



version: 166.0

# 1 SDK Introduction

---

## Introduction



# QCAP SDK

The QCAP SDK technology provides the user a simple way to access capture device, perform various operations with captured video stream, recording, broadcasting and files playback/editing. QCAP provides a common interface for media across various programming languages, it is a multimedia framework that can render or record media files on demand at the request of the user.

## 1.1 Overview

The purpose of QCAP is significantly different than the previous AMESDK library. The SDK is based on AMESDK, but it has much more and more high-level APIs for a developer to easily access our capture card and to develop a rich client application. It supports following features:

- Video, Audio, 3D and VANC Streams Capture and Preview ( Mirror, Region and Clone )
- Various De-interlace Methods
- Text, Scrolling, Picture and Video OSD Objects with Alpha Blending Engine
- Chroma Key and Mask
- **BMP** and **JPEG** Snapshot ( Continuous, Scaling and Cropping )
- Intel® and NVIDIA® **MPEG2/H264** GPU Encoder Integration
- Multiple Streams Recording to **AVI**, **MP4**, **ASF**, **FLV**, **TS**, **WMV**, **WAV** and **SCF** ( Scaling & Cropping )
- Time-shift Recording **New**
- Share Recording Engine with Transition Effect Animation ( Transform, Scaling, Alpha Blending and Blinds )
- Various **RTSP**, **RTMP**, **HLS** and **MMS** Network Streaming Server & Client
- **Network Delay Live Streaming** **New**
- **ONVIF** Communication Server, Client and Emulator **New**
- File Playback, Editing and Repair Library ( Export, Merge, and Insert )
- Various Callback Functions Registration
- 3D Capture, Recording, Streaming and Playback Functions Support
- Add-on Sound Card and USB Camera Capture Support
- Multiple Cards Support

## 1.2 Programming Language Supported

The API supports VC, VB, C#, QT, Delphi, OpenCV, PowerBuilder, Java, ActiveX and LabView development environments.

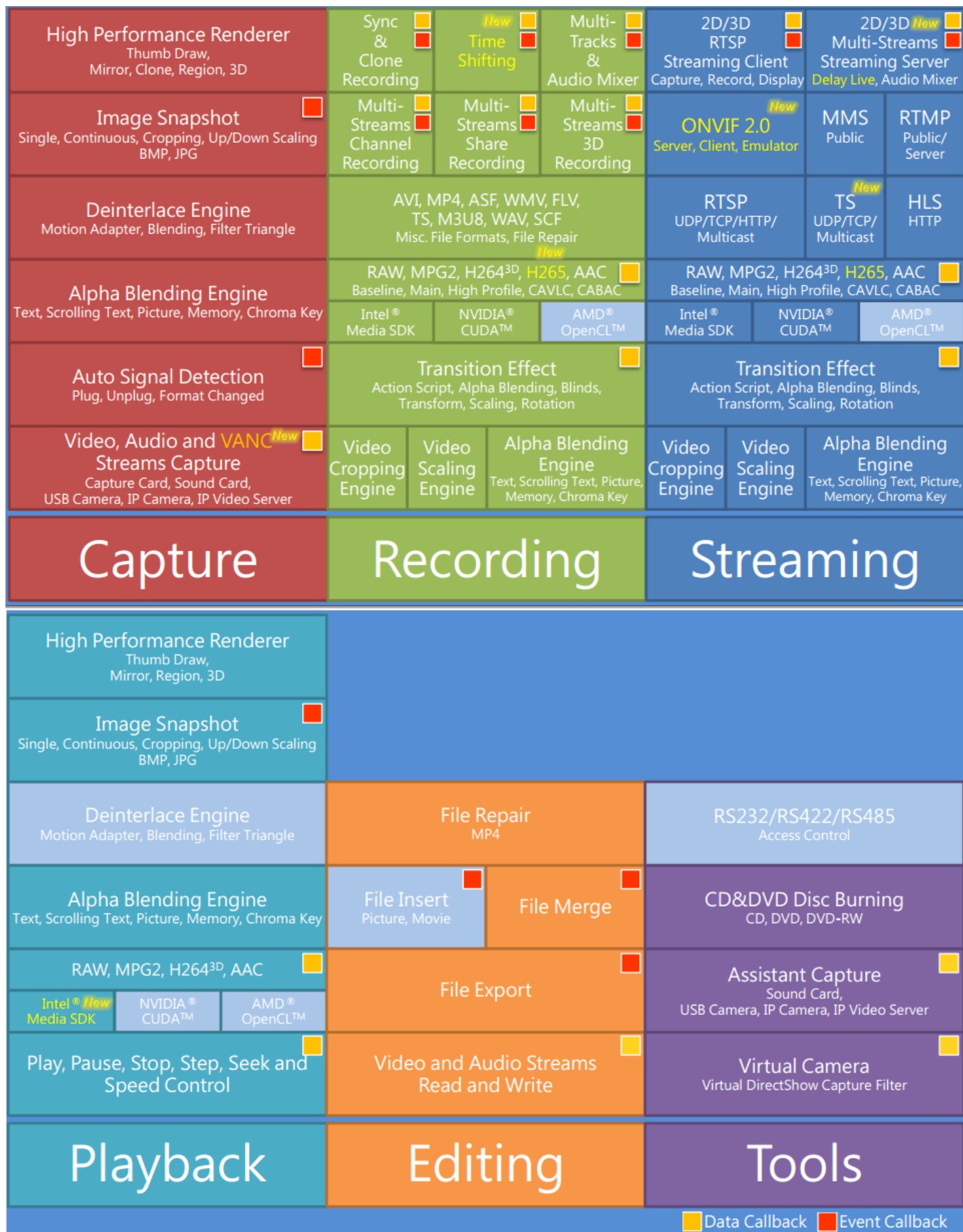
This document mainly use C language to illustrate APIs function and example.

Please refer to the sample codes to get examples of other languages.

## 1.3 SDK Function Blocks

The QCAP SDK is a **Capture, Recording, Streaming, Playback and Editing in One Library**.

QCAP SDK Function Block Diagram ENG 1.1.0





## 1.4 Using the SDK

In this section, we will guide and teach users how step by step to use QCAP SDK functions that with a capture card. The step-by-step instructions:

- Install Codec by executing **CODECS 1.1.0.exe** -
- Install Device Driver by executing **DRIVER.XXX\_YY\_1.1.0.exe**
  - Please make sure there is no *question marks* on the capture device icon.
- Unarchive **RELEASES 1.1.0.7z** to **QCAP\_SDK\**
  - Please use **7-Zip** ([www.7-zip.org](http://www.7-zip.org)) to extract **.7z** file.
- Select a sample code to **\SAMPLE\**
- Setting QCAP Header/Static/Run-time Libraries to your sample folder:
  - Copy **QCAP.H** from **QCAP\_SDK\QCAP\INC** to your project directory **SAMPLE\**
  - Copy **QCAP.LIB** from **QCAP\_SDK\QCAP\LIB\X86\VC.GPU** to your project directory **SAMPLE\**
  - Copy **QCAP.DLL / AMESDK.DLL** from **QCAP\_SDK\QCAP\LIB\X86\VC.GPU** to your executable directory **SAMPLE\Release** (or **.Debug**)
  - **Make sure the QCAP DLLs and your executable in the same directory!**
- Compile and run the sample

