RapidFire Programming Guide

Contents

Preface

About This Document

This document is a programming guide for the RapidFire SDK. The major goal of this document is to provide an introduction to the implementation of cloud gaming and virtualization applications using the RapidFire SDK.

AMD’s RapidFire technology is a combination of hardware and software that enables ISVs to benefit from an open API and adapt existing graphics applications easily to the cloud requirements.

Audience

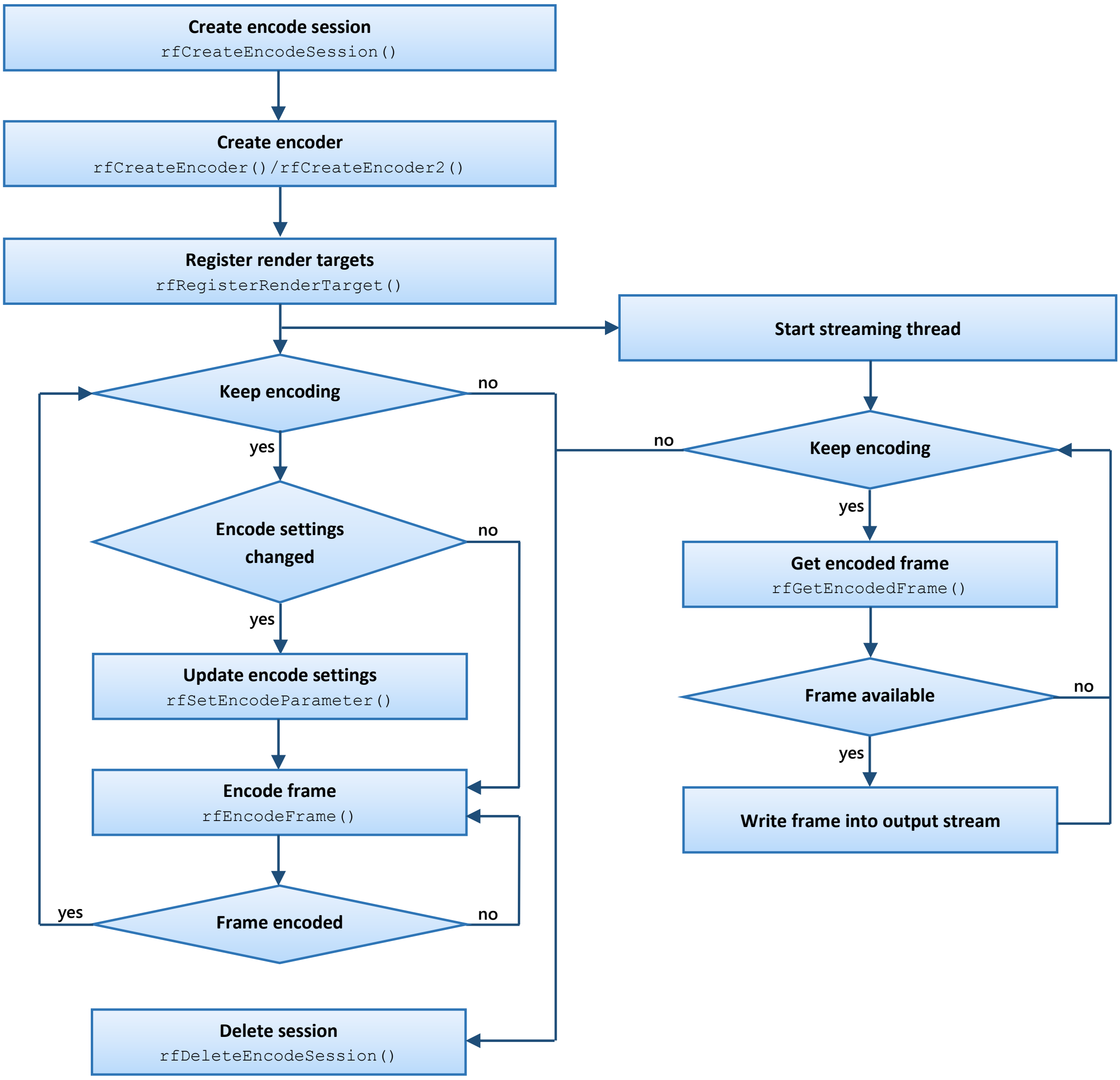
This document is intended for programmers. Prior knowledge in writing code for CPUs and a basic understanding of video encoding, video decoding, OpenGL, D3D9 and D3D11 is assumed. While a basic understanding of GPU architectures and computer graphics is useful.

