NVIDIA Framebuffer Capture API for Linux Reference Manual

November 23, 2021

Version 8.0.8

Contents

Chapter 1

NVIDIA Framebuffer Capture (NvFBC) for Linux.

NvFBC is a high performance, low latency API to capture 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

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

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 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: Required. NvFBC relies on OpenGL to perform frame capture and post-processing.
- Vulkan 1.1: Required.
- libcuda.so.1 >= 5.5: Optional. Used for capture to video memory with CUDA interop.

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: Required. 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: Required. 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.

6.2 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 'ulTimestampUs' to NVFBC_FRAME_GRAB_INFO.
- Added 'dwSamplingRateMs' to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.
- Added 'bPushModel' to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.

NvFBC Linux API version 1.8

- Retired the NVFBC_CAPTURE_TO_HW_ENCODER interface. This interface has been deprecated since NvFBC 1.2 and has received no updates or new features since. We recommend using the NVIDIA Video Codec SDK to encode NvFBC frames. See: https://developer.nvidia.com/nvidia-video-codec-sdk
- Added a 'Capture Modes' section to those headers.
- Added a 'Post Processing' section to those headers.

- Added an 'Environment Variables' section to those headers.
- Added 'bInModeset' to NVFBC_GET_STATUS_PARAMS.
- Added 'bAllowDirectCapture' to NVFBC_CREATE_CAPTURE_SESSION_PARAMS.
- Added 'bDirectCaptured' to NVFBC_FRAME_GRAB_INFO.
- Added 'bRequiredPostProcessing' to NVFBC_FRAME_GRAB_INFO.

6.3 Capture Modes

When creating a capture session, NvFBC instantiates a capture subsystem living in the NVIDIA X driver.

When creating a capture session, NvFBC instantiates a capture subsystem living in the NVIDIA X driver.

This subsystem listens for damage events coming from applications then generates (composites) frames for NvFBC when new content is available.

This capture server can operate on a timer where it periodically checks if there are any pending damage events, or it can generate frames as soon as it receives a new damage event. See NVFBC_CREATE_CAPTURE_SESSION_PARAMS::dwSamplingRateMs, and NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bPushModel.

NvFBC can also attach itself to a fullscreen unoccluded application and have it copy its frames directly into a buffer owned by NvFBC upon present. This mode bypasses the X server. See NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bAllowDirectCapture.

NvFBC is designed to capture frames with as few copies as possible. The NVIDIA X driver composites frames directly into the NvFBC buffers, and direct capture copies frames directly into these buffers as well.

Depending on the configuration of a capture session, an extra copy (rendering pass) may be needed. See the 'Post Processing' section.

6.4 Post Processing

Depending on the configuration of a capture session, NvFBC might require to do post processing on frames, which consists of an extra frame copy.

Depending on the configuration of a capture session, NvFBC might require to do post processing on frames, which consists of an extra frame copy.

Post processing is required for the following reasons:

- NvFBC needs to do a pixel format conversion.
- Diffmaps are requested.
- Capture to system memory is requested.

NvFBC needs to do a conversion if the requested pixel format does not match the native format. The native format is NVFBC_BUFFER_FORMAT_BGRA.

Note: post processing is ∗not∗ required for frame scaling and frame cropping.

Skipping post processing can reduce capture latency and video memory usage. An application can know whether post processing was required by checking NVFBC_FRAME_GRAB_INFO::bRequiredPostProcessing.

6.5 Environment Variables

Below are the environment variables supported by NvFBC:.

Below are the environment variables supported by NvFBC:.

- NVFBC_LOG_LEVEL Bitfield where the first bit enables debug logs and the second bit enables performance logs. Both can be enabled by setting this envvar to 3.
- NVFBC_LOG_FILE Write all NvFBC logs to the given file. Specifying this variable allows the NVIDIA X driver to generate logs as well, provided the X server has permission to access the given file.
- NVFBC_FORCE_ALLOW_DIRECT_CAPTURE Used to override NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bAllowDirectCapture.
- NVFBC_FORCE_POST_PROCESSING Used to force the post processing step, even if it could be skipped. See the 'Post Processing' section.