## Step A

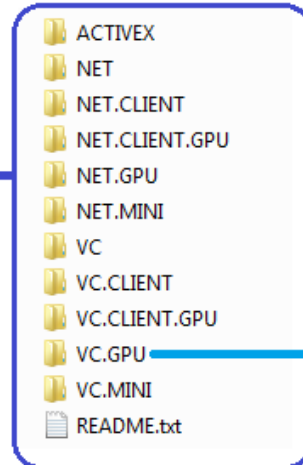
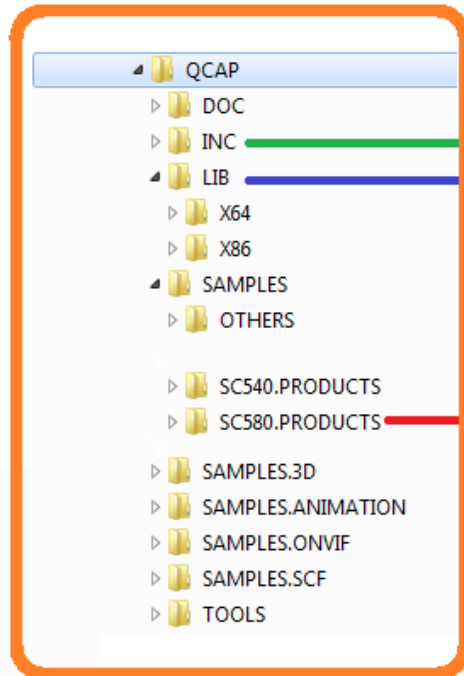
- BCB
- DELPHI
- DLL
- JAVA
- POWER.BUILDER
- VB6
- QCAP.H

## Step B

- AMESDK.DLL
- QCAP.DLL
- QCAP.X86.DLL
- QCAP.LIB
- QCAP.X86.LIB

## Step C

- HLS.SC580N1.VC6
- RTMP.SC580N1.VC6
- RTMP.WP.SC580N1.VC6
- SC580N1.NET.C#
- SC580N1.NET.C#.X86.VS2008
- SC580N1.NET.VB
- SC580N1.VC6
- SC580N1.VC2010
- SC580N4.VC6



## 1.5 SDK Package Organization

The SDK directories in **RELEASES 1.1.0.7z** archive and it descriptions explained in the list:

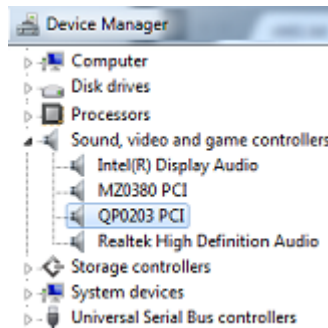
Folder	Descriptions
<b>AMESDK/</b>	A low-level library to allow you can directly access Capture card, Codec, File Operatinos and Network lib
<b>CVRSDK/</b>	A high-performance rendering engine to support multiple-channels display and alpha blending based on Direct3D
<b>DSHOW/</b>	Allows to use DirectShow to access our capture directly
<b>QCAP/</b>	A easy lib allow you to use some high-level APIs to develop your software based on AMESDK
QCAP/DOC	<b>QCAP SDK Documentations and User's Manual</b>
QCAP/INC	<b>QCAP SDK Include header files</b>
QCAP/LIB/	<b>QCAP SDK Static Library/Dynamic Link Libraries</b>
QCAP/LIB/X86/	QCAP SDK Static Library/Dynamic Link Libraries for X86 platform
QCAP/LIB/X86/VC/	[VC User] For VC6 / VC2005~VS2015 / LABVIEW / DELPHI Languages
QCAP/LIB/X86/VC.GPU/	[VC User] For INTEL/NVIDIA H.264 GPU Encoder/Decoder
QCAP/LIB/X86/VC.CLIENT/	[VC User] For RTSP client applications without Encoder implementation
QCAP/LIB/X86/NET/	[.NET User] For C#, VB, J# Languages
QCAP/LIB/X86/NET.GPU/	[.NET User] INTEL/NVIDIA H.264 GPU Encoder/Decoder
QCAP/LIB/X86/NET.CLIENT/	[.NET User] RTSP client applications without Encoder implementation
QCAP/LIB/X64/	QCAP SDK Static Library/Dynamic Link Libraries for X64 platform. ( please refer to X86/)
QCAP/SAMPLES/	<b>QCAP SDK Sample Code and Sample Programs</b>
QCAP/SAMPLES.3D/	QCAP SDK Sample Code for 3D video format
QCAP/SAMPLES.ANIMATION/	QCAP SDK Sample Code for Animation
QCAP/SAMPLES.ONVIF/	QCAP SDK Sample Code for ONVIF communication
QCAP/SAMPLES.SCF/	QCAP SDK Sample Code for SCF file playback
QCAP/TOOLS/	QCAP SDK Utilities

In **QCAP/LIB** directory, the binary **QCAP.X86.DLL/.LIB** are equal to **QCAP.DLL/.LIB**. The reason to have different names is because **QCAP.DLL** conflicts to system driver **QCAP.SYS**. So it is suggested to use **QCAP.X86.DLL**.

For LABVIEW user need to set `_QCAP_LABVIEW_IMPORT` on "Preprocessor Definitions" during importing

## 1.6 Device Software Flow

Before Using the QCAP SDK, user must make sure the capture card driver had properly installed. It can be make sure in Windows *Device Manager* that the device is shwon with no question mark (?):



The Software flow of using a QCAP SDK:

Programming Step	Related API Functions
Initialize <b>COM</b>	<a href="#">CoInitialize()</a>
Create Device	<a href="#">QCAP_CREATE()</a>
Set Device Parameters	<a href="#">QCAP_SET_VIDEO_INPUT()</a> <a href="#">QCAP_SET_VIDEO_DEINTERLACE()</a> <a href="#">QCAP_SET_VIDEO_BRIGHTNESS()</a> <a href="#">QCAP_SET_AUDIO_VOLUME()</a>
Start Capturing	<a href="#">QCAP_RUN()</a>
Snapshot	<a href="#">QCAP_SNAPSHOT_BMP()</a> <a href="#">QCAP_SNAPSHOT_JPG()</a>
Start Recording	<a href="#">QCAP_SET_VIDEO_RECORD_PROPERTY()</a> <a href="#">QCAP_SET_AUDIO_RECORD_PROPERTY()</a> <a href="#">QCAP_START_RECORD()</a>
Stop Recording	<a href="#">QCAP_STOP_RECORD()</a>
Stop Capturing	<a href="#">QCAP_STOP()</a>
Delete Device	<a href="#">QCAP_DESTROY()</a>
Un-initialize <b>COM</b>	<a href="#">CoUninitialize()</a>



Application must call [CoInitializes\(\)](#) to initialize the Windows **COM** library first.

## 1.7 Sample Code

Rich VC, VB, C#, QT, Delphi, PowerBuilder, OpenCV, Java, ActiveX and LabView sample code and the sample program are available in **QCAP SDK\QCAP\SAMPLES**:

Folder	Sample topic
SAMPLES/OTHERS	Bidirectional voices
	Camera
	Colorkey
	File player
	File Player QT
	File Player with Time shift
	Frame edit file player
	GDI+ cool string
	GDI+ draw arrow
	OpenCV
	Scrolling Horizontal / Vertical
SAMPLES/[CARD]/	1CH/4CH sample
	NTSC/PAL sample
	VANC buffer
	Delay Live Broadcast
	RTSP Server/client
	HLS Server/client
	RTMP Server/client
	RTMP WebPortal
	MMS WebPortal
SAMPLES.3D/	3D File Player
	3D File Player on 3D Display
	3D Recording
	3D Content Streaming
SAMPLES.ANIMATION/	Rotation
	Time-Lapse Video
	Unlocks Screen Recording
	Common (with/without QT)
	Fadeout
	Fadeout. 5CH Recording
	Picture-In-Picture, Picture-On-Picture
	Virtual Camera
	Audio Mixer
	Common 8CH Recording
SAMPLES.ONVIF/	ONVIF Client
	ONVIF Emulator
	ONVIF Server
SAMPLES.SCF/	SCF Recorder
	SCF File Player
	SCF Multifile player (with/without ActiveX)

## 1.8 Platform and Thread Safety

For both X86 and X64 of Windows XP, Windows Vista, Windows 7, Windows 8 and Windows Server 2005/2008 are all supported by this SDK. API calling can be used in the multiple-process and the multiple threads.



