESSION_PARAMS_VER`

8.5.1 Detailed Description

Defines parameters for the [NvFBCDestroyCaptureSession\(\)](#) API call.

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

- [NvFBC.h](#)

8.6 _NVFBC_DESTROY_HANDLE_PARAMS Struct Reference

Defines parameters for the [NvFBCDestroyHandle\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to NVFBC_DESTROY_HANDLE_PARAMS_VER

8.6.1 Detailed Description

Defines parameters for the [NvFBCDestroyHandle\(\)](#) API call.

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

- [NvFBC.h](#)

8.7 `_NVFBC_FRAME_GRAB_INFO` Struct Reference

Describes information about a captured frame.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwWidth`
[out] Width of the captured frame.
- `uint32_t dwHeight`
[out] Height of the captured frame.
- `uint32_t dwByteSize`
[out] Size of the frame in bytes.
- `uint32_t dwCurrentFrame`
[out] Incremental ID of the current frame.
- `NVFBC_BOOL bIsNewFrame`
[out] Whether the captured frame is a new frame.
- `uint64_t ulTimestampUs`
[out] Frame timestamp

8.7.1 Detailed Description

Describes information about a captured frame.

8.7.2 Member Data Documentation

8.7.2.1 `NVFBC_BOOL _NVFBC_FRAME_GRAB_INFO::bIsNewFrame`

[out] Whether the captured frame is a new frame.

When using non blocking calls it is possible to capture a frame that was already captured before if the display server did not render a new frame in the meantime. In that case, this flag will be set to `NVFBC_FALSE`.

When using blocking calls each captured frame will have this flag set to `NVFBC_TRUE` since the blocking mechanism waits for the display server to render a new frame.

Note that this flag does not guarantee that the content of the frame will be different compared to the previous captured frame.

In particular, some compositing managers report the entire framebuffer as damaged when an application refreshes its content.

Consider a single X screen spanned across physical displays A and B and an NvFBC application tracking display A. Depending on the compositing manager, it is possible that an application refreshing itself on display B will trigger a frame capture on display A.

Workarounds include:

- Using separate X screens
- Disabling the composite extension
- Using a compositing manager that properly reports what regions are damaged
- Using NvFBC's diffmaps to find out if the frame changed

8.7.2.2 `uint32_t _NVFBC_FRAME_GRAB_INFO::dwCurrentFrame`

[out] Incremental ID of the current frame.

This can be used to identify a frame.

8.7.2.3 `uint64_t _NVFBC_FRAME_GRAB_INFO::ulTimestampUs`

[out] Frame timestamp

Time in micro seconds when the display server started rendering the frame.

This does not account for when the frame was captured. If capturing an old frame (e.g., `bIsNewFrame` is `NVFBC_FALSE`) the reported timestamp will reflect the time when the old frame was rendered by the display server.

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

- [NvFBC.h](#)

8.8 `_NVFBC_GET_STATUS_PARAMS` Struct Reference

Defines parameters for the `NvFBCGetStatus()` API call.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to `NVFBC_GET_STATUS_PARAMS_VER`
- `NVFBC_BOOL bIsCapturePossible`
[out] Whether or not framebuffer capture is supported by the graphics driver.
- `NVFBC_BOOL bCurrentlyCapturing`
[out] Whether or not there is already a capture session on this system.
- `NVFBC_BOOL bCanCreateNow`
[out] Whether or not it is possible to create a capture session on this system.
- `NVFBC_SIZE screenSize`
[out] Size of the X screen (framebuffer).
- `NVFBC_BOOL bXRandRAvailable`
[out] Whether the XRandR extension is available.
- `NVFBC_RANDR_OUTPUT_INFO outputs [NVFBC_OUTPUT_MAX]`
[out] Array of outputs connected to the X screen.
- `uint32_t dwOutputNum`
[out] Number of outputs connected to the X screen.
- `uint32_t dwNvFBCVersion`
[out] Version of the NvFBC library running on this system.

8.8.1 Detailed Description

Defines parameters for the `NvFBCGetStatus()` API call.

8.8.2 Member Data Documentation

8.8.2.1 `NVFBC_BOOL _NVFBC_GET_STATUS_PARAMS::bXRandRAvailable`

[out] Whether the XRandR extension is available.

If this extension is not available then it is not possible to have information about RandR outputs.

8.8.2.2 uint32_t _NVFBC_GET_STATUS_PARAMS::dwOutputNum

[out] Number of outputs connected to the X screen.

This must be used to parse the array of connected outputs.

Only if XRandR is available.

8.8.2.3 NVFBC_RANDR_OUTPUT_INFO _NVFBC_GET_STATUS_PARAMS::outputs[NVFBC_OUTPUT_MAX]

[out] Array of outputs connected to the X screen.

An application can track a specific output by specifying its ID when creating a capture session.

Only if XRandR is available.

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

- [NvFBC.h](#)

8.9 _NVFBC_HWENC_CONFIG Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Sets to `NVFBC_HWENC_CONFIG_VER`.
- [uint32_t dwProfile](#)
[in] Codec profile that the HW encoder should use for video encoding.
- [uint32_t dwFrameRateNum](#)
[in] Frame rate numerator.
- [uint32_t dwFrameRateDen](#)
[in] Frame rate denominator.
- [uint32_t dwAvgBitRate](#)
[in] Target Average bitrate.
- [uint32_t dwPeakBitRate](#)
[in] Maximum bitrate that the HW encoder can hit while encoding video in VBR [variable bit rate mode].
- [uint32_t dwGOPLength](#)
[in] Number of pictures in a GOP.
- [uint32_t dwQP](#)
[in] QP value to be used for rate control in ConstQP mode.
- [NVFBC_HWENC_PARAMS_RC_MODE eRateControl](#)
[in] Rate Control Mode to be used by the HW encoder.
- [NVFBC_HWENC_PRESET ePresetConfig](#)
[in] Specifies the encoding preset used for fine tuning for encoding parameters.
- [NVFBC_BOOL bOutBandSPSPPS](#)
[in] Enables out of band generation of SPS, PPS headers.
- [NVFBC_BOOL bIntraFrameOnRequest](#)
[in] Allows to use the flag `NVFBC_HWENC_PARAM_FLAG_FORCEIDR`.
- [NVFBC_BOOL bUseMaxRCQP](#)
[in] Enable this if client wants to specify `maxRCQP[]`.
- [NVFBC_BOOL bEnableIntraRefresh](#)
[in] Enables gradual decoder refresh or intra-refresh.
- [NVFBC_HWENC_SLICING_MODE dwSlicingMode](#)

[in] Refer to enum [NVFBC_HWENC_SLICING_MODE](#) for usage.

- [uint32_t dwSlicingModeParam](#)

[in] Refer to enum [NVFBC_HWENC_SLICING_MODE](#) for usage.

- [uint32_t dwVBVBufferSize](#)

[in]: VBV buffer size can be used to cap the frame size of encoded bitstream, reducing the need for buffering at decoder end.

- [uint32_t dwVBVInitialDelay](#)

[in] Number of bits to buffer at the decoder end.

- [uint32_t maxRCQP](#) [3]

[in] $maxQP[0]$ = max QP for P frame, $maxRCQP[1]$ = max QP for B frame, $maxRCQP[2]$ = max QP for I frame respectively.

- [uint32_t dwMaxNumRefFrames](#)

[in] This is used to specify the DPB size used for encoding.

- [NVFBC_BOOL bEnableMSE](#)

[in] Enables returning the mean squared error (MSE) per channel in [NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#).

- [NVFBC_BOOL bEnableAQ](#)

[in] Enables adaptive quantization.

- [NVFBC_BUFFER_FORMAT eInputBufferFormat](#)

[in] Input format.

- [NVFBC_HWENC_CODEC codec](#)

[in] Codec used to encode frames.

8.9.1 Detailed Description

Deprecated

Describes HW encoder configuration.

8.9.2 Member Data Documentation

8.9.2.1 NVFBC_BOOL_NVFBC_HWENC_CONFIG::bEnableAQ

[in] Enables adaptive quantization.

Currently supported only when [NVFBC_HWENC_CONFIG::eRateControl](#) is set to [NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY](#) or [NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP](#).

8.9.2.2 NVFBC_BOOL_NVFBC_HWENC_CONFIG::bEnableIntraRefresh

[in] Enables gradual decoder refresh or intra-refresh.

If the GOP structure uses B frames this will be ignored.

8.9.2.3 NVFBC_BOOL_NVFBC_HWENC_CONFIG::bEnableMSE

[in] Enables returning the mean squared error (MSE) per channel in [NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#).

NOTE: Enabling this bit will affect performance severely; set it only if the caller wants to evaluate quality of encoder.

8.9.2.4 NVFBC_BOOL_NVFBC_HWENC_CONFIG::bOutBandSPSPPS

[in] Enables out of band generation of SPS, PPS headers.

The frame header can be queried using [NvFBCToHwEncGetHeader\(\)](#).

8.9.2.5 uint32_t_NVFBC_HWENC_CONFIG::dwAvgBitRate

[in] Target Average bitrate.

HW Encoder will try to achieve this bitrate during video encoding.

This is the only bitrate setting useful for Constant Bit Rate RateControl mode.

8.9.2.6 uint32_t_NVFBC_HWENC_CONFIG::dwFrameRateDen

[in] Frame rate denominator.

Encoding frame rate = $\text{dwFrameRateNum}/\text{dwFrameRateDen}$.

For example, the "1000" in 33333/1000 (for a frame rate of 33.333 fps).

This is not related to rate at which frames are grabbed.

8.9.2.7 uint32_t_NVFBC_HWENC_CONFIG::dwFrameRateNum

[in] Frame rate numerator.

Encoding frame rate = $\text{dwFrameRateNum}/\text{dwFrameRateDen}$.

For example, the "33333" in 33333/1000 (for a frame rate of 33.333 fps).

This is not related to rate at which frames are grabbed.

8.9.2.8 uint32_t_NVFBC_HWENC_CONFIG::dwGOPLength

[in] Number of pictures in a GOP.

Every GOP begins with an I frame, so this is the same as I-frame interval.

8.9.2.9 uint32_t_NVFBC_HWENC_CONFIG::dwMaxNumRefFrames

[in] This is used to specify the DPB size used for encoding.

Setting this to 0 will allow encoder to use the default DPB size. Low latency applications are recommended to use a large DPB size (recommended size is 16) so that it allows clients to invalidate corrupt frames and use older frames for reference to improve error resiliency.

8.9.2.10 uint32_t _NVFBC_HWENC_CONFIG::dwProfile

[in] Codec profile that the HW encoder should use for video encoding.

For the H.264 codec: 0: Autoselect appropriate codec profile. 66: Baseline (fastest encode/decode times, lowest image quality, lowest bitrate) 77: Main (balanced encode/decode times, image) 100: High (slowest encode/decode times, highest image quality, highest bitrate) 244: High, used for lossless and YUV444P encoding

For the H.265/HEVC codec: 1: Main profile

8.9.2.11 uint32_t _NVFBC_HWENC_CONFIG::dwVBVBufferSize

[in]: VBV buffer size can be used to cap the frame size of encoded bitstream, reducing the need for buffering at decoder end.

For lowest latency, VBV buffersize = single frame size = channel bitrate/frame rate.

I-frame size may be larger than P or B frames. Overridden by NvFBC in case of NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY or NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP rate control modes.

8.9.2.12 uint32_t _NVFBC_HWENC_CONFIG::dwVBVInitialDelay

[in] Number of bits to buffer at the decoder end.

For lowest latency, set to be equal to dwVBVBufferSize.

Overridden by NvFBC in case of NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY or NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP rate control modes.

8.9.2.13 NVFBC_BUFFER_FORMAT _NVFBC_HWENC_CONFIG::eInputBufferFormat

[in] Input format.

Must be set to NVFBC_BUFFER_FORMAT_YUV420P or NVFBC_BUFFER_FORMAT_YUV444P.

If set to NVFBC_BUFFER_FORMAT_YUV444P, [NVFBC_HWENC_CONFIG::dwProfile](#) is forced to 244. Bitrate should be manually set ~20-30% higher than what is set otherwise for NVFBC_BUFFER_FORMAT_YUV420P.

NVFBC_BUFFER_FORMAT_YUV444 is available on Maxwell GPUs onwards and is only supported with the H.264 codec.