RapidFire Server Overview

The RapidFire SDK enables ISVs to implement remote rendering solutions.

The server component of the API handles the asynchronous encoding of the frame buffer as a H.264 or uncompressed stream; it supports the following rendering APIs: OpenGL 4.3, DirectX 9 and Direct11. The API also provides functions that allow the capturing and encoding of the entire desktop.

API Flow Diagram:



RapidFire Server API

rfCreateEncodeSession

RFStatus rfCreateEncodeSession(RFEncodeSession* session,
RFProperties* properties)

Description: Creates an encode session. A session encapsulates the encoder to be used, the graphics context that is used by the calling application and the OpenCL context that is used for the color space conversion. If the environment variable **RF_LOG_PATH** is set, a log file is generated at the specified path that contains further information for the session.

A session can have different sources for the encoding. It could be a render target, the whole desktop or a single window. The following sessions can be created:

- OpenGL session.
- D3D9/D3D9Ex session.
- D3D11 session.
- Desktop capture session.

The parameter properties specifies a list of session property names and their corresponding values. Each property name is immediately followed by the corresponding value. The list is terminated with 0. Depending on the graphics API that is used by the application, this function takes the following properties as input:

Property name	Description	Property type
RF_GL_GRAPHICS_CTX	OpenGL Context	HGLRC
RF_GL_DEVICE_CTX	Device Context	HDC
RF_D3D9_DEVICE	Direct3D 9 Device	IDirect3DDevice9*
RF_D3D9EX_DEVICE	Direct3D 9 Ex Device	IDirect3DDevice9Ex*
RF_D3D11_DEVICE	Direct3D 11 Device	ID3D11Device*
RF_DESKTOP	Index of the Desktop	unsigned int
RF_DESKTOP_DSP_ID	Index of the Display	unsigned int
RF_DESKTOP_UPDATE_ - ON_CHANGE	If set to TRUE **rfEncodeFrame** will only enqueue a captured frame if the Desktop has changed	BOOL
RF_DESKTOP_BLOCK_ - UNTIL_CHANGE	If set to TRUE **rfEncodeFrame** will block until the Desktop has changed	BOOL

RF_FLIP_SOURCE	If set to TRUE the source image is flipped vertically	BOOL
RF_ASYNC_SOURCE_COPY	If set to TRUE the captured frame will be copied to system memory asynchronously	BOOL
RF_ENCODER	Specifies the encoder that will be used	RFEncoderID
RF_ENCODER_BLOCKING_ - READ	If set to TRUE **rfGetEncodedFrame** will block until AMF has finished encoding a frame	BOOL
RF_MOUSE_DATA	Enables capturing of the mouse cursor shape	BOOL

Possible **RFEncoderID**s:

RF_AMF	RF_IDENTITY	RF_DIFFERENCE

AMF stands for AMD Media Foundation library encoder (HW) that returns an H.264 stream. The **IDENTITY** encoder returns a buffer with the uncompressed source image in the specified color format and the **DIFFERENCE** encoder returns a difference map consisting of blocks with 1 being stored in regions where the source image has changed and 0 otherwise.

rfCreateEncodeSession returns **RF_STATUS_OK** if the session was created successfully. Otherwise the session is set to NULL and one of the following error codes is returned:

RF_STATUS_INVALID_SESSION_PROPERTIES if one or more of the properties is invalid.

RF_STATUS_INVALID_ENCODER if **RF_ENCODER** is not one of the possible **RFEncoderID**s.

RF_STATUS_INVALID_D3D_DEVICE if the input Direct3D device is invalid.

RF_STATUS_INVALID_OPENGL_CONTEXT if the input OpenGL context is invalid.

RF_STATUS_INVALID_DESKTOP_ID if the input Desktop ID is invalid.

RF_STATUS_INVALID_OPENCL_CONTEXT if the OpenCL context could not be created.

RF_STATUS_AMF_FAIL if AMF functions fail.