Intel® Media SDK and NVIDIA® H.264 encoders are not supported on Windows XP system

---

## 1.9 SDK Return Value

All QCAP API functions can return following values.

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QCAP_RS_ERROR_OUT_OF_RESOURCE	Fail due to out of resource
QCAP_RS_ERROR_INVALID_DEVICE	Cannot open video or audio capture device
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_NON_SUPPORT	Fail due to encoder type doesn't support
QCAP_RS_ERROR_TIMEOUT	Fail due to timeout
QCAP_RS_ERROR_INVALID_ANIMATION_SCRIPT	Fail due to invalid animation script
QCAP_RS_ERROR_NO_SIGNAL_DETECTED	Fail due to no signal detected
QCAP_RS_ERROR_NEED_MORE_DATA	Fail due to needing more data information
QCAP_RS_ERROR_CONNECT_FAIL	Fail due to connecting fail
QCAP_RS_ERROR_FILE_ACCESS_FAIL	Fail due to file access fail
QCAP_RS_ERROR_NETWORK_ACCESS_FAIL	Fail due to network access fail
QCAP_RS_ERROR_FILE_IS_BOX_MOVED	Fail due to file is moved
QCAP_RS_ERROR_FRAME_IS_COPIED	Fail due to the frame is already copied

## 1.10 SDK Easy Programming Guide

QCAP SDK Quick Programming Guide for Hardware Compression Card Series 1.1.0

QCAP SDK Quick Programming Guide for Software Compression Card Series 1.1.0

This section will give users a first glance of a typical QCAP SDK program example.

The following samples include both **Software Encoder** and **Hardware encoder** (in blue). For user who have no hardware encoder in the capture card and use pure software encoder only, can just ignore the blue lines in the program samples.

### 1.10.1 CAPTURE DEVICE APIs

#### 1.10.1.1 Initialize Device APIs

```
QCAP_SET_SYSTEM_CONFIGURATION( ... );
QCAP_CREATE( "FH8735 PCI", 0, ... );

QCAP_REGISTER_FORMAT_CHANGED_CALLBACK( pDevice, on_format_changed_callback, ... );
QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK( pDevice, on_no_signal_detected_callback, ... );
QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK( pDevice, on_no_signal_removed_callback, ... );
QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK( pDevice, on_video_preview_callback, ... );
QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK( pDevice, on_audio_preview_callback, ... );
QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK( pDevice, 0, on_video_main_encoder_callback, ... );
QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK( pDevice, 1, on_video_sub_encoder_callback, ... );

QCAP_SET_VIDEO_DEINTERLACE_TYPE( pDevice, QCAP_SOFTWARE_DEINTERLACE_TYPE_BLENDING );
QCAP_SET_VIDEO_DEINTERLACE( pDevice, TRUE );
QCAP_SET_VIDEO_INPUT( pDevice, QCAP_INPUT_TYPE_AUTO );
QCAP_SET_AUDIO_INPUT( pDevice, QCAP_INPUT_TYPE_EMBEDDED_AUDIO );
QCAP_SET_AUDIO_VOLUME( pDevice, 100 );

QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX( pDevice, 0, ... );
QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX( pDevice, 1, ... );

QCAP_RUN( pDevice );
```



**Blue Lines** for Hardware encoder only!

#### 1.10.1.2 Uninitialize Device APIs

```
QCAP_STOP( pDevice );
QCAP_DESTROY( pDevice );
```



## 1.10.2 CHANNEL RECORD APIs

### 1.10.2.1 Start Channel Record APIs

#### Channel Record / Capture-Only Card

This is for the capture card that has no hardware encoder on it, so it can ONLY do software encode.

With Capture-Only capture card, the *iRecNum*=0..3 are reserved for software encoder to use.

```
QCAP_SET_VIDEO_RECORD_PROPERTY( m_pDevice , 0, ... );  
QCAP_SET_AUDIO_RECORD_PROPERTY( m_pDevice , 0, ... );  
QCAP_START_RECORD( pDevice , 0, "CHANNEL01.MP4" );
```



**Blue Lines** for Hardware encoder only!

#### Channel Record / Hardware Main Encoder

This is for the capture card that has Single-Stream (1 hardware encoder) on it.

With hardware encoder card, the *iRecNum*=0/1, is used by main/sub hardware encoder, respectively.

```
QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice , 0, QCAP_ENCODER_TYPE_HARDWARE, ... );  
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice , 0, QCAP_ENCODER_TYPE_SOFTWARE, ... );  
QCAP_START_RECORD( pDevice , 0, "CHANNEL01.MP4" );
```



**Blue Lines** for Hardware encoder only!

#### Channel Record / Hardware Sub Encoder

This is for the capture card that has Dual-Stream (2 hardware encoders) on it.

With hardware encoder card, the *iRecNum*=0/1, is used by main/sub hardware encoder, respectively.

```
QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice , 1, QCAP_ENCODER_TYPE_HARDWARE, ... );  
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice , 1, QCAP_ENCODER_TYPE_SOFTWARE, ... );  
QCAP_START_RECORD( pDevice , 1, "CHANNEL01.MP4" );
```



**Blue Lines** for Hardware encoder only!

## Channel Record / General Software Encoder

This is for the capture card that has hardware encoder on it, and it STILL can do general software encode.

With hardware encoder card, the *iRecNum*=0/1, is used by main/sub hardware encoder, respectively. So *iRecNum*=2/3 are reserved for software encoder to use.

```
QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice , 2 or 3, QCAP_ENCODER_TYPE_SOFTWARE, ... );  
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice , 2 or 3, QCAP_ENCODER_TYPE_SOFTWARE, ... );  
QCAP_START_RECORD( pDevice , 2 or 3, "CHANNEL01.MP4" );
```



**Blue Lines** for Hardware encoder only!

### 1.10.2.2 Stop Channel Record APIs

```
QCAP_STOP_RECORD ( pDevice, ... );
```

## 1.10.3 SHARE RECORD APIs

### 1.10.3.1 Start Share Record APIs

```
QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( 0, ... );
QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( 0, ... );
QCAP_START_SHARE_RECORD( 0, "SHARE01.MP4", dwFlags );
```



For Hardware Encoder user, calls **QCAP\_START\_SHARE\_RECORD()** with **QCAP\_RECORD\_FLAG\_ENCODE** flag cleared could free unused Software Encoder resources.

### 1.10.3.2 Set Share Record Data APIs

#### Set Share Record Data from Uncompression Buffer

```
QRETURN on_video_preview_callback ( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    if( share_record_state > 0 )
    {
        //SET VIDEO BUFFER
        QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );
    }
    ...
}

QRETURN on_audio_preview_callback ( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    if( share_record_state > 0 )
    {
        //SET AUDIO BUFFER
        QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );
    }
    ...
}
```

#### Set Share Record Data from Compression Buffer

```

QRETURN on_video_main_encoder_callback ( ..., BYTE * pStreamBuffer , ULONG nStreamBufferLen, ... )
{
    ...
    if( share_record_state > 0 )
        //SET VIDEO BUFFER (Compressed)
        QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER( ..., pStreamBuffer , nStreamBufferLen, ... );
    ...
}

QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    if( share_record_state > 0 )
        //SET AUDIO BUFFER
        QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );
    ...
}