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

- [NvFBC.h](#)

8.10 _NVFBC_HWENC_ENCODE_PARAMS Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Set to NVFBC_HWENC_ENCODE_PARAMS_VER.
- [uint32_t dwEncodeParamFlags](#)
[in] Specifies bit-wise OR'ed encode param flags.
- [uint32_t dwNewAvgBitRate](#)
[in] Specifies the new average bit rate to be used from this frame onwards.
- [uint32_t dwNewPeakBitRate](#)
[in] Specifies the new peak bit rate to be used from this frame onwards.
- [uint32_t dwNewVBVBufferSize](#)
[in] Specifies the new VBV buffer size to be used from this frame onwards.
- [uint32_t dwNewVBVInitialDelay](#)
[in] Specifies the new VBV initial delay to be used from this frame onwards.
- [uint64_t ulCaptureTimeStamp](#)
[in] Input timestamp to be associated with this input picture.
- [uint64_t ulInvalidFrameTimeStamp](#) [NVFBC_MAX_REF_FRAMES]
[in] Specifies an array of timestamps of the encoder references which the client wants to invalidate.
- [NVFBC_BOOL bInvalidateReferenceFrames](#)
[in] Enable this if client wants encoder reference frames to be invalidated.
- [NVFBC_BOOL bStartIntraRefresh](#)
[in] Enable this if the client wants to force Intra Refresh with intra refresh period *dwIntraRefreshCnt*.
- [NVFBC_BOOL bReEncodePrevFrame](#)
[in] Enable this if client wants to re-encode previous most recently captured frame with different encoding params, NvFBC will not capture new frame.
- [uint32_t dwNumRefFramesToInvalidate](#)
[in] Specifies number of encoder references which the client wants to invalidate
- [uint32_t dwIntraRefreshCnt](#)
[in] Specifies the number of frames over which intra refresh will happen, if *bStartIntraRefresh* is set.

8.10.1 Detailed Description

Deprecated

Describes encode parameters.

8.10.2 Member Data Documentation

8.10.2.1 `NVFBC_BOOL_NVFBC_HWENC_ENCODE_PARAMS::bInvalidateReferenceFrames`

[in] Enable this if client wants encoder reference frames to be invalidated.

Ignored if Intra-Refresh is enabled for the session.

8.10.2.2 `NVFBC_BOOL_NVFBC_HWENC_ENCODE_PARAMS::bReEncodePrevFrame`

[in] Enable this if client wants to re-encode previous most recently captured frame with different encoding params, NVFBC will not capture new frame.

If no frame has been captured yet, this option is ignored.

Setting this option ignores blocking calls.

8.10.2.3 `uint32_t_NVFBC_HWENC_ENCODE_PARAMS::dwEncodeParamFlags`

[in] Specifies bit-wise OR'ed encode param flags.

See [NVFBC_HWENC_PARAM_FLAGS](#) enum.

8.10.2.4 `uint32_t_NVFBC_HWENC_ENCODE_PARAMS::dwNewVBVBufferSize`

[in] Specifies the new VBV buffer size to be used from this frame onwards.

Client is expected to pass new appropriate VBV values.

8.10.2.5 `uint32_t_NVFBC_HWENC_ENCODE_PARAMS::dwNewVBVInitialDelay`

[in] Specifies the new VBV initial delay to be used from this frame onwards.

Client is expected to pass new appropriate VBV values.

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

- [NvFBC.h](#)

8.11 `_NVFBC_HWENC_FRAME_INFO` Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- [NVFBC_BOOL bIsIFrame](#)
[out] Is `NVFBC_TRUE` if the current frame is an I-frame.
- `uint32_t` [dwByteSize](#)
[out] Size of bitstream produced, in bytes.
- `uint64_t` [ulTimeStamp](#)
[out] Timestamp associated with the encoded frame.

8.11.1 Detailed Description

Deprecated

Describes an encoded frame.

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

- [NvFBC.h](#)

8.12 _NVFBC_OUTPUT Struct Reference

Describes an RandR output.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwId](#)
Identifier of the RandR output.
- [char name](#) [NVFBC_OUTPUT_NAME_LEN]
Name of the RandR output, as reported by tools such as xrandr(1).
- [NVFBC_BOX](#) [trackedBox](#)
Region of the X screen tracked by the RandR CRTC driving this RandR output.

8.12.1 Detailed Description

Describes an RandR output.

Filling this structure relies on the XRandR extension. This feature cannot be used if the extension is missing or its version is below the requirements.

See also:

[Requirements](#)

8.12.2 Member Data Documentation

8.12.2.1 [char _NVFBC_OUTPUT::name](#)[NVFBC_OUTPUT_NAME_LEN]

Name of the RandR output, as reported by tools such as xrandr(1).

Example: "DVI-I-0"

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

- [NvFBC.h](#)

8.13 `_NVFBC_RELEASE_CONTEXT_PARAMS` Struct Reference

Defines parameters for the [NvFBCReleaseContext\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to NVFBC_RELEASE_CONTEXT_PARAMS_VER

8.13.1 Detailed Description

Defines parameters for the [NvFBCReleaseContext\(\)](#) API call.

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

- [NvFBC.h](#)

8.14 _NVFBC_SIZE Struct Reference

Size used to describe the size of a frame.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t w](#)
[in] Width.
- [uint32_t h](#)
[in] Height.

8.14.1 Detailed Description

Size used to describe the size of a frame.

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

- [NvFBC.h](#)

8.15 _NVFBC_TOCUDA_GRAB_FRAME_PARAMS Struct Reference

Defines parameters for the [NvFBCToCudaGrabFrame\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to `NVFBC_TOCUDA_GRAB_FRAME_PARAMS_VER`.
- [uint32_t dwFlags](#)
[in] Flags defining the behavior of this frame capture.
- `void *` [pCUDADeviceBuffer](#)
[out] Pointer to a `CUdeviceptr`
- `NVFBC_FRAME_GRAB_INFO *` [pFrameGrabInfo](#)
[out] Information about the captured frame.
- [uint32_t dwTimeoutMs](#)
[in] Wait timeout in milliseconds.

8.15.1 Detailed Description

Defines parameters for the [NvFBCToCudaGrabFrame\(\)](#) API call.

8.15.2 Member Data Documentation

8.15.2.1 `uint32_t _NVFBC_TOCUDA_GRAB_FRAME_PARAMS::dwTimeoutMs`

[in] Wait timeout in milliseconds.

When capturing frames with the `NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS` or `NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY` flags, NvFBC will wait for a new frame or mouse move until the below timer expires.

When timing out, the last captured frame will be returned. Note that as long as the `NVFBC_TOCUDA_GRAB_FLAGS_FORCE_REFRESH` flag is not set, returning an old frame will incur no performance penalty.

NvFBC clients can use the return value of the grab frame operation to find out whether a new frame was captured, or the timer expired.

Note that the behavior of blocking calls is to wait for a new frame *after* the call has been made. When using timeouts, it is possible that NvFBC will return a new frame (e.g., it has never been captured before) even though no new frame was generated after the grab call.

For the precise definition of what constitutes a new frame, see `bIsNewFrame`.

Set to 0 to disable timeouts.

8.15.2.2 `void* _NVFBC_TOCUDA_GRAB_FRAME_PARAMS::pCUDADeviceBuffer`

[out] Pointer to a `CUdeviceptr`

The application does not need to allocate memory for this CUDA device.

The application does need to create its own CUDA context to use this CUDA device.

This `CUdeviceptr` will be mapped to a segment in video memory containing the frame. It is not possible to process a CUDA device while capturing a new frame. If the application wants to do so, it must copy the CUDA device using `cuMemcpyDtoD` or `cuMemcpyDtoH` beforehand.

8.15.2.3 `NVFBC_FRAME_GRAB_INFO* _NVFBC_TOCUDA_GRAB_FRAME_PARAMS::pFrameGrabInfo`

[out] Information about the captured frame.

Can be `NULL`.

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

- [NvFBC.h](#)

8.16 `_NVFBC_TOCUDA_SETUP_PARAMS` Struct Reference

Defines parameters for the `NvFBCToCudaSetUp()` API call.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to `NVFBC_TOCUDA_SETUP_PARAMS_VER`
- `NVFBC_BUFFER_FORMAT eBufferFormat`
[in] Desired buffer format.

8.16.1 Detailed Description

Defines parameters for the `NvFBCToCudaSetUp()` API call.

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

- `NvFBC.h`

8.17 _NVFBC_TOGL_GRAB_FRAME_PARAMS Struct Reference

Defines parameters for the [NvFBCToGLGrabFrame\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to NVFBC_TOGL_GRAB_FRAME_PARAMS_VER.
- [uint32_t dwFlags](#)
[in] Flags defining the behavior of this frame capture.
- [uint32_t dwTextureIndex](#)
[out] Index of the texture storing the current frame.
- [NVFBC_FRAME_GRAB_INFO * pFrameGrabInfo](#)
[out] Information about the captured frame.
- [uint32_t dwTimeoutMs](#)
[in] Wait timeout in milliseconds.

8.17.1 Detailed Description

Defines parameters for the [NvFBCToGLGrabFrame\(\)](#) API call.

8.17.2 Member Data Documentation

8.17.2.1 [uint32_t _NVFBC_TOGL_GRAB_FRAME_PARAMS::dwTextureIndex](#)

[out] Index of the texture storing the current frame.

This is an index in the [NVFBC_TOGL_SETUP_PARAMS::dwTextures](#) array.

8.17.2.2 [uint32_t _NVFBC_TOGL_GRAB_FRAME_PARAMS::dwTimeoutMs](#)

[in] Wait timeout in milliseconds.

When capturing frames with the [NVFBC_TOGL_GRAB_FLAGS_NOFLAGS](#) or [NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY](#) flags, NvFBC will wait for a new frame or mouse move until the below timer expires.

When timing out, the last captured frame will be returned. Note that as long as the [NVFBC_TOGL_GRAB_FLAGS_FORCE_REFRESH](#) flag is not set, returning an old frame will incur no performance penalty.

NvFBC clients can use the return value of the grab frame operation to find out whether a new frame was captured, or the timer expired.

Note that the behavior of blocking calls is to wait for a new frame *after* the call has been made. When using timeouts, it is possible that NvFBC will return a new frame (e.g., it has never been captured before) even though no new frame was generated after the grab call.

For the precise definition of what constitutes a new frame, see `bIsNewFrame`.

Set to 0 to disable timeouts.

8.17.2.3 NVFBC_FRAME_GRAB_INFO*_NVFBC_TOGL_GRAB_FRAME_PARAMS::pFrameGrabInfo

[out] Information about the captured frame.

Can be NULL.

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

- [NvFBC.h](#)

8.18 _NVFBC_TOGL_SETUP_PARAMS Struct Reference

Defines parameters for the [NvFBCToGLSetup\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to `NVFBC_TOGL_SETUP_PARAMS_VER`
- [NVFBC_BUFFER_FORMAT eBufferFormat](#)
[in] Desired buffer format.
- [NVFBC_BOOL bWithDiffMap](#)
[in] Whether differential maps should be generated.
- `void ** ppDiffMap`
[out] Pointer to a pointer to a buffer in system memory.
- [uint32_t dwDiffMapScalingFactor](#)
[in] Scaling factor of the differential maps.
- [uint32_t dwTextures](#) [`NVFBC_TOGL_TEXTURES_MAX`]
[out] List of GL textures that will store the captured frames.
- [uint32_t dwTexTarget](#)
[out] GL target to which the texture should be bound.
- [uint32_t dwTexFormat](#)
[out] GL format of the textures.
- [uint32_t dwTexType](#)
[out] GL type of the textures.
- [NVFBC_SIZE diffMapSize](#)
[out] Size of the differential map.

8.18.1 Detailed Description

Defines parameters for the [NvFBCToGLSetup\(\)](#) API call.

8.18.2 Member Data Documentation

8.18.2.1 NVFBC_SIZE _NVFBC_TOGL_SETUP_PARAMS::diffMapSize

[out] Size of the differential map.

Only set if `bWithDiffMap` is set to `NVFBC_TRUE`.

8.18.2.2 `uint32_t _NVFBC_TOGL_SETUP_PARAMS::dwDiffMapScalingFactor`

[in] Scaling factor of the differential maps.