RF_STATUS_ DOPP_FAIL if DOPP extension initialization fails.

RF_STATUS_OPENCL_FAIL if OpenCL functions fail.

RF_STATUS_OPENGL_FAIL if the OpenGL context creation fails.

RF_STATUS_MEMORY_FAIL if memory allocation fails.

RF_STATUS_FAIL if other errors occur. More information is given in the log file.

rfDeleteEncodeSession

RFStatus rfDeleteEncodeSession(RFEncodeSession* session)

Description: Deletes the encoding session and frees all associated resources.

Parameters:

session	The encoding session to be deleted.

rfDeleteEncodeSession returns **RF_STATUS_OK** if the session is deleted successfully. Otherwise, it returns the following error value:

RF_STATUS_INVALID_SESSION if the session parameter is not a valid session.

rfCreateEncoder

```
RFStatus  rfCreateEncoder(RFEncodeSession session,
                          unsigned int uiWidth,
                          unsigned int uiHeight,
                          const RFEncodePreset preset)
```

Description: The function creates the encoder and resources like intermediate buffers. It initializes all properties based on the selected encoding preset.

Possible **RFEncodePreset**s:

RF_PRESET_FAST	RF_PRESET_BALANCED	RF_PRESET_QUALITY

Based on the preset, width and height, the encoder is configured. If desktop encoding is selected and the dimensions for the encoding differ from the desktop size, the image is scaled to match the encoding dimensions.

Parameters:

session	The encoding session.
uiWidth	The width of the encoded stream.
uiHeight	The height of the encoded stream.
preset	The encoding preset used.

rfCreateEncoder returns **RF_STATUS_OK** if the encoder is created successfully. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session parameter is not a valid session.

RF_STATUS_INVALID_CONFIG if no preset was specified.

RF_STATUS_INVALID_OPENCL_CONTEXT if the OpenCL context of the session is invalid.

RF_STATUS_INVALID_FORMAT if an invalid data format was specified for the encoder.

RF_STATUS_INVALID_DIMENSION if either uiWidth or uiHeight is 0 or larger than the maximum supported size of the encoder.

RF_STATUS_INVALID_ENCODER if no encoder was specified with the creation of the session.

RF_STATUS_AMF_FAIL if the AMF initialization fails.

RF_STATUS_INVALID_ENCODER_PARAMETER if an encoder property is not supported or has an invalid value.

RF_STATUS_OPENCL_FAIL if OpenCL resources creation fails.

RF_STATUS_FAIL if the creation of the encoder fails or the encoder is already created. More information is given in the log file.

rfCreateEncoder2

```
RFStatus  rfCreateEncoder2(RFEncodeSession session,
                           unsigned int uiWidth,
                           unsigned int uiHeight,
                           const RFProperties* properties)
```

Description: This function creates the encoder by passing the static and dynamic properties: Profile, Level, Bitrate, FPS etc.

Parameters:

session	The encoding session.
uiWidth	The width of the encoded stream.
uiHeight	The height of the encoded stream.
properties	A list of **RFProperties** and their corresponding values. The list is terminated with 0.

Properties specifies a list of encoder property names and their corresponding values. Each property name is immediately followed by the corresponding desired value. The list is terminated with 0. **rfCreateEncoder2** takes the following properties as input:

Description	Property name	Property type	Possible values and presets
Enum specifying input format of encoder	RF_ENCODER_FORMAT	RFFormat	RF_RGBA8 RF_ARGB8 RF_BGRA8 RF_NV12 RF_I420
Enum specifying the preset	RF_ENCODER_PRESET	RFEncodePreset	RF_PRESET_FAST RF_PRESET_BALANCED RF_PRESET_QUALITY
The H.264 Profile	RF_ENCODER_PROFILE	unsigned int	66 – Baseline 77 – Main (default) 100 – High
The H.264 Level	RF_ENCODER_LEVEL	unsigned int	10, 11, 12, 13, 20, 21, 22, 30, 31, 32, 40, 41, 42 (default), 50, 51
Target bitrate of the encoded stream	RF_ENCODER_BITRATE	unsigned int	10 KBits/s – 100 MBits/s 6 MBit/s (fast preset) 10 MBit/s (balanced preset) 20 MBit/s (quality preset)
Peak bitrate of the encoded stream	RF_ENCODER_PEAK_BITRATE	unsigned int	10 KBits/s – 100 MBits/s 6 MBit/s (fast preset) 10 MBit/s (balanced preset) 20 MBit/s (quality preset)
Rate control method	RF_ENCODER_RATE_CONTROL_-METHOD	unsigned int	0 – PEAK_COSNTRAINED_VBR (balanced and quality preset) 1 – LATENCY_CONSTRAINED_VBR (fast preset) 2 – CBR
Minimum Quantizer	RF_ENCODER_MIN_QP	unsigned int	0 – 51 22 (fast and balanced preset) 18 (quality preset)
Maximum Quantizer	RF_ENCODER_MAX_QP	unsigned int	0 – 51 51 (default)
Frames per second	RF_ENCODER_FRAME_RATE	unsigned int	60 (fast and balanced preset) 30 (quality preset)
Frame rate denominator	RF_ENCODER_FRAME_RATE_DEN	unsigned int	1 – frame rate 1 (default)
Peak Signal Noise Ratio tuning and computation. Enables PSNR calculations that are reported on completion at the cost of a small decrease in speed.	RF_ENCODER_PSNR	bool	false (default)
GOP size	RF_ENCODER_GOP_SIZE	unsigned int	0 – 1000 60 (default)
Video Buffering Verifier buffer size	RF_ENCODER_VBV_BUFFER_SIZE	unsigned int	1 KBit – 100 MBit 110 KBit (fast preset) 1 MBit (balanced preset) 20 MBit (quality preset)
Initial fullness of the VBV buffer	RF_ENCODER_VBV_BUFFER_-FULLNESS	unsigned int	0 – 64 64 (default)
Enforce Hypothetical Reference Decoder	RF_ENCODER_ENFORCE_HRD	bool	true (fast and balanced preset) false (quality preset)

Maximum interval between IDR-frames	RF_ENCODER_IDR_PERIOD	unsigned int	0 – 1000 300 (fast and balanced preset) 30 (quality preset)
Intra refresh number of MB per slot	RF_ENCODER_INTRA_REFRESH_-NUM_MB	unsigned int	0 - #MB per frame 225 (fast and balanced preset) 0 (quality preset)
Enable deblocking filter to avoid blocking artifacts	RF_ENCODER_DEBLOCKING_FILTER	bool	false (fast and balanced preset) true (quality preset)
Number of slices per frame	RF_ENCODER_NUM_SLICES_PER_-FRAME	unsigned int	1 - #MBits per frame 1 (default)
Quality preset	RF_ENCODER_QUALITY_PRESET	unsigned int	0 (Balanced) (quality preset) 1 (Speed) (fast and balanced preset) 2 (Quality)
Maximum range of motion search in pixels	RF_ENCODER_MOTION_RANGE	unsigned int	4 – 16 16 (default)
Use half pixels for motion estimation	RF_ENCODER_HALF_PIXEL	unsigned int	0, 1 1 (default)
Use quarter pixels for motion estimation	RF_ENCODER_QUARTER_PIXEL	unsigned int	0, 1 1 (default)
Force intra refresh (pre submission)	RF_ENCODER_FORCE_INTRA_-REFRESH	bool	false (default)
Force I-frame (pre submission)	RF_ENCODER_FORCE_I_FRAME	bool	false (default)
Force P-frame (pre submission)	RF_ENCODER_FORCE_P_FRAME	bool	false (default)
Insert Sequence Parameter Set (pre submission)	RF_ENCODER_INSERT_SPS	bool	false (default)
Insert Picture Parameter Set (pre submission)	RF_ENCODER_INSERT_PPS	bool	false (default)
Block width for difference encoder	RF_DIFF_ENCODER_BLOCK_S	unsigned int	BLOCK_S == multiple of 8 BLOCK_S * BLOCK_T == multiple of 64 16 (default)
Block height for difference encoder	RF_DIFF_ENCODER_BLOCK_T	unsigned int	BLOCK_T == multiple of 8 BLOCK_S * BLOCK_T == multiple of 64 16 (default)
Should be set to true if the **RF_DIFFERENCE** encoder is used and **rfGetEncodedFrame** is called in a different thread than **rfEncodeFrame**.	RF_DIFF_ENCODER_LOCK_BUFFER	bool	false (default)

RFFormat is the data format of the input for the encoder:

RF_ARGB8	RF_BGRA8	RF_RGBA8	RF_NV12	RF_I420

ARGB8, BGRA8 and **RGBA8** define 32-bit RGB values with Alpha, each pixel is represented by one byte each for the red, green, blue, and alpha channels. **NV12** is an 8-bit Y plane followed by an interleaved U/V plane with 2x2 subsampling. **I420** is an 8 bit Y plane followed by 8 bit 2x2 subsampled U plane, and finally followed by 8 bit 2x2 subsampled V plane.