6.6 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 8

  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_BUFFER_FORMAT_YUV420P** NVFBC_BUFFER_FORMAT_NV12
- #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, 2)

  NVFBC_GET_STATUS_PARAMS structure version.

- #define NVFBC_CREATE_CAPTURE_SESSION_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 6)

  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, NVFBC_ERR_VULKAN = 17 }

  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 = 1, NVFBC_CAPTURE_TO_GL = 3 }

  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.

6.6.1 Typedef Documentation

6.6.1.1 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.

6.6.1.2 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

6.6.1.3 typedef enum _NVFBCSTATUS NVFBCSTATUS

Defines error codes.

See also:

NvFBCGetLastErrorStr

6.6.2 Enumeration Type Documentation

6.6.2.1 enum _NVFBC_BOOL

Defines boolean values.

Enumerator:

NVFBC_FALSE False value.

NVFBC_TRUE True value.

6.6.2.2 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 Native format.
No pixel conversion needed. BGRA8888 byte-order format. 32 bpp.

6.6.2.3 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.

NVFBC_CAPTURE_TO_GL Retired.
Do not use.
Capture frames to an OpenGL buffer in video memory.

6.6.2.4 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.

NVFBC_ERR_VULKAN This indicates a Vulkan error.

6.6.2.5 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.

6.6.2.6 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.

6.6.2.7 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.

6.6.2.8 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.

6.7 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 NvFBCCreateInstance (NVFBC_API_FUNCTION_LIST *pFunctionList)

  Entry Points to the NvFBC interface.

6.7.1 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.

6.7.2 Function Documentation

6.7.2.1 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

6.7.2.2 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. 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_MUST_RECREATE
NVFBC_ERR_INTERNAL

6.7.2.3 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

6.7.2.4 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

6.7.2.5 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

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

Destroys the handle of an NvFBC client.

This function uninitializes an FBC client.

This function implicitly calls NvFBCReleaseContext().

After this fucntion 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

6.7.2.7 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.

6.7.2.8 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

6.7.2.9 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 functino 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

6.7.2.10 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
NVFBC_ERR_MUST_RECREATE

See also:

NvFBCCreateCaptureSession
NvFBCToCudaSetUp

6.7.2.11 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_X

6.7.2.12 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
NVFBC_ERR_MUST_RECREATE

See also:

NvFBCCreateCaptureSession
NvFBCToCudaSetUp

6.7.2.13 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_X

6.7.2.14 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
NVFBC_ERR_MUST_RECREATE

See also:

NvFBCCreateCaptureSession
NvFBCToSysSetUp

6.7.2.15 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 7

Class Documentation

7.1 _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

7.1.1 Detailed Description

Defines parameters for the NvFBCBindContext() API call.

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

- NvFBC.h

7.2 _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.

7.2.1 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

7.3 _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

- NVFBC_BOOL bAllowDirectCapture

  [in] Allow direct capture

7.3.1 Detailed Description

Defines parameters for the NvFBCCreateCaptureSession() API call.

7.3.2 Member Data Documentation

7.3.2.1 NVFBC_BOOL _NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bAllowDirectCapture

[in] Allow direct capture

Direct capture allows NvFBC to attach itself to a fullscreen graphics application. Whenever that application presents a frame, it makes a copy of it directly into a buffer owned by NvFBC thus bypassing the X server.

When direct capture is ∗not∗ enabled, the NVIDIA X driver generates a frame for NvFBC when it receives a damage event from an application if push model is enabled, or periodically checks if there are any pending damage events otherwise (see NVFBC_CREATE_CAPTURE_SESSION_PARAMS::dwSamplingRateMs).

Direct capture is possible under the following conditions:

- Direct capture is allowed
- Push model is enabled (see NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bPushModel)
- The mouse cursor is not composited (see NVFBC_CREATE_CAPTURE_SESSION_PARAMS::bWithCursor)
- No viewport transformation is required. This happens when the remote desktop is e.g. rotated.