See also:

[NVFBC_TOSYS_SETUP_PARAMS::dwDiffMapScalingFactor](#)

8.18.2.3 `uint32_t _NVFBC_TOGL_SETUP_PARAMS::dwTextures[NVFBC_TOGL_TEXTURES_MAX]`

[out] List of GL textures that will store the captured frames.

This array is 0 terminated. The number of textures varies depending on the capture settings (such as whether diffmaps are enabled).

An application wishing to interop with, for example, EncodeAPI will need to register these textures prior to start encoding frames.

After each frame capture, the texture holding the current frame will be returned in `NVFBC_TOGL_GRAB_FRAME_PARAMS::dwTexture`.

8.18.2.4 `void** _NVFBC_TOGL_SETUP_PARAMS::ppDiffMap`

[out] Pointer to a pointer to a buffer in system memory.

See also:

[NVFBC_TOSYS_SETUP_PARAMS::ppDiffMap](#)

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

- [NvFBC.h](#)

8.19 _NVFBC_TOHWENC_GET_CAPS_PARAMS Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to `NVFBC_TOHWENC_GET_CAPS_PARAMS_VER`.
- `NVFBC_HWENC_CODEC` codec
[in] Codec against which the capabilities are queried.
- `NVFBC_BOOL bCodecSupported`
[out] Whether the requested codec is supported.
- `NVFBC_BOOL bYUV444`
[out] Whether `NVFBC_BUFFER_FORMAT_YUV444P` is supported.
- `NVFBC_BOOL bLossless`
[out] Whether `NVFBC_HWENC_PRESET_LOSSLESS_HP` is supported.
- `uint32_t dwMaxWidth`
[out] Maximum frame width supported.
- `uint32_t dwMaxHeight`
[out] Maximum frame height supported.
- `uint32_t dwMaxMB`
[out] Maximum frame size in MB.
- `uint32_t dwMaxMBPerSec`
[out] Maximum aggregate throughput in MB/s.
- `NVFBC_BOOL bRcConstQP`
[out] Whether `NVFBC_HWENC_PARAMS_RC_CONSTQP` is supported.
- `NVFBC_BOOL bRcVbr`
[out] Whether `NVFBC_HWENC_PARAMS_RC_VBR` is supported.
- `NVFBC_BOOL bRcCbr`
[out] Whether `NVFBC_HWENC_PARAMS_RC_CBR` is supported.
- `NVFBC_BOOL bRc2PassQuality`
[out] Whether `NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY` is supported.
- `NVFBC_BOOL bRc2PassFramesizeCap`
[out] Whether `NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP` is supported.
- `NVFBC_BOOL bDynResChange`

[out] Whether dynamic resolution change is supported.

- [NVFBC_BOOL bDynBitrateChange](#)

[out] Whether [NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE](#) is supported.

- [NVFBC_BOOL bIntraRefresh](#)

[out] Whether [NVFBC_HWENC_CONFIG::bEnableIntraRefresh](#) is supported.

- [NVFBC_BOOL bCustomVBVBufSize](#)

[out] Whether custom VBV buffer size is supported.

8.19.1 Detailed Description

Deprecated

Defines parameters for the `ToHwGetCaps()` API call.

8.19.2 Member Data Documentation

8.19.2.1 [NVFBC_BOOL _NVFBC_TOHWENC_GET_CAPS_PARAMS::bCodecSupported](#)

[out] Whether the requested codec is supported.

Note: If the codec is not supported, the rest of the structure will stay untouched.

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

- [NvFBC.h](#)

8.20 _NVFBC_TOHWENC_GET_HEADER_PARAMS Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to `NVFBC_TOHWENC_GET_HEADER_PARAMS_VER`.
- `uint32_t dwByteSize`
[out] Contains size in bytes of the SPS/PPS header data written by the HW encoder.
- `uint8_t * pBuffer`
[out] Pointer to a client allocated buffer.

8.20.1 Detailed Description

Deprecated

Defines parameters for the `NvFBCToHwEncGetHeader()` API call.

8.20.2 Member Data Documentation

8.20.2.1 `uint8_t* _NVFBC_TOHWENC_GET_HEADER_PARAMS::pBuffer`

[out] Pointer to a client allocated buffer.

NvFBC HW encoder writes SPS/PPS data to this buffer.

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

- [NvFBC.h](#)

8.21 `_NVFBC_TOHWENC_GRAB_FRAME_PARAMS` Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER.
- `uint32_t dwFlags`
[in] Flags defining the behavior of this frame capture.
- `NVFBC_FRAME_GRAB_INFO * pFrameGrabInfo`
[out] Information about the captured frame.
- `void ** ppBitStreamBuffer`
[out] Pointer to a pointer to a bitstream buffer in system memory.
- `NVFBC_HWENC_ENCODE_PARAMS * pEncodeParams`
[in] Pointer to a structure containing per-frame configuration parameters for the HW encoder.
- `NVFBC_HWENC_FRAME_INFO * pEncFrameInfo`
[out] Pointer to a structure containing data about the captured frame.
- `uint32_t dwEncFrameInfoVer`
[in] Must be set to NVFBC_HWENC_FRAME_INFO_VER.
- `uint32_t dwMSE [3]`
[out] Mean squared error used to evaluate quality.

8.21.1 Detailed Description

Deprecated

Defines parameters for the `NvFBCToHwEncGrabFrame()` API call.

8.21.2 Member Data Documentation

8.21.2.1 `uint32_t _NVFBC_TOHWENC_GRAB_FRAME_PARAMS::dwMSE[3]`

[out] Mean squared error used to evaluate quality.

Set the `bEnableMSE` flag in `NVFBC_HWENC_CONFIG` to enable getting MSE values per channel (Y, U, V).

8.21.2.2 `NVFBC_FRAME_GRAB_INFO* _NVFBC_TOHWENC_GRAB_FRAME_PARAMS::pFrameGrabInfo`

[out] Information about the captured frame.

Can be NULL.

8.21.2.3 `void** _NVFBC_TOHWENC_GRAB_FRAME_PARAMS::ppBitStreamBuffer`

[out] Pointer to a pointer to a bitstream buffer in system memory.

The application does not need to allocate memory for this buffer. It does not need to free this buffer either.

NvFBC will write encoded bitstream data in the buffer.

The content of this buffer cannot be used while capturing a new frame. If an application wants to behave that way, it must copy this buffer beforehand.

The size of this buffer is returned in the `dwByteSize` field of the [NVFBC_FRAME_GRAB_INFO](#) structure.

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

- [NvFBC.h](#)

8.22 `_NVFBC_TOHWENC_SETUP_PARAMS` Struct Reference

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to NVFBC_TOHWENC_SETUP_PARAMS_VER.
- `NVFBC_HWENC_CONFIG * pEncodeConfig`
[in] HW encoder initial configuration parameters.

8.22.1 Detailed Description

Deprecated

Defines parameters for the `ToHwEncSetUp()` API call.

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

- [NvFBC.h](#)

8.23 _NVFBC_TOSYS_GRAB_FRAME_PARAMS Struct Reference

Defines parameters for the [NvFBCToSysGrabFrame\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to `NVFBC_TOSYS_GRAB_FRAME_PARAMS_VER`
- [uint32_t dwFlags](#)
[in] Flags defining the behavior of this frame capture.
- [NVFBC_FRAME_GRAB_INFO * pFrameGrabInfo](#)
[out] Information about the captured frame.
- [uint32_t dwTimeoutMs](#)
[in] Wait timeout in milliseconds.

8.23.1 Detailed Description

Defines parameters for the [NvFBCToSysGrabFrame\(\)](#) API call.

8.23.2 Member Data Documentation

8.23.2.1 [uint32_t _NVFBC_TOSYS_GRAB_FRAME_PARAMS::dwTimeoutMs](#)

[in] Wait timeout in milliseconds.

When capturing frames with the `NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS` or `NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY` flags, NvFBC will wait for a new frame or mouse move until the below timer expires.

When timing out, the last captured frame will be returned. Note that as long as the `NVFBC_TOSYS_GRAB_FLAGS_FORCE_REFRESH` flag is not set, returning an old frame will incur no performance penalty.

NvFBC clients can use the return value of the grab frame operation to find out whether a new frame was captured, or the timer expired.

Note that the behavior of blocking calls is to wait for a new frame *after* the call has been made. When using timeouts, it is possible that NvFBC will return a new frame (e.g., it has never been captured before) even though no new frame was generated after the grab call.

For the precise definition of what constitutes a new frame, see `bIsNewFrame`.

Set to 0 to disable timeouts.

8.23.2.2 [NVFBC_FRAME_GRAB_INFO* _NVFBC_TOSYS_GRAB_FRAME_PARAMS::pFrameGrabInfo](#)

[out] Information about the captured frame.

Can be NULL.

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

- [NvFBC.h](#)

8.24 _NVFBC_TOSYS_SETUP_PARAMS Struct Reference

Defines parameters for the [NvFBCToSysSetUp\(\)](#) API call.

```
#include <NvFBC.h>
```

Public Attributes

- [uint32_t dwVersion](#)
[in] Must be set to NVFBC_TOSYS_SETUP_PARAMS_VER
- [NVFBC_BUFFER_FORMAT eBufferFormat](#)
[in] Desired buffer format.
- [void ** ppBuffer](#)
[out] Pointer to a pointer to a buffer in system memory.
- [NVFBC_BOOL bWithDiffMap](#)
[in] Whether differential maps should be generated.
- [void ** ppDiffMap](#)
[out] Pointer to a pointer to a buffer in system memory.
- [uint32_t dwDiffMapScalingFactor](#)
[in] Scaling factor of the differential maps.
- [NVFBC_SIZE diffMapSize](#)
[out] Size of the differential map.

8.24.1 Detailed Description

Defines parameters for the [NvFBCToSysSetUp\(\)](#) API call.

8.24.2 Member Data Documentation

8.24.2.1 NVFBC_SIZE _NVFBC_TOSYS_SETUP_PARAMS::diffMapSize

[out] Size of the differential map.

Only set if `bWithDiffMap` is set to `NVFBC_TRUE`.

8.24.2.2 uint32_t _NVFBC_TOSYS_SETUP_PARAMS::dwDiffMapScalingFactor

[in] Scaling factor of the differential maps.

For example, a scaling factor of 16 means that one pixel of the diffmap will represent 16x16 pixels of the original frames.

If any of these 16x16 pixels is different between the current and the previous frame, then the corresponding pixel in the diffmap will be set to non-zero.

The default scaling factor is 1. A `dwDiffMapScalingFactor` of 0 will be set to 1.

8.24.2.3 void** _NVFBC_TOSYS_SETUP_PARAMS::ppBuffer

[out] Pointer to a pointer to a buffer in system memory.

This buffer contains the pixel value of the requested format. Refer to the description of the buffer formats to understand the memory layout.

The application does not need to allocate memory for this buffer. It should not free this buffer either. This buffer is automatically re-allocated when needed (e.g., when the resolution changes).

This buffer is allocated by the NvFBC library to the proper size. This size is returned in the dwByteSize field of the [NVFBC_FRAME_GRAB_INFO](#) structure.

8.24.2.4 void** _NVFBC_TOSYS_SETUP_PARAMS::ppDiffMap

[out] Pointer to a pointer to a buffer in system memory.

This buffer contains the differential map of two frames. It must be read as an array of unsigned char. Each unsigned char is either 0 or non-zero. 0 means that the pixel value at the given location has not changed since the previous captured frame. Non-zero means that the pixel value has changed.

The application does not need to allocate memory for this buffer. It should not free this buffer either. This buffer is automatically re-allocated when needed (e.g., when the resolution changes).

This buffer is allocated by the NvFBC library to the proper size. The size of the differential map is returned in diffMapSize.

This option is not compatible with the NVFBC_BUFFER_FORMAT_YUV420P and [NVFBC_BUFFER_FORMAT_YUV444P](#) buffer formats.