RFEncodePreset is the preset for the encoder:

RF_ENCODE_FAST	RF_ENCODE_BALANCED	RF_ENCODE_QUALITY

FAST is fast encoding. **BALANCED** is balanced between quality and speed. **QUALITY** is high video quality.

rfCreateEncoder2 returns **RF_STATUS_OK** if the encoder is created successfully. Otherwise, it returns an error value listed in function **rfCreateEncoder** as well as the following error code:

RF_STATUS_INVALID_PROPERTIES if one of the properties is invalid.

rfRegisterRenderTarget

```
RFStatus  rfRegisterRenderTarget(RFEncodeSession session,
                                 RFRenderTarget renderTarget,
                                 unsigned int uiRTWidth,
                                 unsigned int uiRTHeight,
                                 unsigned int* idx)
```

Description: This function registers a render target that is created by the user and returns the index used for this render target in idx. The render target must have the same dimensions as the encoder. Otherwise the call fails. Up to 3 render targets can be registered and used for triple buffering.

Parameters:

session	The encoding session.
renderTarget	The handle of the render target.
uiRTWidth	The width of the render target.
uiRTHeight	The height of the render target.
idx	The index used for this render target.

Possible **RFRenderTarget** formats are:

unsigned int	OpenGL texture
ID3D11Texture2D*	DirectX 11 texture
IDirect3DSurface9*	DirectX 9 texture

rfRegisterRenderTarget returns **RF_STATUS_OK** if it registers a render target successfully. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_DIMENSION if uiRTWidth or uiRTHeight is zero or if the render target dimension does not match the encoder dimension.

RF_STATUS_INVALID_RENDER_TARGET if the renderTarget is an invalid handle.

RF_STATUS_INVALID_OPENCL_CONTEXT if the OpenCL context of the session is invalid.

RF_STATUS_RENDER_TARGET_FAIL if the application has already registered the maximum number of 3 render targets.

RF_STATUS_OPENCL_FAIL if the creation of an OpenCL buffer for the render target failed.

RF_STATUS_FAIL if other errors occur. More information is given in the log file.

rfRemoveRenderTarget

RFStatus rfRemoveRenderTarget(RFEncodeSession session,
unsigned int idx)

Description: This function removes a registered render target with the index idx which is no longer used in the session.

Parameters:

session	The encoding session.
idx	The index of the render target.

rfRemoveRenderTarget returns **RF_STATUS_OK** if it removes a registered render target successfully. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_INDEX if the index idx exceeds the maximum number of 3 render targets.

RF_STATUS_INVALID_RENDER_TARGET if the render target is not known or not registered.

rfGetRenderTargetState

```
RFStatus  rfGetRenderTargetState(RFEncodeSession session,
                                 RFRenderTargetState* state,
                                 unsigned int idx)
```

Description: This function gets the state of the render target with the index idx.

Parameters:

session	The encoding session.
state	The state of the render target.
idx	The index of the render target.

RFRenderTargetState is the state of a render target:

RF_STATE_INVALID	RF_STATE_FREE	RF_STATE_BLOCKED

INVALID: The render target is not known or not registered yet.

FREE: The render target was successfully registered and is currently not used by the API. It can be used by the application.

BLOCKED: The render target was submitted for encoding and is currently in use by the API. It should not be used by the application.

rfGetRenderTargetState returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_OPENCL_CONTEXT if the OpenCL context of the session is invalid.

RF_STATUS_INVALID_INDEX if the index idx is out of range or the render target was not registered.

rfResizeSession

```
RFStatus rfResizeSession(RFEncodeSession session,
                         unsigned int uiWidth,
                         unsigned int uiHeight)
```

Description: This function resizes the session and the encoder if the encoder supports resizing. Render targets that are registered by the application won’t be resized automatically.