When direct capture is possible, NvFBC will automatically attach itself to a fullscreen unoccluded application, if such exists.

Notes:

- This includes compositing desktops such as GNOME (e.g., gnome-shell is the fullscreen unoccluded application).
- There can be only one fullscreen unoccluded application at a time.
- The NVIDIA X driver monitors which application qualifies or no longer qualifies.

For example, if a fullscreen application is launched in GNOME, NvFBC will detach from gnome-shell and attach to that application.

Attaching and detaching happens automatically from the perspective of an NvFBC client. When detaching from an application, the X driver will transparently resume generating frames for NvFBC.

An application can know whether a given frame was obtained through direct capture by checking NVFBC_FRAME_GRAB_INFO::bDirectCapture.

7.3.2.2 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 occured and can automatically re-create a capture session with the same settings as before, then resume its frame capture session transparently.

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.

Note that during modeset recovery, NvFBC will try to re-create the capture session every second until it succeeds.

7.3.2.3 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.

7.3.2.4 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.

7.3.2.5 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.

7.3.2.6 NVFBC_BOX _NVFBC_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.

7.3.2.7 uint32_t _NVFBC_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 ($\sim$ 60 Hz).

7.3.2.8 NVFBC_CAPTURE_TYPE _NVFBC_CREATE_CAPTURE_SESSION_PARAMS::eCaptureType

[in] Desired capture type.

Note that when specyfing NVFBC_CAPTURE_SHARED_CUDA NvFBC will try to dlopen() the corresponding libraries. This means that NvFBC can run on a system without the CUDA library since it does not link against them.

7.3.2.9 NVFBC_SIZE _NVFBC_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

7.4 _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

7.4.1 Detailed Description

Defines parameters for the CreateHandle() API call.

7.4.2 Member Data Documentation

7.4.2.1 NVFBC_BOOL _NVFBC_CREATE_HANDLE_PARAMS::bExternallyManagedContext

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

NvFBC internally uses OpenGL to perfom 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.

7.4.2.2 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.

7.4.2.3 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

7.5 _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

7.5.1 Detailed Description

Defines parameters for the NvFBCDestroyCaptureSession() API call.

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

- NvFBC.h

7.6 _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

7.6.1 Detailed Description

Defines parameters for the NvFBCDestroyHandle() API call.

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

- NvFBC.h

7.7 _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

- uint32_t **dwMissedFrames**
- NVFBC_BOOL **bRequiredPostProcessing**
- NVFBC_BOOL **bDirectCapture**

7.7.1 Detailed Description

Describes information about a captured frame.

7.7.2 Member Data Documentation

7.7.2.1 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

7.7.2.2 uint32_t _NVFBC_FRAME_GRAB_INFO::dwCurrentFrame

[out] Incremental ID of the current frame.

This can be used to identify a frame.

7.7.2.3 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

7.8 _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.

- NVFBC_BOOL bInModeset

  [out] Whether the X server is currently in modeset.

7.8.1 Detailed Description

Defines parameters for the NvFBCGetStatus() API call.

7.8.2 Member Data Documentation

7.8.2.1 NVFBC_BOOL _NVFBC_GET_STATUS_PARAMS::bInModeset

[out] Whether the X server is currently in modeset.

When the X server is in modeset, it must give up all its video memory allocations. It is not possible to create a capture session until the modeset is over.

Note that VT-switches are considered modesets.

7.8.2.2 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.

7.8.2.3 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.

7.8.2.4 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

7.9 _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.

7.9.1 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

7.9.2 Member Data Documentation

7.9.2.1 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

7.10 _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

7.10.1 Detailed Description

Defines parameters for the NvFBCReleaseContext() API call.

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

- NvFBC.h

7.11 _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.

7.11.1 Detailed Description

Size used to describe the size of a frame.

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

- NvFBC.h

7.12 _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.