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

- [NvFBC.h](#)

8.25 NVFBC_API_FUNCTION_LIST Struct Reference

Structure populated with API function pointers.

```
#include <NvFBC.h>
```

Public Attributes

- `uint32_t dwVersion`
[in] Must be set to NVFBC_VERSION.
- `PNVFBCGETLASTERRORSTR nvFBCGetLastErrorStr`
[out] Pointer to [NvFBCGetLastErrorStr\(\)](#).
- `PNVFBCCREATEHANDLE nvFBCCreateHandle`
[out] Pointer to [NvFBCCreateHandle\(\)](#).
- `PNVFBCDESTROYHANDLE nvFBCDestroyHandle`
[out] Pointer to [NvFBCDestroyHandle\(\)](#).
- `PNVFBCGETSTATUS nvFBCGetStatus`
[out] Pointer to [NvFBCGetStatus\(\)](#).
- `PNVFBCCREATECAPTURESESSION nvFBCCreateCaptureSession`
[out] Pointer to [NvFBCCreateCaptureSession\(\)](#).
- `PNVFBCDESTROYCAPTURESESSION nvFBCDestroyCaptureSession`
[out] Pointer to [NvFBCDestroyCaptureSession\(\)](#).
- `PNVFBCOTOSYSSETUP nvFBCToSysSetUp`
[out] Pointer to [NvFBCToSysSetUp\(\)](#).
- `PNVFBCOTOSYSGRABFRAME nvFBCToSysGrabFrame`
[out] Pointer to [NvFBCToSysGrabFrame\(\)](#).
- `PNVFBCTOCUDASETUP nvFBCToCudaSetUp`
[out] Pointer to [NvFBCToCudaSetUp\(\)](#).
- `PNVFBCTOCUDAGRABFRAME nvFBCToCudaGrabFrame`
[out] Pointer to [NvFBCToCudaGrabFrame\(\)](#).
- `PNVFBCTOH264SETUP nvFBCToH264SetUp`
[out] Pointer to [NvFBCToH264SetUp\(\)](#).
- `PNVFBCTOH264GRABFRAME nvFBCToH264GrabFrame`
[out] Pointer to [NvFBCToH264GrabFrame\(\)](#).
- `PNVFBCTOH264GETHEADER nvFBCToH264GetHeader`
[out] Pointer to [NvFBCToH264GetHeader\(\)](#).

- PNVFBCBINDCONTEXT [nvFBCBindContext](#)
[out] Pointer to [NvFBCBindContext\(\)](#).
- PNVFBCRELEASECONTEXT [nvFBCReleaseContext](#)
[out] Pointer to [NvFBCReleaseContext\(\)](#).
- PNVFBCTOHWENCSETUP [nvFBCToHwEncSetUp](#)
[out] Pointer to [NvFBCToHwEncSetUp\(\)](#).
- PNVFBCTOHWENCGRABFRAME [nvFBCToHwEncGrabFrame](#)
[out] Pointer to [NvFBCToHwEncGrabFrame\(\)](#).
- PNVFBCTOHWENCGETHEADER [nvFBCToHwEncGetHeader](#)
[out] Pointer to [NvFBCToHwEncGetHeader\(\)](#).
- PNVFBCTOHWENCGETCAPS [nvFBCToHwEncGetCaps](#)
[out] Pointer to [nvFBCToHwEncGetCaps\(\)](#).
- PNVFBCTOGLSETUP [nvFBCToGLSetUp](#)
[out] Pointer to [nvFBCToGLSetUp\(\)](#).
- PNVFBCTOGLGRABFRAME [nvFBCToGLGrabFrame](#)
[out] Pointer to [nvFBCToGLGrabFrame\(\)](#).

8.25.1 Detailed Description

Structure populated with API function pointers.

8.25.2 Member Data Documentation

8.25.2.1 `uint32_t NVFBC_API_FUNCTION_LIST::dwVersion`

[in] Must be set to `NVFBC_VERSION`.

8.25.2.2 `PNVFBCBINDCONTEXT NVFBC_API_FUNCTION_LIST::nvFBCBindContext`

[out] Pointer to [NvFBCBindContext\(\)](#).

8.25.2.3 `PNVFBCCREATECAPTURESESSION NVFBC_API_FUNCTION_LIST::nvFBCCreateCaptureSession`

[out] Pointer to [NvFBCCreateCaptureSession\(\)](#).

8.25.2.4 `PNVFBCCREATEHANDLE NVFBC_API_FUNCTION_LIST::nvFBCCreateHandle`

[out] Pointer to [NvFBCCreateHandle\(\)](#).

8.25.2.5 PNVFBCDESTROYCAPTURESESSION NVFBC_API_FUNCTION_LIST::nvFBCDestroyCaptureSession

[out] Pointer to [NvFBCDestroyCaptureSession\(\)](#).

8.25.2.6 PNVFBCDESTROYHANDLE NVFBC_API_FUNCTION_LIST::nvFBCDestroyHandle

[out] Pointer to [NvFBCDestroyHandle\(\)](#).

8.25.2.7 PNVFBCGETLASTERRORSTR NVFBC_API_FUNCTION_LIST::nvFBCGetLastErrorStr

[out] Pointer to [NvFBCGetLastErrorStr\(\)](#).

8.25.2.8 PNVFBCGETSTATUS NVFBC_API_FUNCTION_LIST::nvFBCGetStatus

[out] Pointer to [NvFBCGetStatus\(\)](#).

8.25.2.9 PNVFBCRELEASECONTEXT NVFBC_API_FUNCTION_LIST::nvFBCReleaseContext

[out] Pointer to [NvFBCReleaseContext\(\)](#).

8.25.2.10 PNVFBCTOCUDAGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToCudaGrabFrame

[out] Pointer to [NvFBCToCudaGrabFrame\(\)](#).

8.25.2.11 PNVFBCTOCUDASETUP NVFBC_API_FUNCTION_LIST::nvFBCToCudaSetUp

[out] Pointer to [NvFBCToCudaSetUp\(\)](#).

8.25.2.12 PNVFBCTOGLGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToGLGrabFrame

[out] Pointer to [nvFBCToGLGrabFrame\(\)](#).

8.25.2.13 PNVFBCTOGLSETUP NVFBC_API_FUNCTION_LIST::nvFBCToGLSetUp

[out] Pointer to [nvFBCToGLSetup\(\)](#).

8.25.2.14 PNVFBCTOH264GETHEADER NVFBC_API_FUNCTION_LIST::nvFBCToH264GetHeader

[out] Pointer to [NvFBCToH264GetHeader\(\)](#).

8.25.2.15 PNVFBCTOH264GRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToH264GrabFrame

[out] Pointer to [NvFBCToH264GrabFrame\(\)](#).

8.25.2.16 PNVFBCTOH264SETUP NVFBC_API_FUNCTION_LIST::nvFBCToH264Setup

[out] Pointer to [NvFBCToH264Setup\(\)](#).

8.25.2.17 PNVFBCTOHWENCGETCAPS NVFBC_API_FUNCTION_LIST::nvFBCToHwEncGetCaps

[out] Pointer to [nvFBCToHwEncGetCaps\(\)](#).

8.25.2.18 PNVFBCTOHWENCGETHEADER NVFBC_API_FUNCTION_LIST::nvFBCToHwEncGetHeader

[out] Pointer to [NvFBCToHwEncGetHeader\(\)](#).

8.25.2.19 PNVFBCTOHWENCGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToHwEncGrabFrame

[out] Pointer to [NvFBCToHwEncGrabFrame\(\)](#).

8.25.2.20 PNVFBCTOHWENCSETUP NVFBC_API_FUNCTION_LIST::nvFBCToHwEncSetup

[out] Pointer to [NvFBCToHwEncSetup\(\)](#).

8.25.2.21 PNVFBCTOSYSGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToSysGrabFrame

[out] Pointer to [NvFBCToSysGrabFrame\(\)](#).

8.25.2.22 PNVFBCTOSYSSETUP NVFBC_API_FUNCTION_LIST::nvFBCToSysSetup

[out] Pointer to [NvFBCToSysSetup\(\)](#).

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

- [NvFBC.h](#)

Chapter 9

File Documentation

9.1 NvFBC.h File Reference

This file contains the interface constants, structure definitions and function prototypes defining the NvFBC API for Linux.

```
#include <stdint.h>
```

Classes

- struct [_NVFBC_BOX](#)
Box used to describe an area of the tracked region to capture.
- struct [_NVFBC_SIZE](#)
Size used to describe the size of a frame.
- struct [_NVFBC_FRAME_GRAB_INFO](#)
Describes information about a captured frame.
- struct [_NVFBC_CREATE_HANDLE_PARAMS](#)
Defines parameters for the `CreateHandle()` API call.
- struct [_NVFBC_DESTROY_HANDLE_PARAMS](#)
Defines parameters for the `NvFBCDestroyHandle()` API call.
- struct [_NVFBC_OUTPUT](#)
Describes an RandR output.
- struct [_NVFBC_GET_STATUS_PARAMS](#)
Defines parameters for the `NvFBCGetStatus()` API call.
- struct [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS](#)
Defines parameters for the `NvFBCCreateCaptureSession()` API call.
- struct [_NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#)
Defines parameters for the `NvFBCDestroyCaptureSession()` API call.

- struct [_NVFBC_BIND_CONTEXT_PARAMS](#)
Defines parameters for the `NvFBCBindContext()` API call.
- struct [_NVFBC_RELEASE_CONTEXT_PARAMS](#)
Defines parameters for the `NvFBCReleaseContext()` API call.
- struct [_NVFBC_TOSYS_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToSysSetUp()` API call.
- struct [_NVFBC_TOSYS_GRAB_FRAME_PARAMS](#)
Defines parameters for the `NvFBCToSysGrabFrame()` API call.
- struct [_NVFBC_TOCUDA_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToCudaSetUp()` API call.
- struct [_NVFBC_TOCUDA_GRAB_FRAME_PARAMS](#)
Defines parameters for the `NvFBCToCudaGrabFrame()` API call.
- struct [_NVFBC_TOGL_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToGLSetUp()` API call.
- struct [_NVFBC_TOGL_GRAB_FRAME_PARAMS](#)
Defines parameters for the `NvFBCToGLGrabFrame()` API call.
- struct [_NVFBC_HWENC_CONFIG](#)
- struct [_NVFBC_HWENC_ENCODE_PARAMS](#)
- struct [_NVFBC_HWENC_FRAME_INFO](#)
- struct [_NVFBC_TOHWENC_GET_CAPS_PARAMS](#)
- struct [_NVFBC_TOHWENC_SETUP_PARAMS](#)
- struct [_NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#)
- struct [_NVFBC_TOHWENC_GET_HEADER_PARAMS](#)
- struct [NVFBC_API_FUNCTION_LIST](#)
Structure populated with API function pointers.

Defines

- #define [NVFBCAPI](#)
Calling convention.
- #define [NVFBC_VERSION_MAJOR](#) 1
NvFBC API major version.
- #define [NVFBC_VERSION_MINOR](#) 6
NvFBC API minor version.
- #define [NVFBC_VERSION](#) (uint32_t) (NVFBC_VERSION_MINOR | (NVFBC_VERSION_MAJOR << 8))
NvFBC API version.

- #define `NVFBC_STRUCT_VERSION`(typeName, ver) (uint32_t) (sizeof(typeName) | ((ver) << 16) | (NVFBC_VERSION << 24))
Creates a version number for structure parameters.
- #define `NVFBC_ERR_STR_LEN` 512
Maximum size in bytes of an error string.
- #define `NVFBC_CREATE_HANDLE_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_CREATE_HANDLE_PARAMS, 2)`
NVFBC_CREATE_HANDLE_PARAMS structure version.
- #define `NVFBC_DESTROY_HANDLE_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_DESTROY_HANDLE_PARAMS, 1)`
NVFBC_DESTROY_HANDLE_PARAMS structure version.
- #define `NVFBC_OUTPUT_MAX` 5
Maximum number of connected RandR outputs to an X screen.
- #define `NVFBC_OUTPUT_NAME_LEN` 128
Maximum size in bytes of an RandR output name.
- #define `NVFBC_GET_STATUS_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_GET_STATUS_PARAMS, 1)`
NVFBC_GET_STATUS_PARAMS structure version.
- #define `NVFBC_CREATE_CAPTURE_SESSION_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 5)`
NVFBC_CREATE_CAPTURE_SESSION_PARAMS structure version.
- #define `NVFBC_DESTROY_CAPTURE_SESSION_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_DESTROY_CAPTURE_SESSION_PARAMS, 1)`
NVFBC_DESTROY_CAPTURE_SESSION_PARAMS structure version.
- #define `NVFBC_BIND_CONTEXT_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_BIND_CONTEXT_PARAMS, 1)`
NVFBC_BIND_CONTEXT_PARAMS structure version.
- #define `NVFBC_RELEASE_CONTEXT_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_RELEASE_CONTEXT_PARAMS, 1)`
NVFBC_RELEASE_CONTEXT_PARAMS structure version.
- #define `NVFBC_TOSYS_SETUP_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_TOSYS_SETUP_PARAMS, 3)`
NVFBC_TOSYS_SETUP_PARAMS structure version.
- #define `NVFBC_TOSYS_GRAB_FRAME_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_TOSYS_GRAB_FRAME_PARAMS, 2)`
NVFBC_TOSYS_GRAB_FRAME_PARAMS structure version.