```



**Blue Lines** for Hardware encoder only!

### 1.10.3.3 Stop Share Record APIs

```

QCAP_STOP_SHARE_RECORD ( 0 );

```

### 1.10.3.4 Share Record with Multi-Threading

```
DWORD WINAPI on_video_preview_callback_ex ( LPVOID params )
{
    ...
    HANDLE events [ 2 ] = { h_share_record_thread_stop_events[ 0 ],
                           h_share_record_buffer_ready_events [ 0 ] };
    while( TRUE ) {
        DWORD returns = WaitForMultipleObjects ( 2, events, FALSE, INFINITE );
        if( returns == (WAIT_OBJECT_0) ) { break ; }
        if( returns == (WAIT_OBJECT_0 + 1) )
        {
            EnterCriticalSection( h_share_record_access_critical_sections[ 0 ] );
            BYTE * po = share_record_buffers[ 0 ];
            ULONG sz = share_record_buffer_lengths[ 0 ];

            if ( share_record_state > 0 )
            {
                LeaveCriticalSection( h_share_record_access_critical_sections[ 0 ] );

                //SET VIDEO BUFFER
                QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER ( ..., po , sz, ... );
            }
            else {
                LeaveCriticalSection( h_share_record_access_critical_sections[ 0 ] );
            }
        }
    }
    ...
}

QRETURN on_video_preview_callback ( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    EnterCriticalSection( h_share_record_access_critical_sections[ 0 ] );
    if ( share_record_state > 0 )
    {
        share_record_buffers[ 0 ] = pFrameBuffer;
        share_record_buffer_lengths[ 0 ] = nFrameBufferLen;
        SetEvent( h_share_record_buffer_ready_events[ 0 ] );
    }
    LeaveCriticalSection( h_share_record_access_critical_sections[ 0 ] );
    ...
}
```

```

DWORD WINAPI on_audio_preview_callback_ex ( LPVOID params )
{
    ...
    HANDLE events [ 2 ] = { h_share_record_thread_stop_events [ 1 ],
                           h_share_record_buffer_ready_events [ 1 ] };

    while( TRUE )
    {
        DWORD returns = WaitForMultipleObjects ( 2, events, FALSE, INFINITE );
        if( returns == (WAIT_OBJECT_0) ) { break ; }
        if( returns == (WAIT_OBJECT_0 + 1) )
        {
            EnterCriticalSection( h_share_record_access_critical_sections[ 1 ] );
            BYTE * po = share_record_buffers[ 1 ];
            ULONG sz = share_record_buffer_lengths[ 1 ];
            if ( share_record_state > 0 )
            {
                LeaveCriticalSection( h_share_record_access_critical_sections[ 1 ] );

                //SET AUDIO BUFFER
                QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., po, sz, ... );
            }
            else
            {
                LeaveCriticalSection( h_share_record_access_critical_sections[ 1 ] );
            }
        }
    }
    ...
}

QRETURN on_audio_preview_callback ( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )
{
    ...
    EnterCriticalSection( h_share_record_access_critical_sections[ 1 ] );
    if ( share_record_state > 0 )
    {
        share_record_buffers[ 1 ] = pFrameBuffer;
        share_record_buffer_lengths[ 1 ] = nFrameBufferLen;
        SetEvent( h_share_record_buffer_ready_events[ 1 ] );
    }
    LeaveCriticalSection( h_share_record_access_critical_sections[ 1 ] );
    ...
}

```

## 1.10.4 BROADCAST APIs

### 1.10.4.1 Start Broadcast APIs

```
QCAP_CREATE_BROADCAST_RTSP_SERVER ( 0, ..., &pServer, ... );
QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer , ..., dwFlags );
QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY( pServer , ... );
QCAP_START_BROADCAST_SERVER ( pServer );
```



For Hardware Encoder user, calls **QCAP\_START\_SHARE\_RECORD()** with **QCAP\_RECORD\_FLAG\_ENCODE** flag cleared could free unused Software Encoder resources.

### 1.10.4.2 Set Broadcast Data APIs

### 1.10.4.3 Set Broadcast Data from Uncompression Buffer

```
QRETURN on_video_preview_callback( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    if( broadcast_server_state > 0 )
    {
        //SET VIDEO BUFFER
        QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ...
    );
    }
    ...
}

QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    if( broadcast_server_state > 0 )
    {
        //SET AUDIO BUFFER
        QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ...
    );
    }
    ...
}
```



#### 1.10.4.4 Set Broadcast Data from Compression Buffer

```
QRETURN on_video_main_encoder_callback ( ..., BYTE * pStreamBuffer, ULONG nStreamBufferLen , ... )
{
    ...
    if( broadcast_server_state > 0 )
        //SET VIDEO BUFFER (Compressed)
        QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER( ..., pStreamBuffer, nStreamBufferLen ,
    ... );
    ...
}

QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    if( broadcast_server_state > 0 )
        //SET AUDIO BUFFER
        QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ...
    );
    ...
}
```



**Blue Lines** for Hardware encoder only!

#### 1.10.4.5 Stop Broadcast APIs

```
QCAP_STOP_BROADCAST_SERVER ( pServer );
```

#### 1.10.4.6 Broadcast with Multi-Threading

```
DWORD WINAPI on_video_preview_callback_ex ( LPVOID params )
{
    ...
    HANDLE events [ 2 ] = { h_broadcast_server_thread_stop_events[ 0 ],
                           h_broadcast_server_buffer_ready_events [ 0 ] };
    while( TRUE )
    {
        DWORD returns = WaitForMultipleObjects ( 2, events, FALSE, INFINITE );
        if( returns == (WAIT_OBJECT_0) ) { break ; }
        if( returns == (WAIT_OBJECT_0 + 1) )
        {
            EnterCriticalSection( h_broadcast_server_access_critical_sections[ 0 ] );
            BYTE * po = broadcast_server_buffers[ 0 ];
            ULONG sz = broadcast_server_buffer_lengths[ 0 ];
            if ( broadcast_server_state > 0 )
            {
                LeaveCriticalSection( h_broadcast_server_access_critical_sections[ 0 ] );
                //SET VIDEO BUFFER
                QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., po, sz, ... );
            }
            else
            {
                LeaveCriticalSection( h_broadcast_server_access_critical_sections[ 0 ] );
            }
        }
    }
    ...
}

QRETURN on_video_preview_callback ( ..., BYTE * pFrameBuffer , ULONG nFrameBufferLen, ... )
{
    ...
    EnterCriticalSection( h_broadcast_server_access_critical_sections[ 0 ] );
    if ( broadcast_server_state > 0 )
    {
        broadcast_server_buffers[ 0 ] = pFrameBuffer;
        broadcast_server_buffer_lengths[ 0 ] = nFrameBufferLen;
        SetEvent( h_broadcast_server_buffer_ready_events[ 0 ] );
    }
    LeaveCriticalSection( h_broadcast_server_access_critical_sections[ 0 ] );
    ...
}
```

```

DWORD WINAPI on_audio_preview_callback_ex ( LPVOID params )
{
    ...
    HANDLE events [ 2 ] = { h_broadcast_server_thread_stop_events[ 1 ],
                           h_broadcast_server_buffer_ready_events [ 1 ] };
    while( TRUE )
    {
        DWORD returns = WaitForMultipleObjects ( 2, events, FALSE, INFINITE );
        if( returns == (WAIT_OBJECT_0) ) { break ; }
        if( returns == (WAIT_OBJECT_0 + 1) )
        {
            EnterCriticalSection( h_broadcast_server_access_critical_sections[ 1 ] );
            BYTE * po = broadcast_server_buffers[ 1 ];
            ULONG sz = broadcast_server_buffer_lengths[ 1 ];
            if ( broadcast_server_state > 0 )
            {
                LeaveCriticalSection( h_broadcast_server_access_critical_sections[ 1 ] );
                //SET AUDIO BUFFER
                QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., po, sz, ... );
            }
            else {
                LeaveCriticalSection( h_broadcast_server_access_critical_sections[ 1 ] );
            }
        }
    }
    ...
}

QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )
{
    ...
    EnterCriticalSection( h_broadcast_server_access_critical_sections[ 1 ] );
    if ( broadcast_server_state > 0 )
    {
        broadcast_server_buffers[ 1 ] = pFrameBuffer;
        broadcast_server_buffer_lengths[ 1 ] = nFrameBufferLen;
        SetEvent( h_broadcast_server_buffer_ready_events[ 1 ] );
    }
    LeaveCriticalSection( h_broadcast_server_access_critical_sections[ 1 ] );
    ...
}

```

## 1.10.5 CAPTURE CARD CUSTOM PROPERTY

For more custom device property programming, please refer to more QCAP SDK Capture Product Extra Programming Guide in **QCAP SDK\QCAP\DOC\PRODUCT MANUALS**. In there we give examples SC580 OSD text and picture, and SC510 HDCP protection status.