7.12.1 Detailed Description

Defines parameters for the NvFBCToCudaGrabFrame() API call.

7.12.2 Member Data Documentation

7.12.2.1 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.

7.12.2.2 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.

7.12.2.3 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

7.13 _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.

7.13.1 Detailed Description

Defines parameters for the NvFBCToCudaSetUp() API call.

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

- NvFBC.h

7.14 _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.

7.14.1 Detailed Description

Defines parameters for the NvFBCToGLGrabFrame() API call.

7.14.2 Member Data Documentation

7.14.2.1 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.

7.14.2.2 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.

7.14.2.3 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

7.15 _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.

7.15.1 Detailed Description

Defines parameters for the NvFBCToGLSetUp() API call.

7.15.2 Member Data Documentation

7.15.2.1 NVFBC_SIZE _NVFBC_TOGL_SETUP_PARAMS::diffMapSize

[out] Size of the differential map.

Only set if bWithDiffMap is set to NVFBC_TRUE.

7.15.2.2 uint32_t _NVFBC_TOGL_SETUP_PARAMS::dwDiffMapScalingFactor

[in] Scaling factor of the differential maps.

See also:

NVFBC_TOSYS_SETUP_PARAMS::dwDiffMapScalingFactor

7.15.2.3 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.

7.15.2.4 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

7.16 _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.

7.16.1 Detailed Description

Defines parameters for the NvFBCToSysGrabFrame() API call.

7.16.2 Member Data Documentation

7.16.2.1 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.

7.16.2.2 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

7.17 _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.

7.17.1 Detailed Description

Defines parameters for the NvFBCToSysSetUp() API call.

7.17.2 Member Data Documentation

7.17.2.1 NVFBC_SIZE _NVFBC_TOSYS_SETUP_PARAMS::diffMapSize

[out] Size of the differential map.

Only set if bWithDiffMap is set to NVFBC_TRUE.

7.17.2.2 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.

7.17.2.3 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.

7.17.2.4 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

7.18 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().

- PNVFBCTOSYSSETUP nvFBCToSysSetUp

  [out] Pointer to NvFBCToSysSetUp().

- PNVFBCTOSYSGRABFRAME nvFBCToSysGrabFrame

  [out] Pointer to NvFBCToSysGrabFrame().

- PNVFBCTOCUDASETUP nvFBCToCudaSetUp

  [out] Pointer to NvFBCToCudaSetUp().

- PNVFBCTOCUDAGRABFRAME nvFBCToCudaGrabFrame

  [out] Pointer to NvFBCToCudaGrabFrame().

- void * pad1

  [out] Retired.

- void * pad2

  [out] Retired.

- void * pad3

  [out] Retired.

- PNVFBCBINDCONTEXT nvFBCBindContext
  [out] Pointer to NvFBCBindContext().
- PNVFBCRELEASECONTEXT nvFBCReleaseContext
  [out] Pointer to NvFBCReleaseContext().
- void * pad4
  [out] Retired.
- void * pad5
  [out] Retired.
- void * pad6
  [out] Retired.
- void * pad7
  [out] Retired.
- PNVFBCTOGLSETUP nvFBCToGLSetUp
  [out] Pointer to nvFBCToGLSetup().
- PNVFBCTOGLGRABFRAME nvFBCToGLGrabFrame
  [out] Pointer to nvFBCToGLGrabFrame().

7.18.1 Detailed Description

Structure populated with API function pointers.

7.18.2 Member Data Documentation

7.18.2.1 uint32_t NVFBC_API_FUNCTION_LIST::dwVersion

[in] Must be set to NVFBC_VERSION.

7.18.2.2 PNVFBCBINDCONTEXT NVFBC_API_FUNCTION_LIST::nvFBCBindContext

[out] Pointer to NvFBCBindContext().

7.18.2.3 PNVFBCCREATECAPTURESESSION NVFBC_API_FUNCTION_LIST::nvFBCCreateCaptureSession

[out] Pointer to NvFBCCreateCaptureSession().