- `#define NVFBC_TOCUDA_SETUP_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOCUDA_SETUP_PARAMS, 1)`
NVFBC_TOCUDA_SETUP_PARAMS structure version.
- `#define NVFBC_TOCUDA_GRAB_FRAME_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOCUDA_GRAB_FRAME_PARAMS, 2)`
NVFBC_TOCUDA_GRAB_FRAME_PARAMS structure version.
- `#define NVFBC_TOGL_TEXTURES_MAX 2`
Maximum number of GL textures that can be used to store frames.
- `#define NVFBC_TOGL_SETUP_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOGL_SETUP_PARAMS, 2)`
NVFBC_TOGL_SETUP_PARAMS structure version.
- `#define NVFBC_TOGL_GRAB_FRAME_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOGL_GRAB_FRAME_PARAMS, 2)`
NVFBC_TOGL_GRAB_FRAME_PARAMS structure version.
- `#define NVFBC_MAX_REF_FRAMES 0x10`
- `#define NVFBC_HWENC_CONFIG_VER NVFBC_STRUCT_VERSION(NVFBC_HWENC_CONFIG, 5)`
- `#define NVFBC_HWENC_ENCODE_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_HWENC_ENCODE_PARAMS, 1)`
- `#define NVFBC_HWENC_FRAME_INFO_VER NVFBC_STRUCT_VERSION(NVFBC_HWENC_FRAME_INFO, 1)`
- `#define NVFBC_TOHWENC_GET_CAPS_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GET_CAPS_PARAMS, 1)`
- `#define NVFBC_TOHWENC_SETUP_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_SETUP_PARAMS, 1)`
- `#define NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 2)`
- `#define NVFBC_TOHWENC_GET_HEADER_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_TOHWENC_GET_HEADER_PARAMS, 1)`
- `#define NVFBC_CAPTURE_TO_H264_HW_ENCODER NVFBC_CAPTURE_TO_HW_ENCODER`
- `#define NVFBC_TOH264_GRAB_FLAGS_NOFLAGS NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS`
- `#define NVFBC_TOH264_GRAB_FLAGS_NOWAIT NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT`
- `#define NVFBC_TOH264_GRAB_FLAGS NVFBC_TOHWENC_GRAB_FLAGS`
- `#define NVFBC_H264_PRESET_LOW_LATENCY_HP NVFBC_HWENC_PRESET_LOW_LATENCY_HP`
- `#define NVFBC_H264_PRESET_LOW_LATENCY_HQ NVFBC_HWENC_PRESET_LOW_LATENCY_HQ`
- `#define NVFBC_H264_PRESET_LOW_LATENCY_DEFAULT NVFBC_HWENC_PRESET_LOW_LATENCY_DEFAULT`
- `#define NVFBC_H264_PRESET_LOSSLESS_HP NVFBC_HWENC_PRESET_LOSSLESS_HP`
- `#define NVFBC_H264_PRESET NVFBC_HWENC_PRESET`
- `#define NVFBC_H264_ENC_PARAMS_RC_CONSTQP NVFBC_HWENC_PARAMS_RC_CONSTQP`
- `#define NVFBC_H264_ENC_PARAMS_RC_VBR NVFBC_HWENC_PARAMS_RC_VBR`
- `#define NVFBC_H264_ENC_PARAMS_RC_CBR NVFBC_HWENC_PARAMS_RC_CBR`
- `#define NVFBC_H264_ENC_PARAMS_RC_2_PASS_QUALITY NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY`

- #define `NVFBC_H264_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP` `NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP`
- #define `NVFBC_H264_RATE_CONTROL_CBR_IFRAME_2_PASS` `NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_PASS`
- #define `NVFBC_H264_ENC_PARAMS_RC_MODE` `NVFBC_HWENC_PARAMS_RC_MODE`
- #define `NVFBC_H264_ENC_PARAM_FLAG_FORCEIDR` `NVFBC_HWENC_PARAM_FLAG_FORCEIDR`
- #define `NVFBC_H264_ENC_PARAM_FLAG_DYN_BITRATE_CHANGE` `NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE`
- #define `NVFBC_H264_ENC_PARAM_FLAGS` `NVFBC_HWENC_PARAM_FLAGS`
- #define `NVFBC_H264_ENC_SLICING_MODE_DISABLED` `NVFBC_HWENC_SLICING_MODE_DISABLED`
- #define `NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_MBS` `NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MBS`
- #define `NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_BYTES` `NVFBC_HWENC_SLICING_MODE_FIXED_NUM_BYTES`
- #define `NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_MB_ROWS` `NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MB_ROWS`
- #define `NVFBC_H264_ENC_SLICING_MODE_FIXED_NUM_SLICES` `NVFBC_HWENC_SLICING_MODE_FIXED_NUM_SLICES`
- #define `NVFBC_H264_ENC_SLICING_MODE` `NVFBC_HWENC_SLICING_MODE`
- #define `NVFBC_H264_HW_ENC_CONFIG` `NVFBC_HWENC_CONFIG`
- #define `NVFBC_H264_HW_ENC_CONFIG_VER` `NVFBC_STRUCT_VERSION(NVFBC_H264_HW_ENC_CONFIG, 4)`
- #define `NVFBC_H264_HW_ENC_ENCODE_PARAMS` `NVFBC_HWENC_ENCODE_PARAMS`
- #define `NVFBC_H264_HW_ENC_ENCODE_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_H264_HW_ENC_ENCODE_PARAMS, 1)`
- #define `NVFBC_H264_HW_ENC_FRAME_INFO` `NVFBC_HWENC_FRAME_INFO`
- #define `NVFBC_H264_HW_ENC_FRAME_INFO_VER` `NVFBC_STRUCT_VERSION(NVFBC_H264_HW_ENC_FRAME_INFO, 1)`
- #define `NVFBC_TOH264_SETUP_PARAMS` `NVFBC_TOHWENC_SETUP_PARAMS`
- #define `NVFBC_TOH264_SETUP_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_TOH264_SETUP_PARAMS, 1)`
- #define `NVFBC_TOH264_GRAB_FRAME_PARAMS` `NVFBC_TOHWENC_GRAB_FRAME_PARAMS`
- #define `NVFBC_TOH264_GRAB_FRAME_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_TOH264_GRAB_FRAME_PARAMS, 2)`
- #define `NVFBC_TOH264_GET_HEADER_PARAMS` `NVFBC_TOHWENC_GET_HEADER_PARAMS`
- #define `NVFBC_TOH264_GET_HEADER_PARAMS_VER` `NVFBC_STRUCT_VERSION(NVFBC_TOH264_GET_HEADER_PARAMS, 1)`
- #define `NVFBC_BUFFER_FORMAT_YUV420P` `NVFBC_BUFFER_FORMAT_NV12`

Typedefs

- typedef enum `_NVFBCSTATUS` `NVFBCSTATUS`
Defines error codes.
- typedef enum `_NVFBC_BOOL` `NVFBC_BOOL`
Defines boolean values.
- typedef enum `_NVFBC_CAPTURE_TYPE` `NVFBC_CAPTURE_TYPE`
Capture type.

- typedef enum [_NVFBC_BUFFER_FORMAT](#) [NVFBC_BUFFER_FORMAT](#)
Buffer format.
- typedef uint64_t [NVFBC_SESSION_HANDLE](#)
Handle used to identify an NvFBC session.
- typedef struct [_NVFBC_BOX](#) [NVFBC_BOX](#)
Box used to describe an area of the tracked region to capture.
- typedef struct [_NVFBC_SIZE](#) [NVFBC_SIZE](#)
Size used to describe the size of a frame.
- typedef struct [_NVFBC_FRAME_GRAB_INFO](#) [NVFBC_FRAME_GRAB_INFO](#)
Describes information about a captured frame.
- typedef struct [_NVFBC_CREATE_HANDLE_PARAMS](#) [NVFBC_CREATE_HANDLE_PARAMS](#)
Defines parameters for the `CreateHandle()` API call.
- typedef struct [_NVFBC_DESTROY_HANDLE_PARAMS](#) [NVFBC_DESTROY_HANDLE_PARAMS](#)
Defines parameters for the `NvFBCDestroyHandle()` API call.
- typedef struct [_NVFBC_OUTPUT](#) [NVFBC_RANDR_OUTPUT_INFO](#)
Describes an RandR output.
- typedef struct [_NVFBC_GET_STATUS_PARAMS](#) [NVFBC_GET_STATUS_PARAMS](#)
Defines parameters for the `NvFBCGetStatus()` API call.
- typedef struct [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS](#) [NVFBC_CREATE_CAPTURE_SESSION_PARAMS](#)
Defines parameters for the `NvFBCCreateCaptureSession()` API call.
- typedef struct [_NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#) [NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#)
Defines parameters for the `NvFBCDestroyCaptureSession()` API call.
- typedef struct [_NVFBC_BIND_CONTEXT_PARAMS](#) [NVFBC_BIND_CONTEXT_PARAMS](#)
Defines parameters for the `NvFBCBindContext()` API call.
- typedef struct [_NVFBC_RELEASE_CONTEXT_PARAMS](#) [NVFBC_RELEASE_CONTEXT_PARAMS](#)
Defines parameters for the `NvFBCReleaseContext()` API call.
- typedef struct [_NVFBC_TOSYS_SETUP_PARAMS](#) [NVFBC_TOSYS_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToSysSetUp()` API call.
- typedef struct [_NVFBC_TOSYS_GRAB_FRAME_PARAMS](#) [NVFBC_TOSYS_GRAB_FRAME_PARAMS](#)
Defines parameters for the `NvFBCToSysGrabFrame()` API call.
- typedef struct [_NVFBC_TOCUDA_SETUP_PARAMS](#) [NVFBC_TOCUDA_SETUP_PARAMS](#)
Defines parameters for the `NvFBCToCudaSetUp()` API call.

- typedef struct `_NVFBC_TOCUDA_GRAB_FRAME_PARAMS` `NVFBC_TOCUDA_GRAB_FRAME_PARAMS`
Defines parameters for the `NvFBCToCudaGrabFrame()` API call.
- typedef struct `_NVFBC_TOGL_SETUP_PARAMS` `NVFBC_TOGL_SETUP_PARAMS`
Defines parameters for the `NvFBCToGLSetUp()` API call.
- typedef struct `_NVFBC_TOGL_GRAB_FRAME_PARAMS` `NVFBC_TOGL_GRAB_FRAME_PARAMS`
Defines parameters for the `NvFBCToGLGrabFrame()` API call.
- typedef enum `_NVFBC_HWENC_PARAMS_RC_MODE` `NVFBC_HWENC_PARAMS_RC_MODE`
- typedef struct `_NVFBC_HWENC_CONFIG` `NVFBC_HWENC_CONFIG`
- typedef struct `_NVFBC_HWENC_ENCODE_PARAMS` `NVFBC_HWENC_ENCODE_PARAMS`
- typedef struct `_NVFBC_HWENC_FRAME_INFO` `NVFBC_HWENC_FRAME_INFO`
- typedef struct `_NVFBC_TOHWENC_GET_CAPS_PARAMS` `NVFBC_TOHWENC_GET_CAPS_PARAMS`
- typedef struct `_NVFBC_TOHWENC_SETUP_PARAMS` `NVFBC_TOHWENC_SETUP_PARAMS`
- typedef struct `_NVFBC_TOHWENC_GRAB_FRAME_PARAMS` `NVFBC_TOHWENC_GRAB_FRAME_PARAMS`
- typedef struct `_NVFBC_TOHWENC_GET_HEADER_PARAMS` `NVFBC_TOHWENC_GET_HEADER_PARAMS`
- typedef `NVFBCSTATUS(NVFBCAPI * PNVFBCCREATEINSTANCE)(NVFBC_API_FUNCTION_LIST *pFunctionList)`
Defines function pointer for the `NvFBCCreateInstance()` API call.