Example 1: SC580 OSD text in hardware encoder.

```
//STEP 1. Set OSD format

//STEP 2. Set OSD device custom property

QCAP_SET_DEVICE_CUSTOM_PROPERTY( pDevice, 929, 0x00000001 );

QCAP_SET_DEVICE_CUSTOM_PROPERTY( pDevice, 920, 0x00000000 );

CHAR path [] = "C:/OSD.TXT";

QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice, 921, (BYTE *) ( path ), strlen( path ) );
```

Example 2: SC580 OSD picture in hardware encoder.

```
ULONG params[ 4 ] = { 0, /*Picture index is 0 or 1 */
                     1, /*Picture start left position*/
                     1, /*Picture start top position*/
                     255 /*Picture transparent is from 0~255*/
};

CHAR path [] = "C:/WINDOWS/FH8735/Slimmer_64.bmp";

QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice, 970, ( BYTE * )params, sizeof( params ) );

QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice, 971, ( BYTE * )(path), strlen( path ) );
```



SC580 allow software to load one 8 bits (256 colors) BMP file into its board memory.

Example 3: SC510 video input's media content owns HDCP or MarcoVision protection.

```
QCAP_GET_DEVICE_CUSTOM_PROPERTY( pDevice, 202, &HDCP );

if( HDCP == 1 ) { RECORD_FUNCTION = DISABLE; }

if( HDCP == 0 ) { RECORD_FUNCTION = ENABLE; }
```

## 2 System Function API

---

### Introduction



This chapter provides the user an interface to get the information about the QCAP SDK framework version, capture card hardware capabilities, encoder/decoder information, and system configurations.

# 2.1 QCAP\_GET\_VERSION

## Introduction

The user can use this function to get major and minor software versions of QCAP SDK framework.

## Parameters

type	parameter	I/O	descriptions
ULONG *	pMajorVersion	OUT	Pointer of major version
ULONG *	pMinorVersion	OUT	Pointer of minor version

## Return values

QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QCAP_RS_ERROR_OUT_OF_RESOURCE	Fail due to out of resource
QCAP_RS_ERROR_INVALID_DEVICE	Cannot open video or audio capture device
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_NON_SUPPORT	Fail due to encoder type not supported
QCAP_RS_ERROR_TIMEOUT	Fail due to timeout
QCAP_RS_ERROR_INVALID_ANIMATION_SCRIPT	Fail due to invalid animation script
QCAP_RS_ERROR_NO_SIGNAL_DETECTED	Fail due to no signal detected
QCAP_RS_ERROR_NEED_MORE_DATA	Fail due to needing more data information
QCAP_RS_ERROR_CONNECT_FAIL	Fail due to connecting fail
QCAP_RS_ERROR_FILE_ACCESS_FAIL	Fail due to file access error
QCAP_RS_ERROR_NETWORK_ACCESS_FAIL	Fail due to network access problem
QCAP_RS_ERROR_FILE_IS_BOX_MOVED	Fail due to file is moved
QCAP_RS_ERROR_FRAME_IS_COPIED	Fail due to the frame is already copied

## Examples

*Example : Get the QCAP SDK framework software version*

```
ULONG nMajorVersion, nMinorVersion;

QCAP_GET_VERSION( &nMajorVersion, &nMinorVersion );
```



## 2.2 QCAP\_SET\_SYSTEM\_CONFIGURATION

### Introduction

The user can use this function to set global parameters of the QCAP SDK framework.

When user has variety color signal devices may have compatible problems.

The user can use *nSystemColorRangeType* to set the current source signal color range to fix "color offset in snapshot" issue in medical fields application.



Set *bEnableSCF* to true, then QCAP will recording to **SCF** format which is for Security Surveillance applications.

### Parameters

type	parameter	I/O	descriptions
BOOL	bEnableMultipleUsersAccess	IN	<b>default TRUE</b> If true QCAP will allow multiple users to access the device.
BOOL	bEnableVideoPreviewDevice	IN	<b>default TRUE</b> If true the video capture device will be enabled
BOOL	bEnableAudioPreviewDevice	IN	<b>default TRUE</b> If true the audio capture device will be enabled
BOOL	bEnableVideoHardwareMainEncoderDevice	IN	<b>default TRUE</b> If true the main hardware encoder will be enabled
BOOL	bEnableVideoHardwareSubEncoderDevice	IN	<b>default TRUE</b> If true the sub hardware encoder will be enabled
ULONG	nAutoInputDetectionTimeout	IN	<b>default 3000</b> Specify the timeout duration of the auto input detection in ms
BOOL	bEnableSCF	IN	<b>default FALSE</b> If true the <b>SCF</b> file recording function will be turned on
CHAR *	pszDB3	IN	<b>default NULL</b> Specify the file path of SQL database name ( path + filename ), the default path is local folder
BOOL	bEnableAsyncBackgroundSnapshot	IN	<b>default FALSE</b> If true snapshot function will be moved from preview callback into another thread
BOOL	bEnableEnhancedVideoRenderer	IN	<b>default TRUE</b> If true QCAP will enable the DirectShow's Enhanced Video Renderer Filter
BOOL	bEnableSystemTimeCallback	IN	<b>default FALSE</b> If true the preview's and encoder's callbacks will return one local system time at dSampleTime
BOOL	bEnableFileRepairFunction	IN	<b>default TRUE</b> If true the recorded video file will have self-repaired ability
BOOL	bEnableNewRTSPLibrary	IN	



## 2.3 QCAP\_QUERY\_ENCODER\_TYPE\_CAP

### Introduction

The user can use this function to query encoder type and its capabilities.  
We suggest user query the GPU Encoder availability before using GPU encoder functionality.

### Parameters

type	parameter	I/O	descriptions
ULONG	nEncoderType	IN	Specify query the encoder type: QCAP_ENCODER_TYPE_SOFTWARE QCAP_ENCODER_TYPE_HARDWARE QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK QCAP_ENCODER_TYPE_AMD_STREAM QCAP_ENCODER_TYPE_NVIDIA_CUDA QCAP_ENCODER_TYPE_NVIDIA_NVENC
ULONG	nEncoderFormat	IN	Specify query encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Query the Encoder*

```
QCAP_QUERY_ENCODER_TYPE_CAP( nEncoderType, nEncoderFormat );
```

## 2.4 QCAP\_QUERY\_DECODER\_TYPE\_CAP

### Introduction

The user can use this function to query the decoder type & capabilities.

### Parameters

type	parameter	I/O	descriptions
ULONG	nDecoderType	IN	Specify query the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
ULONG	nDecoderFormat	IN	Specify query encoder format: QCAP_DECODER_FORMAT_MPEG2 QCAP_DECODER_FORMAT_H264 QCAP_DECODER_FORMAT_H264_3D QCAP_DECODER_FORMAT_H264_VC QCAP_DECODER_FORMAT_RAW QCAP_DECODER_FORMAT_H265

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Query the decoder capabilities*

```
QCAP_QUERY_DECODER_TYPE_CAP( nDecoderType, nDecoderFormat );
```

## 2.5 QCAP\_QUERY\_ENCODER\_STATUS

### Introduction

The user can use this function to check the current number of encoders and its availabilities.