7.18.2.4 PNVFBCCREATEHANDLE NVFBC_API_FUNCTION_LIST::nvFBCCreateHandle

[out] Pointer to NvFBCCreateHandle().

7.18.2.5 PNVFBCDESTROYCAPTURESESSION NVFBC_API_FUNCTION_-LIST::nvFBCDestroyCaptureSession

[out] Pointer to NvFBCDestroyCaptureSession().

7.18.2.6 PNVFBCDESTROYHANDLE NVFBC_API_FUNCTION_LIST::nvFBCDestroyHandle

[out] Pointer to NvFBCDestroyHandle().

7.18.2.7 PNVFBCGETLASTERRORSTR NVFBC_API_FUNCTION_LIST::nvFBCGetLastErrorStr

[out] Pointer to NvFBCGetLastErrorStr().

7.18.2.8 PNVFBCGETSTATUS NVFBC_API_FUNCTION_LIST::nvFBCGetStatus

[out] Pointer to NvFBCGetStatus().

7.18.2.9 PNVFBCRELEASECONTEXT NVFBC_API_FUNCTION_LIST::nvFBCReleaseContext

[out] Pointer to NvFBCReleaseContext().

7.18.2.10 PNVFBCTOCUDAGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToCudaGrabFrame

[out] Pointer to NvFBCToCudaGrabFrame().

7.18.2.11 PNVFBCTOCUDASETUP NVFBC_API_FUNCTION_LIST::nvFBCToCudaSetUp

[out] Pointer to NvFBCToCudaSetUp().

7.18.2.12 PNVFBCTOGLGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToGLGrabFrame

[out] Pointer to nvFBCToGLGrabFrame().

7.18.2.13 PNVFBCTOGLSETUP NVFBC_API_FUNCTION_LIST::nvFBCToGLSetUp

[out] Pointer to nvFBCToGLSetup().

7.18.2.14 PNVFBCTOSYSGRABFRAME NVFBC_API_FUNCTION_LIST::nvFBCToSysGrabFrame

[out] Pointer to NvFBCToSysGrabFrame().

7.18.2.15 PNVFBCTOSYSSETUP NVFBC_API_FUNCTION_LIST::nvFBCToSysSetUp

[out] Pointer to NvFBCToSysSetUp().

7.18.2.16 void* NVFBC_API_FUNCTION_LIST::pad1

[out] Retired.

Do not use.

7.18.2.17 void* NVFBC_API_FUNCTION_LIST::pad2

[out] Retired.

Do not use.

7.18.2.18 void* NVFBC_API_FUNCTION_LIST::pad3

[out] Retired.

Do not use.

7.18.2.19 void* NVFBC_API_FUNCTION_LIST::pad4

[out] Retired.

Do not use.

7.18.2.20 void* NVFBC_API_FUNCTION_LIST::pad5

[out] Retired.

Do not use.

7.18.2.21 void* NVFBC_API_FUNCTION_LIST::pad6

[out] Retired.

Do not use.

7.18.2.22 void* NVFBC_API_FUNCTION_LIST::pad7

[out] Retired.

Do not use.

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

- NvFBC.h

Chapter 8

File Documentation

8.1 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_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 8

  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_BUFFER_FORMAT_YUV420P** NVFBC_BUFFER_FORMAT_NV12
- #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, 2)

  NVFBC_GET_STATUS_PARAMS structure version.

- #define NVFBC_CREATE_CAPTURE_SESSION_PARAMS_VER NVFBC_STRUCT_VERSION(NVFBC_CREATE_CAPTURE_SESSION_PARAMS, 6)

  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.

- 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, NVFBC_ERR_VULKAN = 17 }

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 = 1, NVFBC_CAPTURE_TO_GL = 3 }

  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.

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 NvFBCCreateInstance (NVFBC_API_FUNCTION_LIST *pFunctionList)

  Entry Points to the NvFBC interface.

8.1.1 Detailed Description

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

Index

Notice

Trademarks

Copyright