Enumerations

- enum `_NVFBCSTATUS` {
`NVFBC_SUCCESS` = 0, `NVFBC_ERR_API_VERSION` = 1, `NVFBC_ERR_INTERNAL` = 2, `NVFBC_ERR_INVALID_PARAM` = 3,
`NVFBC_ERR_INVALID_PTR` = 4, `NVFBC_ERR_INVALID_HANDLE` = 5, `NVFBC_ERR_MAX_CLIENTS` = 6, `NVFBC_ERR_UNSUPPORTED` = 7,
`NVFBC_ERR_OUT_OF_MEMORY` = 8, `NVFBC_ERR_BAD_REQUEST` = 9, `NVFBC_ERR_X` = 10, `NVFBC_ERR_GLX` = 11,
`NVFBC_ERR_GL` = 12, `NVFBC_ERR_CUDA` = 13, `NVFBC_ERR_ENCODER` = 14, `NVFBC_ERR_CONTEXT` = 15,
`NVFBC_ERR_MUST_RECREATE` = 16 }
Defines error codes.
- enum `_NVFBC_BOOL` { `NVFBC_FALSE` = 0, `NVFBC_TRUE` }
Defines boolean values.
- enum `_NVFBC_CAPTURE_TYPE` { `NVFBC_CAPTURE_TO_SYS` = 0, `NVFBC_CAPTURE_SHARED_CUDA`, `NVFBC_CAPTURE_TO_HW_ENCODER`, `NVFBC_CAPTURE_TO_GL` }
Capture type.
- enum `NVFBC_TRACKING_TYPE` { `NVFBC_TRACKING_DEFAULT` = 0, `NVFBC_TRACKING_OUTPUT`, `NVFBC_TRACKING_SCREEN` }

Tracking type.

- enum `_NVFBC_BUFFER_FORMAT` {
`NVFBC_BUFFER_FORMAT_ARGB = 0, NVFBC_BUFFER_FORMAT_RGB, NVFBC_BUFFER_FORMAT_NV12, NVFBC_BUFFER_FORMAT_YUV444P,`
`NVFBC_BUFFER_FORMAT_RGBA, NVFBC_BUFFER_FORMAT_BGRA` }

Buffer format.

- enum `NVFBC_TOSYS_GRAB_FLAGS` { `NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS = 0, NVFBC_TOSYS_GRAB_FLAGS_NOWAIT = (1 << 0), NVFBC_TOSYS_GRAB_FLAGS_FORCE_REFRESH = (1 << 1), NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY = (1 << 2)` }

Defines flags that can be used when capturing to system memory.

- enum `NVFBC_TOCUDA_FLAGS` { `NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS = 0, NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT = (1 << 0), NVFBC_TOCUDA_GRAB_FLAGS_FORCE_REFRESH = (1 << 1), NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY = (1 << 2)` }

Defines flags that can be used when capturing to a CUDA buffer in video memory.

- enum `NVFBC_TOGL_FLAGS` { `NVFBC_TOGL_GRAB_FLAGS_NOFLAGS = 0, NVFBC_TOGL_GRAB_FLAGS_NOWAIT = (1 << 0), NVFBC_TOGL_GRAB_FLAGS_FORCE_REFRESH = (1 << 1), NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY = (1 << 2)` }

Defines flags that can be used when capturing to an OpenGL buffer in video memory.

- enum `NVFBC_TOHWENC_GRAB_FLAGS` { `NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS = 0, NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT = (1 << 0)` }
- enum `NVFBC_HWENC_PRESET` { `NVFBC_HWENC_PRESET_LOW_LATENCY_HP = 0, NVFBC_HWENC_PRESET_LOW_LATENCY_HQ, NVFBC_HWENC_PRESET_LOW_LATENCY_DEFAULT, NVFBC_HWENC_PRESET_LOSSLESS_HP` }
- enum `_NVFBC_HWENC_PARAMS_RC_MODE` {
`NVFBC_HWENC_PARAMS_RC_CONSTQP = 0, NVFBC_HWENC_PARAMS_RC_VBR, NVFBC_HWENC_PARAMS_RC_CBR, NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY,`
`NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP, NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_PASS` }
- enum `NVFBC_HWENC_PARAM_FLAGS` { `NVFBC_HWENC_PARAM_FLAG_FORCEIDR = (1 << 0), NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE = (1 << 1)` }
- enum `NVFBC_HWENC_SLICING_MODE` {
`NVFBC_HWENC_SLICING_MODE_DISABLED = 0, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MBS, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_BYTES, NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MB_ROWS,`
`NVFBC_HWENC_SLICING_MODE_FIXED_NUM_SLICES` }
- enum `NVFBC_HWENC_CODEC` { `NVFBC_HWENC_CODEC_H264 = 0, NVFBC_HWENC_CODEC_HEVC` }

Functions

- const char *NVFBCAPI `NvFBCGetLastErrorStr` (const `NVFBC_SESSION_HANDLE` sessionHandle)
Gets the last error message that got recorded for a client.
- `NVFBCSTATUS` NVFBCAPI `NvFBCCreateHandle` (`NVFBC_SESSION_HANDLE` *pSessionHandle, `NVFBC_CREATE_HANDLE_PARAMS` *pParams)

Allocates a new handle for an NvFBC client.

- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCDestroyHandle](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_DESTROY_HANDLE_PARAMS](#) *pParams)
Destroys the handle of an NvFBC client.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCGetStatus](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_GET_STATUS_PARAMS](#) *pParams)
Gets the current status of the display driver.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCBindContext](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_BIND_CONTEXT_PARAMS](#) *pParams)
Binds the FBC context to the calling thread.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCReleaseContext](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_RELEASE_CONTEXT_PARAMS](#) *pParams)
Releases the FBC context from the calling thread.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCCreateCaptureSession](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_CREATE_CAPTURE_SESSION_PARAMS](#) *pParams)
Creates a capture session for an FBC client.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCDestroyCaptureSession](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_DESTROY_CAPTURE_SESSION_PARAMS](#) *pParams)
Destroys a capture session for an FBC client.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToSysSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOSYS_SETUP_PARAMS](#) *pParams)
Sets up a capture to system memory session.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToSysGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOSYS_GRAB_FRAME_PARAMS](#) *pParams)
Captures a frame to a buffer in system memory.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToCudaSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOCUDA_SETUP_PARAMS](#) *pParams)
Sets up a capture to video memory session.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToCudaGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOCUDA_GRAB_FRAME_PARAMS](#) *pParams)
Captures a frame to a CUDA device in video memory.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToGLSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOGL_SETUP_PARAMS](#) *pParams)
Sets up a capture to OpenGL buffer in video memory session.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToGLGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOGL_GRAB_FRAME_PARAMS](#) *pParams)
Captures a frame to an OpenGL buffer in video memory.
- [NVFBCSTATUS](#) [NVFBCAPI](#) [NvFBCToH264SetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOH264_SETUP_PARAMS](#) *pParams)

Sets up a capture to H.264 compressed frames in system memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToH264GrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOH264_GRAB_FRAME_PARAMS](#) *pParams)

Captures a H.264 compressed frame to a bitstream in system memory.

- [NVFBCSTATUS NVFBCAPI NvFBCToH264GetHeader](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOH264_GET_HEADER_PARAMS](#) *pParams)

Gets SPS/PPS headers.

- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncGetCaps](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_GET_CAPS_PARAMS](#) *pParams)
- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncSetUp](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_SETUP_PARAMS](#) *pParams)
- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncGrabFrame](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_GRAB_FRAME_PARAMS](#) *pParams)
- [NVFBCSTATUS NVFBCAPI NvFBCToHwEncGetHeader](#) (const [NVFBC_SESSION_HANDLE](#) sessionHandle, [NVFBC_TOHWENC_GET_HEADER_PARAMS](#) *pParams)
- [NVFBCSTATUS NVFBCAPI NvFBCCreateInstance](#) ([NVFBC_API_FUNCTION_LIST](#) *pFunctionList)

Entry Points to the NvFBC interface.

9.1.1 Detailed Description

This file contains the interface constants, structure definitions and function prototypes defining the NvFBC API for Linux.

Copyright (c) 2013-2018, NVIDIA CORPORATION. All rights reserved.

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR 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 OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Index

- `_NVFBCSTATUS`
 - `FBC_STRUCT`, 24
- `_NVFBC_BIND_CONTEXT_PARAMS`, 49
- `_NVFBC_BOOL`
 - `FBC_STRUCT`, 23
- `_NVFBC_BOX`, 50
- `_NVFBC_BUFFER_FORMAT`
 - `FBC_STRUCT`, 23
- `_NVFBC_CAPTURE_TYPE`
 - `FBC_STRUCT`, 24
- `_NVFBC_CREATE_CAPTURE_SESSION_PARAMS`, 51
 - `bDisableAutoModesetRecovery`, 52
 - `bPushModel`, 52
 - `bRoundFrameSize`, 52
 - `bWithCursor`, 52
 - `captureBox`, 52
 - `dwSamplingRateMs`, 53
 - `eCaptureType`, 53
 - `frameSize`, 53
- `_NVFBC_CREATE_HANDLE_PARAMS`, 54
 - `bExternallyManagedContext`, 54
 - `glxCtx`, 54
 - `glxFBConfig`, 54
- `_NVFBC_DESTROY_CAPTURE_SESSION_PARAMS`, 56
- `_NVFBC_DESTROY_HANDLE_PARAMS`, 57
- `_NVFBC_FRAME_GRAB_INFO`, 58
 - `bIsNewFrame`, 58
 - `dwCurrentFrame`, 59
 - `ulTimestampUs`, 59
- `_NVFBC_GET_STATUS_PARAMS`, 60
 - `bXRandRAvailable`, 60
 - `dwOutputNum`, 60
 - `outputs`, 61
- `_NVFBC_HWENC_CONFIG`, 62
 - `bEnableAQ`, 63
 - `bEnableIntraRefresh`, 63
 - `bEnableMSE`, 63
 - `bOutBandSPSPPS`, 64
 - `dwAvgBitRate`, 64
 - `dwFrameRateDen`, 64
 - `dwFrameRateNum`, 64
 - `dwGOPLength`, 64
 - `dwMaxNumRefFrames`, 64
 - `dwProfile`, 64
 - `dwVBVBufferSize`, 65
 - `dwVBVInitialDelay`, 65
 - `eInputBufferFormat`, 65
- `_NVFBC_HWENC_ENCODE_PARAMS`, 66
 - `bInvalidateReferenceFrames`, 67
 - `bReEncodePrevFrame`, 67
 - `dwEncodeParamFlags`, 67
 - `dwNewVBVBufferSize`, 67
 - `dwNewVBVInitialDelay`, 67
- `_NVFBC_HWENC_FRAME_INFO`, 68
- `_NVFBC_HWENC_PARAMS_RC_MODE`
 - `FBC_DEPRECATED_STRUCT`, 32
- `_NVFBC_OUTPUT`, 69
 - `name`, 69
- `_NVFBC_RELEASE_CONTEXT_PARAMS`, 70
- `_NVFBC_SIZE`, 71
- `_NVFBC_TOCUDA_GRAB_FRAME_PARAMS`, 72
 - `dwTimeoutMs`, 72
 - `pCUDADeviceBuffer`, 72
 - `pFrameGrabInfo`, 73
- `_NVFBC_TOCUDA_SETUP_PARAMS`, 74
- `_NVFBC_TOGL_GRAB_FRAME_PARAMS`, 75
 - `dwTextureIndex`, 75
 - `dwTimeoutMs`, 75
 - `pFrameGrabInfo`, 76
- `_NVFBC_TOGL_SETUP_PARAMS`, 77
 - `diffMapSize`, 77
 - `dwDiffMapScalingFactor`, 77
 - `dwTextures`, 78
 - `ppDiffMap`, 78
- `_NVFBC_TOHWENC_GET_CAPS_PARAMS`, 79
 - `bCodecSupported`, 80
- `_NVFBC_TOHWENC_GET_HEADER_PARAMS`, 81
 - `pBuffer`, 81
- `_NVFBC_TOHWENC_GRAB_FRAME_PARAMS`, 82
 - `dwMSE`, 82
 - `pFrameGrabInfo`, 82
 - `ppBitStreamBuffer`, 82
- `_NVFBC_TOHWENC_SETUP_PARAMS`, 84
- `_NVFBC_TOSYS_GRAB_FRAME_PARAMS`, 85
 - `dwTimeoutMs`, 85
 - `pFrameGrabInfo`, 85
- `_NVFBC_TOSYS_SETUP_PARAMS`, 87
 - `diffMapSize`, 87