### Parameters

type	parameter	I/O	descriptions
ULONG	nEncoderType	IN	Specify query the encoder type: QCAP_ENCODER_TYPE_SOFTWARE QCAP_ENCODER_TYPE_HARDWARE QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK QCAP_ENCODER_TYPE_AMD_STREAM QCAP_ENCODER_TYPE_NVIDIA_CUDA QCAP_ENCODER_TYPE_NVIDIA_NVENC
ULONG *	pExistInstances	OUT	Returns the current number of encoders

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Query the different type of encoders and availabilities*

```
QCAP_QUERY_ENCODER_STATUS( nEncoderType, &pExistInstances );
```

C

# 3 Device Function API

---

## Introduction

In order to make capture device working properly, the device function APIs must be called in a certain manner. This chapter includes the following topics that will help you to get the capturing task done: Major functions, Callback functions, Data Access Functions, Input Property Functions, Format Property Functions, Video Property Functions, Audio Property Functions, Advance Property Functions, Helper Functions.



## QCAP Enhanced Functions

In the following chapters, there are functions with original / enhanced version:

- "**API\_EX()**" that means an extended version of **API()** with a more detailed interface.
- Some function parameters were only for the enhanced version will be noted in the parameter table.

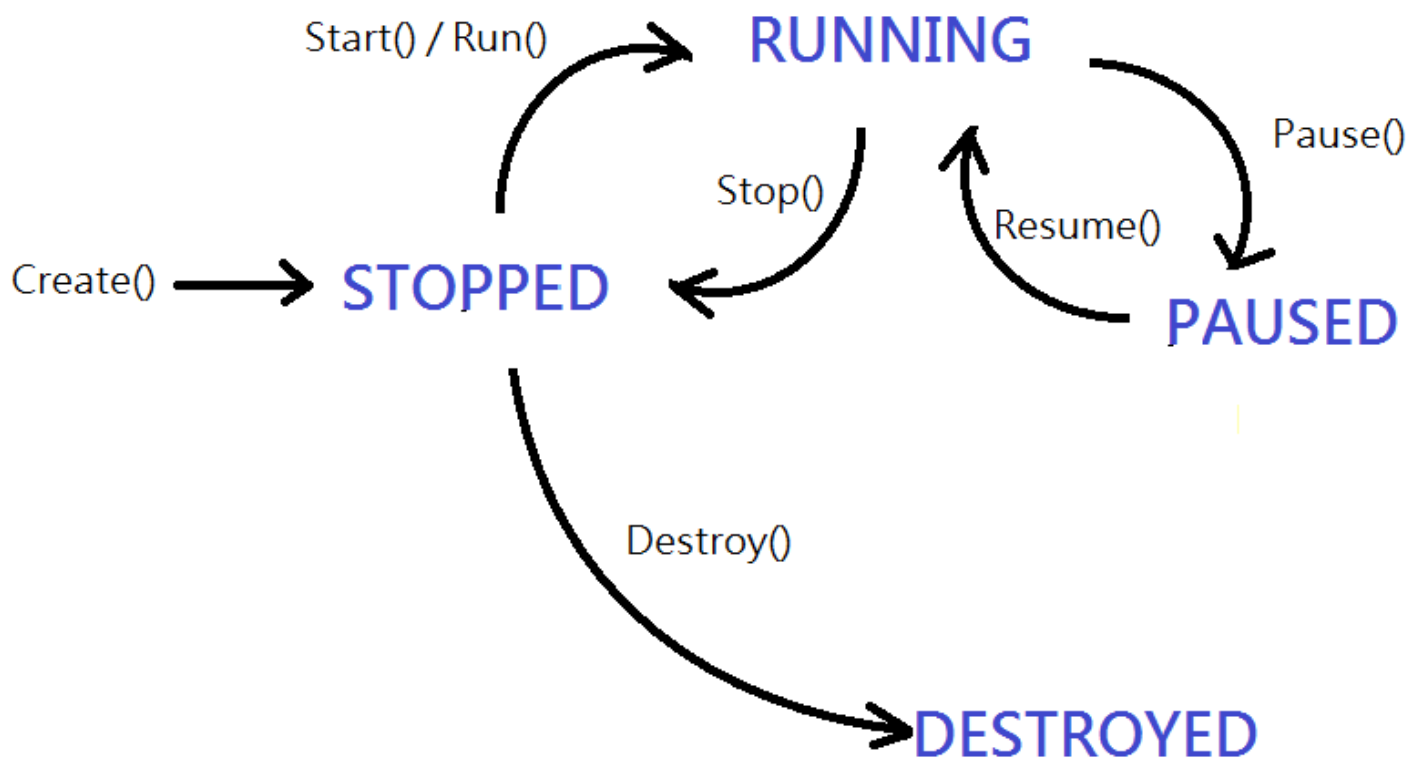
# 3.1 Major Functions

## Introduction



The first task of capture applications creates the device handle, set starting capturing video and preview on a window handle. Then use data callback function to manipulate the data from each frame, and event callback for signal changes. The user can also use helper functions to query / enumeration of the devices. After all done then release the resources to the system.

Here are the life cycle of the device create functions:





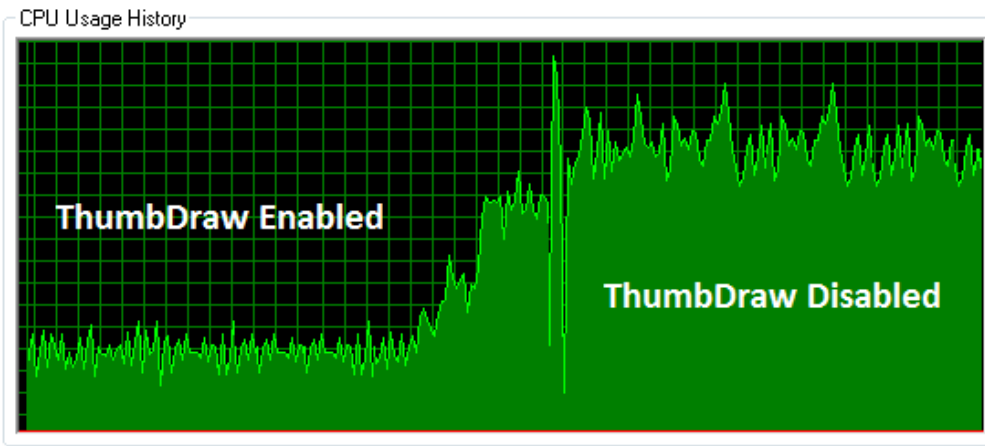
### 3.1.1 QCAP\_CREATE

#### Introduction

Before using a capture device, you need to create a device handle by passing the correct device name string to **QCAP\_CREATE()**. For multiple capture devices, you need to call multiple times by setting different device index. To run multiple capture devices simultaneously between processes are safe synchronized.

The preview video could be drawn on the given window handle. Without preview window, the capture device enables in background mode. If the *bMaintainAspectRatio* is true, the ratio of the width to the height of a video will keep original ratio no matter the windows size is, or the preview video will be scaled to any window size.

The **ThumbDraw** option can reduce the CPU usage to obtain the best rendering performance dramatically, especially when the display window size is smaller than the input video size. If return values is negative an error occurred, please remember to release capture device by calling **QCAP\_DESTROY()**.