Parameters:

session	The encoding session.
uiWidth	The new width of the stream.
uiHeight	The new height of the stream.

rfResizeSession returns **RF_STATUS_OK** if the resources of the session were resized successfully. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_FAIL if other errors occur. More information is given in the log file.

rfEncodeFrame

RFStatus rfEncodeFrame(RFEncodeSession session,
unsigned int idx)

Description: This function is called once the application has finished rendering into the render target with the index idx. The render target will then be captured and encoded.

Parameters:

session	The encoding session.
idx	The index of the render target which will be encoded.

rfEncodeFrame returns **RF_STATUS_OK** if capturing and encoding are successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_CONTEXT if the AMF context of the encoder is invalid.

RF_STATUS_INVALID_OPENCL_CONTEXT if the OpenCL context of the session is invalid.

RF_STATUS_INVALID_ENCODER if the encoder is invalid or not created.

RF_STATUS_INVALID_FORMAT if the format of the encoder is invalid.

RF_STATUS_INVALID_INDEX if the index idx is out of range or the render target was not registered.

RF_STATUS_INVALID_OPENCL_MEMOBJ if the encoder can’t load the OpenCL result buffer for the encoding.

RF_STATUS_DOPP_NO_UPDATE if the session parameter **RF_DESKTOP_UPDATE_ON_CHANGE** was set and the desktop texture didn't change since the last call.

RF_STATUS_QUEUE_FULL if there are no more free result buffers in the queue. Call **rfGetEncodedFrame** to free the buffers.

RF_STATUS_OPENCL_FAIL if OpenCL functions fail.

RF_STATUS_AMF_FAIL if AMF encoding fails.

rfGetEncodedFrame

RFStatus rfGetEncodedFrame(RFEncodeSession session,
unsigned int* uiSize,
void pBitStream)**

Description: This function returns the encoded frame in pBitStream and the size in bytes of the encoded frame in uiSize.

For the **RF_DIFFERENCE** encoder the returned buffer is a 2D array of byte-blocks for the captured image, where a 1 stored in a block indicates that the image has changed. The captured image can be obtained with the **rfGetSourceFrame** function.

Parameters:

session	The encoding session.
uiSize	The size of the bit stream.
pBitStream	The bit stream of the encoded frame.

rfGetEncodedFrame returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_ENCODER if the encoder is invalid.

RF_STATUS_NO_ENCODED_FRAME if there is no encoded frame in the result buffer queue.

RF_STATUS_AMF_FAIL if an AMF function fails.

rfGetSourceFrame

```
RFStatus rfGetSourceFrame(RFEncodeSession session,
                          unsigned int* uisSize,
                          void** pBitStream)
```

Description: This function returns the image that was used as the input for the encoder. If a color format conversion is required for the encoder, the returned image contains the result of the color format conversion. The function needs to be called prior to **rfGetEncodedFrame** to guarantee that the returned image is the source image of the encoded image returned by **rfGetEncodedFrame**.

Parameters:

session	The encoding session.
uiSize	The size of the bit stream.
pBitStream	The bit stream of the encoded frame.

rfGetSourceFrame returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_ENCODER if the encoder is invalid.

RF_STATUS_NO_ENCODED_FRAME if there is no encoded frame in the result buffer queue.

rfSetEncodeParameter

```
RFStatus  rfSetEncodeParameter(RFEncodeSession session,
                               const int property,
                               RFProperties value)
```

Description: This function changes encoding properties that do not require a re-creation of the encoder, e.g. adapting the encoding quality to the available network bandwidth. The available properties depend on the selected encoder. This function should be called after **rfCreateEncoder/ rfCreateEncoder2**, otherwise the function does not have any effect.

Parameters:

session	The encoding session.
property	The encoding property to change.
value	The new value of the encoding property.

For a list of the other valid properties and descriptions refer to the properties table for **rfCreateEncoder2.**

rfSetEncodeParameter returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_ENCODER if the encoder is invalid.

RF_STATUS_INVALID_CONFIG if the encoder configuration is invalid.

RF_STATUS_INVALID_ENCODER_PARAMETER if the property is invalid.