- dwDiffMapScalingFactor, [87](#)
- ppBuffer, [87](#)
- ppDiffMap, [88](#)
- API Entry Points, [35](#)
- bCodecSupported
 - [_NVFBC_TOHWENC_GET_CAPS_PARAMS, 80](#)
- bDisableAutoModesetRecovery
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 52](#)
- bEnableAQ
 - [_NVFBC_HWENC_CONFIG, 63](#)
- bEnableIntraRefresh
 - [_NVFBC_HWENC_CONFIG, 63](#)
- bEnableMSE
 - [_NVFBC_HWENC_CONFIG, 63](#)
- bExternallyManagedContext
 - [_NVFBC_CREATE_HANDLE_PARAMS, 54](#)
- bInvalidateReferenceFrames
 - [_NVFBC_HWENC_ENCODE_PARAMS, 67](#)
- bIsNewFrame
 - [_NVFBC_FRAME_GRAB_INFO, 58](#)
- bOutBandSPSPPS
 - [_NVFBC_HWENC_CONFIG, 64](#)
- bPushModel
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 52](#)
- bReEncodePrevFrame
 - [_NVFBC_HWENC_ENCODE_PARAMS, 67](#)
- bRoundFrameSize
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 52](#)
- bWithCursor
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 52](#)
- bXRandRAvailable
 - [_NVFBC_GET_STATUS_PARAMS, 60](#)
- captureBox
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 52](#)
- ChangeLog, [16](#)
- Deprecated Structure Definition, [28](#)
- diffMapSize
 - [_NVFBC_TOGL_SETUP_PARAMS, 77](#)
 - [_NVFBC_TOSYS_SETUP_PARAMS, 87](#)
- dwAvgBitRate
 - [_NVFBC_HWENC_CONFIG, 64](#)
- dwCurrentFrame
 - [_NVFBC_FRAME_GRAB_INFO, 59](#)
- dwDiffMapScalingFactor
 - [_NVFBC_TOGL_SETUP_PARAMS, 77](#)
 - [_NVFBC_TOSYS_SETUP_PARAMS, 87](#)
- dwEncodeParamFlags
 - [_NVFBC_HWENC_ENCODE_PARAMS, 67](#)
- dwFrameRateDen
 - [_NVFBC_HWENC_CONFIG, 64](#)
- dwFrameRateNum
 - [_NVFBC_HWENC_CONFIG, 64](#)
- dwGOPLength
 - [_NVFBC_HWENC_CONFIG, 64](#)
- dwMaxNumRefFrames
 - [_NVFBC_HWENC_CONFIG, 64](#)
- dwMSE
 - [_NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 82](#)
- dwNewVBVBufferSize
 - [_NVFBC_HWENC_ENCODE_PARAMS, 67](#)
- dwNewVBVInitialDelay
 - [_NVFBC_HWENC_ENCODE_PARAMS, 67](#)
- dwOutputNum
 - [_NVFBC_GET_STATUS_PARAMS, 60](#)
- dwProfile
 - [_NVFBC_HWENC_CONFIG, 64](#)
- dwSamplingRateMs
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 53](#)
- dwTextureIndex
 - [_NVFBC_TOGL_GRAB_FRAME_PARAMS, 75](#)
- dwTextures
 - [_NVFBC_TOGL_SETUP_PARAMS, 78](#)
- dwTimeoutMs
 - [_NVFBC_TOCUDA_GRAB_FRAME_PARAMS, 72](#)
 - [_NVFBC_TOGL_GRAB_FRAME_PARAMS, 75](#)
 - [_NVFBC_TOSYS_GRAB_FRAME_PARAMS, 85](#)
- dwVBVBufferSize
 - [_NVFBC_HWENC_CONFIG, 65](#)
- dwVBVInitialDelay
 - [_NVFBC_HWENC_CONFIG, 65](#)
- dwVersion
 - [NVFBC_API_FUNCTION_LIST, 90](#)
- eCaptureType
 - [_NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 53](#)
- eInputBufferFormat
 - [_NVFBC_HWENC_CONFIG, 65](#)
- FBC_DEPRECATED_STRUCT
 - [NVFBC_HWENC_CODEC_H264, 33](#)
 - [NVFBC_HWENC_CODEC_HEVC, 33](#)
 - [NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE, 33](#)
 - [NVFBC_HWENC_PARAM_FLAG_FORCEIDR, 33](#)
 - [NVFBC_HWENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP, 33](#)

- NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY, 33
- NVFBC_HWENC_PARAMS_RC_CBR, 33
- NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_PASS, 33
- NVFBC_HWENC_PARAMS_RC_CONSTQP, 32
- NVFBC_HWENC_PARAMS_RC_VBR, 32
- NVFBC_HWENC_PRESET_LOSSLESS_HP, 33
- NVFBC_HWENC_PRESET_LOW_LATENCY_DEFAULT, 33
- NVFBC_HWENC_PRESET_LOW_LATENCY_HP, 33
- NVFBC_HWENC_PRESET_LOW_LATENCY_HQ, 33
- NVFBC_HWENC_SLICING_MODE_DISABLED, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_BYTES, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MB_ROWS, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_MBS, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_SLICES, 34
- NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS, 34
- NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT, 34
- FBC_STRUCT
 - NVFBC_BUFFER_FORMAT_ARGB, 24
 - NVFBC_BUFFER_FORMAT_BGRA, 24
 - NVFBC_BUFFER_FORMAT_NV12, 24
 - NVFBC_BUFFER_FORMAT_RGB, 24
 - NVFBC_BUFFER_FORMAT_RGBA, 24
 - NVFBC_BUFFER_FORMAT_YUV444P, 24
 - NVFBC_CAPTURE_SHARED_CUDA, 24
 - NVFBC_CAPTURE_TO_GL, 24
 - NVFBC_CAPTURE_TO_HW_ENCODER, 24
 - NVFBC_CAPTURE_TO_SYS, 24
 - NVFBC_ERR_API_VERSION, 24
 - NVFBC_ERR_BAD_REQUEST, 25
 - NVFBC_ERR_CONTEXT, 25
 - NVFBC_ERR_CUDA, 25
 - NVFBC_ERR_ENCODER, 25
 - NVFBC_ERR_GL, 25
 - NVFBC_ERR_GLX, 25
 - NVFBC_ERR_INTERNAL, 24
 - NVFBC_ERR_INVALID_HANDLE, 25
 - NVFBC_ERR_INVALID_PARAM, 25
 - NVFBC_ERR_INVALID_PTR, 25
 - NVFBC_ERR_MAX_CLIENTS, 25
 - NVFBC_ERR_MUST_RECREATE, 25
 - NVFBC_ERR_OUT_OF_MEMORY, 25
 - NVFBC_ERR_UNSUPPORTED, 25
 - NVFBC_ERR_X, 25
 - NVFBC_FALSE, 23
 - NVFBC_SUCCESS, 24
 - NVFBC_TOCUDA_GRAB_FLAGS_FORCE_REFRESH, 26
 - NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS, 25
 - NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT, 25
 - NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY, 26
 - NVFBC_TOGL_GRAB_FLAGS_FORCE_REFRESH, 26
 - NVFBC_TOGL_GRAB_FLAGS_NOFLAGS, 26
 - NVFBC_TOGL_GRAB_FLAGS_NOWAIT, 26
 - NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY, 26
 - NVFBC_TOSYS_GRAB_FLAGS_FORCE_REFRESH, 27
 - NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS, 26
 - NVFBC_TOSYS_GRAB_FLAGS_NOWAIT, 27
 - NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_NEW_FRAME_READY, 27
 - NVFBC_TRACKING_DEFAULT, 27
 - NVFBC_TRACKING_OUTPUT, 27
 - NVFBC_TRACKING_SCREEN, 27
 - NVFBC_TRUE, 23
- FBC_DEPRECATED_STRUCT
 - _NVFBC_HWENC_PARAMS_RC_MODE, 32
 - NVFBC_HWENC_CODEC, 33
 - NVFBC_HWENC_CONFIG, 31
 - NVFBC_HWENC_CONFIG_VER, 30
 - NVFBC_HWENC_ENCODE_PARAMS, 31
 - NVFBC_HWENC_ENCODE_PARAMS_VER, 30
 - NVFBC_HWENC_FRAME_INFO, 31
 - NVFBC_HWENC_FRAME_INFO_VER, 30
 - NVFBC_HWENC_PARAM_FLAGS, 33
 - NVFBC_HWENC_PARAMS_RC_MODE, 32
 - NVFBC_HWENC_PRESET, 33
 - NVFBC_HWENC_SLICING_MODE, 33
 - NVFBC_MAX_REF_FRAMES, 31
 - NVFBC_TOHWENC_GET_CAPS_PARAMS, 32
 - NVFBC_TOHWENC_GET_CAPS_PARAMS_VER, 31
 - NVFBC_TOHWENC_GET_HEADER_PARAMS, 32
 - NVFBC_TOHWENC_GET_HEADER_PARAMS_VER, 31
 - NVFBC_TOHWENC_GRAB_FLAGS, 34
 - NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 32
 - NVFBC_TOHWENC_GRAB_FRAME_PARAMS_VER, 31
 - NVFBC_TOHWENC_SETUP_PARAMS, 32
 - NVFBC_TOHWENC_SETUP_PARAMS_VER, 31

- FBC_FUNC
 - NvFBCBindContext, 37
 - NvFBCCreateCaptureSession, 37
 - NvFBCCreateHandle, 38
 - NvFBCCreateInstance, 38
 - NvFBCDestroyCaptureSession, 39
 - NvFBCDestroyHandle, 39
 - NvFBCGetLastErrorStr, 40
 - NvFBCGetStatus, 40
 - NvFBCReleaseContext, 40
 - NvFBCToCudaGrabFrame, 41
 - NvFBCToCudaSetUp, 41
 - NvFBCToGLGrabFrame, 42
 - NvFBCToGLSetUp, 42
 - NvFBCToH264GetHeader, 43
 - NvFBCToH264GrabFrame, 43
 - NvFBCToH264SetUp, 44
 - NvFBCToHwEncGetCaps, 45
 - NvFBCToHwEncGetHeader, 45
 - NvFBCToHwEncGrabFrame, 46
 - NvFBCToHwEncSetUp, 46
 - NvFBCToSysGrabFrame, 47
 - NvFBCToSysSetUp, 48
- FBC_STRUCT
 - _NVFBCSTATUS, 24
 - _NVFBC_BOOL, 23
 - _NVFBC_BUFFER_FORMAT, 23
 - _NVFBC_CAPTURE_TYPE, 24
 - NVFBC_BOX, 23
 - NVFBC_RANDR_OUTPUT_INFO, 23
 - NVFBC_TOCUDA_FLAGS, 25
 - NVFBC_TOGL_FLAGS, 26
 - NVFBC_TOSYS_GRAB_FLAGS, 26
 - NVFBC_TRACKING_TYPE, 27
 - NVFBCSTATUS, 23
- frameSize
 - _NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 53
- glxCtx
 - _NVFBC_CREATE_HANDLE_PARAMS, 54
- glxFBConfig
 - _NVFBC_CREATE_HANDLE_PARAMS, 54
- name
 - _NVFBC_OUTPUT, 69
- NvFBC.h, 93
- NVFBC_BUFFER_FORMAT_ARGB
 - FBC_STRUCT, 24
- NVFBC_BUFFER_FORMAT_BGRA
 - FBC_STRUCT, 24
- NVFBC_BUFFER_FORMAT_NV12
 - FBC_STRUCT, 24
- NVFBC_BUFFER_FORMAT_RGB
 - FBC_STRUCT, 24
- NVFBC_BUFFER_FORMAT_RGBA
 - FBC_STRUCT, 24
- NVFBC_BUFFER_FORMAT_YUV444P
 - FBC_STRUCT, 24
- NVFBC_CAPTURE_SHARED_CUDA
 - FBC_STRUCT, 24
- NVFBC_CAPTURE_TO_GL
 - FBC_STRUCT, 24
- NVFBC_CAPTURE_TO_HW_ENCODER
 - FBC_STRUCT, 24
- NVFBC_CAPTURE_TO_SYS
 - FBC_STRUCT, 24
- NVFBC_ERR_API_VERSION
 - FBC_STRUCT, 24
- NVFBC_ERR_BAD_REQUEST
 - FBC_STRUCT, 25
- NVFBC_ERR_CONTEXT
 - FBC_STRUCT, 25
- NVFBC_ERR_CUDA
 - FBC_STRUCT, 25
- NVFBC_ERR_ENCODER
 - FBC_STRUCT, 25
- NVFBC_ERR_GL
 - FBC_STRUCT, 25
- NVFBC_ERR_GLX
 - FBC_STRUCT, 25
- NVFBC_ERR_INTERNAL
 - FBC_STRUCT, 24
- NVFBC_ERR_INVALID_HANDLE
 - FBC_STRUCT, 25
- NVFBC_ERR_INVALID_PARAM
 - FBC_STRUCT, 25
- NVFBC_ERR_INVALID_PTR
 - FBC_STRUCT, 25
- NVFBC_ERR_MAX_CLIENTS
 - FBC_STRUCT, 25
- NVFBC_ERR_MUST_RECREATE
 - FBC_STRUCT, 25
- NVFBC_ERR_OUT_OF_MEMORY
 - FBC_STRUCT, 25
- NVFBC_ERR_UNSUPPORTED
 - FBC_STRUCT, 25
- NVFBC_ERR_X
 - FBC_STRUCT, 25
- NVFBC_FALSE
 - FBC_STRUCT, 23
- NVFBC_HWENC_CODEC_H264
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_CODEC_HEVC
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAM_FLAG_DYN_BITRATE_CHANGE
 - FBC_DEPRECATED_STRUCT, 33