#### Parameters

type	parameter	I/O	descriptions
CHAR *	pszDevName	IN	Device Name, currently available are: "DC1150 USB" is for PD652 and PD652.3D "QP0204 USB" is for PD5A0 <sup>New</sup> "UB658G USB" is for UB658G "CY3014 USB" is for UB530 and UB530G "TW6802 PCI" is for SC200,SC300,SC230,SC330 "CX2581 PCI" is for SC310,SC340 "CX2385 PCI" is for SC350 "AH8400 PCI" is for SC290 and SC390 "FH8735 PCI" is for SC2A0 and SC3A0 "TW5864 PCI" is for SC2B0 and SC3B0 "SA7160 PCI" is for SC500, SC510, and TB510 "FH8735 PCI" is for SC580 "TW2809 PCI" is for SC590 "QP0203 PCI" is for SC540 and SC5A0 <sup>New</sup> "MZ0380 PCI" is for SC3C0 and SC5C0 <sup>New</sup> "CAMERA" is for USB CAMERA "SOUNDCARD" is for Sound Card "DESKTOP" is for Desktop
UINT	iDevNum	IN	Device index specified according to pszDevName, start from 0
HWND	hAttachedWindow	IN	Handle of the window to show the preview video
PVOID	ppDevice	OUT	Handle of the capture card object
BOOL	bThumbDraw	IN	Enable/Disable the ThumbDraw renderer
BOOL	bMaintainAspectRatio	IN	To Enable/Disable the maintain aspect ratio

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

Example : create a capture device handle

```
// create a capture device "FH8735 PCI" #0
// with ThumbDraw enabled, no maintain aspect ratio

if( QCAP_CREATE( "FH8735 PCI", 0, hWnd, &hVideoDevice, TRUE, FALSE ) != QCAP_RS_SUCCESSFUL )

    printf("ERROR: creating capture device object.\n");

else

    printf("hVideoDevice is the created handle\n");
```

3.1.2 QCAP\_RUN

3.1.3 QCAP\_RUN\_EX

Introduction

When user call **QCAP\_RUN()** it will start the device to capture video and audio signals.  
To stop video capturing just call **QCAP\_STOP()**.

After **QCAP\_RUN()** called the device start capturing. The Audio/Video preview callback functions will continue to output data stream until the device detects video signal lost. For Advanced user who wants a device to continues output data stream when signal lost, please use **QCAP\_RUN\_EX()** by setting the *bStopAfterSignalRemoved* parameter to false.

Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Device object to release
BOOL	bStopAfterSignalRemoved	IN	default TRUE Only in QCAP_RUN_EX()

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

Example : start video capturing and preview on windows handle

```
QCAP_RUN( pDevice ); //Start capturing

QCAP_RUN_EX( pDevice,
             TRUE ); //extended. Start capturing. If signal loss then pauses data stream output.
```

## 3.1.4 QCAP\_STOP

### Introduction

This function is corresponding to **QCAP\_RUN()** function. Call this function if you have done video capturing. This function won't return all resources just stop a device from capturing the video.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Device object to release

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Stop video capturing*

```
QCAP_STOP( pDevice );
```

## 3.1.5 QCAP\_DESTROY

### Introduction

This function is corresponding **QCAP\_CREATE()**. Once you're done make sure to correctly Destroy the capture device object to make sure that all allocated resources get freed and the device is correctly released. If the user forgets to call **QCAP\_STOP()** before calling this function, it will auto call **QCAP\_STOP()** internally if it detects device is still in capturing.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Device object to release

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

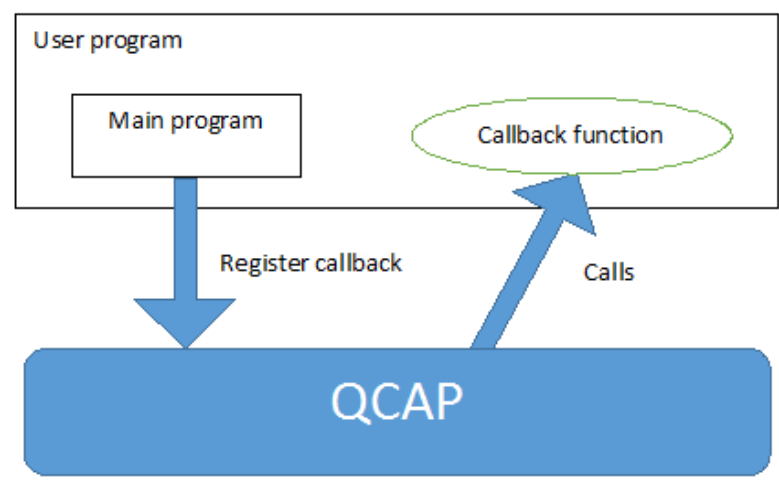
### Examples

*Example : destroy a capture device handle*

```
QCAP_DESTROY( pDevice );
```

# 3.2 Device Callback Functions

## Introduction



The callback function is a pointer to a user-defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes. Event callbacks will notify the device status change, a Data Callbacks usually provide a frame data. For device callback functions to work, you need to register it with a user defined function before **QCAP\_RUN()**. A user defined callback function will be called when an event is occurred, for example when capture frame is ready, signal lost...etc.

Here are the list of the major callback functions behaviors and the calling sequences result:

Calling sequence	SD No Signal	SD Connect Signal	SD Remove Signal	HD No Signal	HD Connect Signal	HD Remove Signal
<b>QCAP_RUN()</b>	FC/SR	FC	SR	NS	FC	SR
1. <b>SET DEFAULT FORMAT</b> 2. <b>QCAP_RUN()</b>	FC/ SET_DEFAULT_FORMAT FC/SR	FC	SR	NS/ SET_DEFAULT_FORMAT FC/SR	FC	SR
<b>QCAP_RUN_EX(pDevice,FALSE)</b>	FC/SR	FC	SR	NS	FC	SR
1. <b>SET DEFAULT FORMAT</b> 2. <b>QCAP_RUN_EX(pDevice,FALSE)</b>	FC/ SET_DEFAULT_FORMAT FC/SR	FC	SR	NS/ SET_DEFAULT_FORMAT FC/SR	FC	SR

Note:  
**NS:** No Signal Detected Callback  
**SR:** Signal Removed Callback  
**FC:** Format Changed Callback

this section contains how to register callback functions and its purpose for:

Register functions	Callback functions
<b>Event Callbacks</b>	
QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK	PF_NO_SIGNAL_DETECTED_CALLBACK
QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK	PF_SIGNAL_REMOVED_CALLBACK
QCAP_REGISTER_FORMAT_CHANGED_CALLBACK	PF_FORMAT_CHANGED_CALLBACK
<b>Data Callbacks</b>	
QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK	PF_VIDEO_PREVIEW_CALLBACK
QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK	PF_AUDIO_PREVIEW_CALLBACK
QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK	PF_VIDEO_HARDWARE_ENCODER_CALLBACK
QCAP_REGISTER_VIDEO_VERTICAL_ANCILLARY_DATA_CALLBACK	PF_VIDEO_VERTICAL_ANCILLARY_DATA_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

## 3.2.1 Event Callback Functions

### 3.2.1.1 QCAP\_REGISTER\_NO\_SIGNAL\_DETECTED\_CALLBACK

#### Introduction

Just like video preview callback, you can also register a *PF\_NO\_SIGNAL\_DETECTED\_CALLBACK* function to detect input signal availability. When capture device cannot detect the signal from the current input source, the user-defined callback function will be called. The user can implement its own function to handle no signal situation.