RF_STATUS_PARAM_ACCESS_DENIED if the encoder property can't be changed by the user.

RF_STATUS_FAIL if the encoder of the session is invalid or the encoder property can't be changed without recreating the encoder.

rfGetEncodeParameter

RFStatus rfGetEncodeParameter(RFEncodeSession session,
const int property,
RFProperties* value)

Description: This function queries the encoder property of the parameter property. The available properties depend on the selected encoder. This function should be called after **rfCreateEncoder**/ **rfCreateEncoder2**, otherwise the function does not have any effect.

Parameters:

session	The encoding session.
property	The requested encoding property.
value	The value of the encoding property.

Possible encoder properties that can be queried are:

Description	Property name	Property type
The color format of the encoder output.	RF_ENCODER_FORMAT	RFFormat
The width of the render target.	RF_ENCODER_WIDTH	unsigned int
The height of the render target.	RF_ENCODER_HEIGHT	unsigned int
The width of the encoder output.	RF_ENCODER_OUTPUT_WIDTH	unsigned int
The height of the encoder output.	RF_ENCODER_OUTPUT_HEIGHT	unsigned int

For a list of other valid properties and descriptions refer to the properties table for **rfCreateEncoder2.**

rfGetEncodeParameter returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_INVALID_ENCODER if the encoder is invalid.

RF_STATUS_PARAM_ACCESS_DENIED if the property is blocked by AMF.

RF_STATUS_INVALID_ENCODER_PARAMETER if the property queried is invalid.

rfGetMouseData

RFStatus rfGetMouseData(RFEncodeSession session,
bool bWaitForShapeChange,
RFMouseData* mouseData)

Description: This function returns the mouse cursor shape data. To use it the session needs to be created with the RF_MOUSE_DATA set to true.

Parameters:

session	The encoding session.
bWaitForShapeChange	If true the call blocks until the mouse shape changed.
mouseData	The returned mouse shape data.

The **RFBitmapBuffer** structure stores the bitmap data:

uiWidth	The width of the bitmap.
uiHeight	The height of the bitmap.
uiPitch	The Pitch of the bitmap.
uiBitsPerPixel	Bits per pixel of the bitmap.
pPixels	The data of the bitmap.

The **RFMouseData** structure stores the cursor shape data:

bVisible	Is true if the cursor is visible and false otherwise.
uiXHot	The horizontal position of the cursor hot spot.
uiYHot	The vertical position of the cursor hot spot.
Mask	The cursor bitmask bitmap. If the cursor is monochrome, this bitmask is formatted so that the upper half is the cursor AND bitmask and the lower half is the XOR bitmask. If the cursor is colored, this mask defines the AND bitmask of the cursor.
color	The cursor color bitmap containing the color data. This member is optional. If the cursor is monochrome color.pPixels is NULL. For pixels with false in the mask bitmap the color is directly blended with the destination pixel. For pixels with true in the mask bitmap the color is XORed with the color of the destination pixel.

rfGetMouseData returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_MOUSEGRAB_NO_CHANGE if the cursor shape did not change.

RF_STATUS_FAIL if the session was not created with RF_MOUSE_DATA set to true.

rfReleaseEvent

RFStatus rfReleaseEvent(RFEncodeSession session,
RFNotification const rfNotification)

Description: This function signals a notification event. This can be used to unblock a thread that is waiting for the event to be signaled.

Parameters:

session	The encoding session.
rfNotification	Specifies which event to signal.

RapidFire uses **RFNotification** events to get signaled on mouse cursor shape changes and desktop changes:

RFDesktopNotification	RFMouseShapeNotification

The **RFDesktopNotification** event is only present if the session was created with the flag **RF_DESKTOP_UPDATE_ON_CHANGE** or **RF_DESKTOP_BLOCK_UNTIL_CHANGE** and is signaled if the desktop texture has changed.

The **RFMouseShapeNotification** event is only present if the session was created using the flag **RF_MOUSE_DATA** and is signaled if the mouse cursor shape has changed.

rfReleaseEvent returns **RF_STATUS_OK** if successful. Otherwise, it returns one of the following error values:

RF_STATUS_INVALID_SESSION if the session is not a valid session.

RF_STATUS_FAIL if **RFNotification** is invalid or the event was not created.