- NVFBC_HWENC_PARAM_FLAG_FORCEIDR
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAMS_RC_2_PASS_-
 - FRAMESIZE_CAP
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAMS_RC_2_PASS_QUALITY
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAMS_RC_CBR
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAMS_RC_CBR_IFRAME_2_-
 - PASS
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAMS_RC_CONSTQP
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_HWENC_PARAMS_RC_VBR
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_HWENC_PRESET_LOSSLESS_HP
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PRESET_LOW_LATENCY_-
 - DEFAULT
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PRESET_LOW_LATENCY_HP
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PRESET_LOW_LATENCY_HQ
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_SLICING_MODE_DISABLED
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_-
 - BYTES
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_-
 - MB_ROWS
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_-
 - MBS
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_HWENC_SLICING_MODE_FIXED_NUM_-
 - SLICES
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_SUCCESS
 - FBC_STRUCT, 24
- NVFBC_TOCUDA_GRAB_FLAGS_FORCE_-
 - REFRESH
 - FBC_STRUCT, 26
- NVFBC_TOCUDA_GRAB_FLAGS_NOFLAGS
 - FBC_STRUCT, 25
- NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT
 - FBC_STRUCT, 25
- NVFBC_TOCUDA_GRAB_FLAGS_NOWAIT_IF_-
 - NEW_FRAME_READY
 - FBC_STRUCT, 26
- NVFBC_TOGL_GRAB_FLAGS_FORCE_REFRESH
 - FBC_STRUCT, 26
- NVFBC_TOGL_GRAB_FLAGS_NOFLAGS
 - FBC_STRUCT, 26
- NVFBC_TOGL_GRAB_FLAGS_NOWAIT_IF_NEW_-
 - FRAME_READY
 - FBC_STRUCT, 26
- NVFBC_TOHWENC_GRAB_FLAGS_NOFLAGS
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_TOHWENC_GRAB_FLAGS_NOWAIT
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_TOSYS_GRAB_FLAGS_FORCE_REFRESH
 - FBC_STRUCT, 27
- NVFBC_TOSYS_GRAB_FLAGS_NOFLAGS
 - FBC_STRUCT, 26
- NVFBC_TOSYS_GRAB_FLAGS_NOWAIT
 - FBC_STRUCT, 27
- NVFBC_TOSYS_GRAB_FLAGS_NOWAIT_IF_-
 - NEW_FRAME_READY
 - FBC_STRUCT, 27
- NVFBC_TRACKING_DEFAULT
 - FBC_STRUCT, 27
- NVFBC_TRACKING_OUTPUT
 - FBC_STRUCT, 27
- NVFBC_TRACKING_SCREEN
 - FBC_STRUCT, 27
- NVFBC_TRUE
 - FBC_STRUCT, 23
- NVFBC_API_FUNCTION_LIST, 89
 - dwVersion, 90
 - nvFBCBindContext, 90
 - nvFBCCreateCaptureSession, 90
 - nvFBCCreateHandle, 90
 - nvFBCDestroyCaptureSession, 90
 - nvFBCDestroyHandle, 91
 - nvFBCGetLastErrorStr, 91
 - nvFBCGetStatus, 91
 - nvFBCReleaseContext, 91
 - nvFBCToCudaGrabFrame, 91
 - nvFBCToCudaSetUp, 91
 - nvFBCToGLGrabFrame, 91
 - nvFBCToGLSetUp, 91
 - nvFBCToH264GetHeader, 91
 - nvFBCToH264GrabFrame, 91
 - nvFBCToH264SetUp, 91
 - nvFBCToHwEncGetCaps, 92
 - nvFBCToHwEncGetHeader, 92
 - nvFBCToHwEncGrabFrame, 92
 - nvFBCToHwEncSetUp, 92
 - nvFBCToSysGrabFrame, 92
 - nvFBCToSysSetUp, 92
- NVFBC_BOX
 - FBC_STRUCT, 23
- NVFBC_HWENC_CODEC
 - FBC_DEPRECATED_STRUCT, 33

- NVFBC_HWENC_CONFIG
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_HWENC_CONFIG_VER
 - FBC_DEPRECATED_STRUCT, 30
- NVFBC_HWENC_ENCODE_PARAMS
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_HWENC_ENCODE_PARAMS_VER
 - FBC_DEPRECATED_STRUCT, 30
- NVFBC_HWENC_FRAME_INFO
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_HWENC_FRAME_INFO_VER
 - FBC_DEPRECATED_STRUCT, 30
- NVFBC_HWENC_PARAM_FLAGS
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_PARAMS_RC_MODE
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_HWENC_PRESET
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_HWENC_SLICING_MODE
 - FBC_DEPRECATED_STRUCT, 33
- NVFBC_MAX_REF_FRAMES
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_RANDR_OUTPUT_INFO
 - FBC_STRUCT, 23
- NVFBC_TOCUDA_FLAGS
 - FBC_STRUCT, 25
- NVFBC_TOGL_FLAGS
 - FBC_STRUCT, 26
- NVFBC_TOHWENC_GET_CAPS_PARAMS
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_TOHWENC_GET_CAPS_PARAMS_VER
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_TOHWENC_GET_HEADER_PARAMS
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_TOHWENC_GET_HEADER_PARAMS_VER
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_TOHWENC_GRAB_FLAGS
 - FBC_DEPRECATED_STRUCT, 34
- NVFBC_TOHWENC_GRAB_FRAME_PARAMS
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_TOHWENC_GRAB_FRAME_PARAMS_-
 - VER
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_TOHWENC_SETUP_PARAMS
 - FBC_DEPRECATED_STRUCT, 32
- NVFBC_TOHWENC_SETUP_PARAMS_VER
 - FBC_DEPRECATED_STRUCT, 31
- NVFBC_TOSYS_GRAB_FLAGS
 - FBC_STRUCT, 26
- NVFBC_TRACKING_TYPE
 - FBC_STRUCT, 27
- NvFBCBindContext
 - FBC_FUNC, 37
- nvFBCBindContext
 - NVFBC_API_FUNCTION_LIST, 90
- NvFBCCreateCaptureSession
 - FBC_FUNC, 37
- nvFBCCreateCaptureSession
 - NVFBC_API_FUNCTION_LIST, 90
- NvFBCCreateHandle
 - FBC_FUNC, 38
- nvFBCCreateHandle
 - NVFBC_API_FUNCTION_LIST, 90
- NvFBCCreateInstance
 - FBC_FUNC, 38
- NvFBCDestroyCaptureSession
 - FBC_FUNC, 39
- nvFBCDestroyCaptureSession
 - NVFBC_API_FUNCTION_LIST, 90
- NvFBCDestroyHandle
 - FBC_FUNC, 39
- nvFBCDestroyHandle
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCGetLastErrorStr
 - FBC_FUNC, 40
- nvFBCGetLastErrorStr
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCGetStatus
 - FBC_FUNC, 40
- nvFBCGetStatus
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCReleaseContext
 - FBC_FUNC, 40
- nvFBCReleaseContext
 - NVFBC_API_FUNCTION_LIST, 91
- NVFBCSTATUS
 - FBC_STRUCT, 23
- NvFBCToCudaGrabFrame
 - FBC_FUNC, 41
- nvFBCToCudaGrabFrame
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToCudaSetUp
 - FBC_FUNC, 41
- nvFBCToCudaSetUp
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToGLGrabFrame
 - FBC_FUNC, 42
- nvFBCToGLGrabFrame
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToGLSetUp
 - FBC_FUNC, 42
- nvFBCToGLSetUp
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToH264GetHeader
 - FBC_FUNC, 43
- nvFBCToH264GetHeader
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToH264GrabFrame

- FBC_FUNC, 43
- nvFBCToH264GrabFrame
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToH264SetUp
 - FBC_FUNC, 44
- nvFBCToH264SetUp
 - NVFBC_API_FUNCTION_LIST, 91
- NvFBCToHwEncGetCaps
 - FBC_FUNC, 45
- nvFBCToHwEncGetCaps
 - NVFBC_API_FUNCTION_LIST, 92
- NvFBCToHwEncGetHeader
 - FBC_FUNC, 45
- nvFBCToHwEncGetHeader
 - NVFBC_API_FUNCTION_LIST, 92
- NvFBCToHwEncGrabFrame
 - FBC_FUNC, 46
- nvFBCToHwEncGrabFrame
 - NVFBC_API_FUNCTION_LIST, 92
- NvFBCToHwEncSetUp
 - FBC_FUNC, 46
- nvFBCToHwEncSetUp
 - NVFBC_API_FUNCTION_LIST, 92
- NvFBCToSysGrabFrame
 - FBC_FUNC, 47
- nvFBCToSysGrabFrame
 - NVFBC_API_FUNCTION_LIST, 92
- NvFBCToSysSetUp
 - FBC_FUNC, 48
- nvFBCToSysSetUp
 - NVFBC_API_FUNCTION_LIST, 92
- outputs
 - _NVFBC_GET_STATUS_PARAMS, 61
- pBuffer
 - _NVFBC_TOHWENC_GET_HEADER_PARAMS, 81
- pCUDADeviceBuffer
 - _NVFBC_TOCUDA_GRAB_FRAME_PARAMS, 72
- pFrameGrabInfo
 - _NVFBC_TOCUDA_GRAB_FRAME_PARAMS, 73
 - _NVFBC_TOGL_GRAB_FRAME_PARAMS, 76
 - _NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 82
 - _NVFBC_TOSYS_GRAB_FRAME_PARAMS, 85
- ppBitStreamBuffer
 - _NVFBC_TOHWENC_GRAB_FRAME_PARAMS, 82
- ppBuffer
 - _NVFBC_TOSYS_SETUP_PARAMS, 87
- ppDiffMap
 - _NVFBC_TOGL_SETUP_PARAMS, 78
 - _NVFBC_TOSYS_SETUP_PARAMS, 88
- Requirements, 15
- Structure Definition, 18
- ulTimestampUs
 - _NVFBC_FRAME_GRAB_INFO, 59

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.

Trademarks

NVIDIA, NVIDIA GRID, and the NVIDIA logo are trademarks and/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

© 2011-2018 NVIDIA Corporation. All rights reserved.