This function need to be called right after **QCAP\_CREATE()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the device handle
<i>PF_NO_SIGNAL_DETECTED_CALLBACK</i>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### *PF\_NO\_SIGNAL\_DETECTED\_CALLBACK*

#### Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	The device handle that callbacks
ULONG	nVideoInput	video input source of the device
ULONG	nAudioInput	audio input source of the device
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

#### Examples

*Example: Register a callback function for no signal detection*

```
// NO SIGNAL DETECTED CALLBACK
QRETURN no_signal_detected(  PVOID pDevice,
                             ULONG nVideoInput,
                             ULONG nAudioInput,
                             PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_NO_SIGNAL_DETECTED_CALLBACK pCB = no_signal_detected;

    QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK( pDevice, pCB, pUserData );
}
```

### 3.2.1.2 QCAP\_REGISTER\_SIGNAL\_REMOVED\_CALLBACK

#### Introduction

Just like no signal detection, you can also register a **PF\_SIGNAL\_REMOVED\_CALLBACK** function to detect signal lost. Unlike NO\_SIGNAL\_DETECTED\_CALLBACK this callback only called when any reason (e.g. unplug the cable) causes the signal change state to loss. The user can implement its own function to handle a signal lost situation.



This function need to be called right after **QCAP\_CREATE()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the device handle
<b>PF_NO_SIGNAL_DETECTED_CALLBACK</b>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_SIGNAL\_REMOVED\_CALLBACK

#### Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	The device handle that callbacks
ULONG	nVideoInput	video input source of the device
ULONG	nAudioInput	audio input source of the device
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

#### Examples

*Example: Register a callback function for signal removed situation*

```
// NO SIGNAL DETECTED CALLBACK
QRETURN signal_removed_detected( PVOID pDevice,
                                ULONG nVideoInput,
                                ULONG nAudioInput,
                                PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SIGNAL_REMOVED_CALLBACK pCB = signal_removed_detected;

    QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK( pDevice, pCB, pUserData );
}
```

### 3.2.1.3 QCAP\_REGISTER\_FORMAT\_CHANGED\_CALLBACK

### 3.2.1.4 QCAP\_REGISTER\_FORMAT\_CHANGED\_CALLBACK\_EX

#### Introduction

You can register a **PF\_FORMAT\_CHANGED\_CALLBACK** function to detect signal format changed. Note that the callback function will be called:

- When the input signal is detected
- When input format is changed
- When input format has no source - The *Width*, *Height* parameters will set to 0 to indicate no source.

This callback function can provide video information (such as input sources, video resolution, interleaved, frames per second), and audio information( input source, channel numbers, sample format, and frequency). The user can implement its own function to handle the input format changed situation.



This function need to be called right after **QCAP\_CREATE()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the device handle
<b>PF_NO_SIGNAL_DETECTED_CALLBACK</b>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_FORMAT\_CHANGED\_CALLBACK

### PF\_FORMAT\_CHANGED\_CALLBACK\_EX

#### Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	The device handle that callbacks
ULONG	nVideoInput	The video input source of the device
ULONG	nAudioInput	The audio input source of the device
ULONG	nVideoWidth	The video frame width
ULONG	nVideoHeight	The video frame height
BOOL	bVideosInterleaved	The video interleaved flag
double	dVideoFrameRate, or dVideoNativeFrameRate	The native video frames per second
double	dVideoGrabFrameRate	The grabber video frames per second <b>Only in PF_FORMAT_CHANGED_CALLBACK_EX()</b>
ULONG	nAudioChannels	The total audio channels (e.g. stereo or mono)
ULONG	nAudioBitsPerSample	the audio bits per sample (e.g.8bits, 16bits)
ULONG	nAudioSampleFrequency	the audio sample frequency (e.g.44kHz, 16kHz)
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

#### Examples



*Example: Register a callback function to detect input format changed*

```
// FORMAT CHANGED CALLBACK
QRETURN signal_format_changed( PVOID pDevice,
                               ULONG nVideoput,
                               ULONG nAudioPut,
                               ULONG nVideoWidth,
                               ULONG nVideoHeight,
                               BOOL bVideoIsterleaved,
                               double dVideoFrameRate,
                               ULONG nAudioChannels,
                               ULONG nAudioBitsPerSample,
                               ULONG nAudioSampleFrequency,
                               PVOID pUserData );
{
    if ( nVideoWidth == 0 || nVideoHeight == 0 )
        printf("NO SOURCE\n");

    return QCAP_RT_OK;
}

void test_callback()
{
    PF_FORMAT_CHANGED_CALLBACK pCB = signal_format_changed;

    QCAP_REGISTER_FORMAT_CHANGED_CALLBACK( pDevice, pCB, pUserData );
}
```

## 3.2.2 Data Callback Functions

### 3.2.2.1 QCAP\_REGISTER\_VIDEO\_PREVIEW\_CALLBACK

#### Introduction

To register this callback right after **QCAP\_CREATE()**, then the preview video could be shown in a window after **QCAP\_RUN()** is called. You can then register a **PF\_VIDEO\_PREVIEW\_CALLBACK** function to manipulate video framebuffer when each frame is captured. When the callback function is called, you can get the framebuffer pointer in YUY2/YV12 video format (depends on devices). You can directly modify the video framebuffer (e.g. alpha blending) and the result will directly effect on both recording file and window display. The return value of a callback function can decide the QCAP next action.

For programming language has no callback mechanism, please use **QCAP\_COPY\_VIDEO\_PREVIEW\_BUFFER()** instead.

If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.



This function need to be called right after **QCAP\_CREATE()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the capture device handle
<b>PF_VIDEO_PREVIEW_CALLBACK</b>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_VIDEO\_PREVIEW\_CALLBACK

#### Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	The device that callbacks
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	pointer to video framebuffer The video format for each device are: <b>YUY2:</b> "AH8400 PCI", "DC1150 USB", "CY3014 USB", "TW6802 PCI", "CX2581 PCI", "SA7160 PCI" <b>YV12:</b> "UB658G USB", "QP0204 USB", "FH8735 PCI", "TW5864 PCI", "FH8735 PCI", "TW2809 PCI", "QP0203 PCI", "MZ0380 PCI"
ULONG	nFrameBufferLen	pointer to buffer length
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	return value to drop the framebuffer (will not display & record) engine on the window
<b>QCAP_RT_SKIP_RECORD_NUM_00..03</b>	To select which channel record number to skip the framebuffer
<b>QCAP_RT_SKIP_DISPLAY</b> <b>QCAP_RT_SKIP_CLONE_DISPLAY</b>	return value let video preview & clone video won't display
<b>QCAP_RT_RESET_RECORD_NUM_00..03</b>	To select which channel to reset framebuffer

## Examples

*Example: Register a video preview callback to a user-defined function*

```
// PREVIEW VIDEO CALLBACK FUNCTION
QRETURN video_preview_callback( PVOID pDevice,
                                double dSampleTime,
                                BYTE* pFrameBuffer,
                                ULONG nFrameBufferLen,
                                PVOID pUserData)
{
    if( pFrameBuffer==NULL || nFrameBufferLen==0 ) printf("NO DATA");

    //process video data here
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_PREVIEW_CALLBACK pCB = video_preview_callback;

    QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK( pDevice, pCB, pUserData );
}
```

### 3.2.2.2 QCAP\_REGISTER\_AUDIO\_PREVIEW\_CALLBACK

#### Introduction

Just like video preview callback, you can also register a **PF\_AUDIO\_PREVIEW\_CALLBACK** function to manipulate audio framebuffer when each frame is ready. Call this function to register an user-defined audio callback function. The format of callback framebuffer is uncompressed PCM audio data. You can directly modify the framebuffer (e.g. low-pass filter) and the result will directly effect on both recording file and sound playing. The return value of a callback function can decide the QCAP next action.

For programming language has no callback mechanism, please use **QCAP\_COPY\_AUDIO\_PREVIEW\_BUFFER()** instead.

If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.



This function need to be called right after **QCAP\_CREATE()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the device handle
<b>PF_AUDIO_PREVIEW_CALLBACK</b>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_AUDIO\_PREVIEW\_CALLBACK

#### Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	The device that callbacks
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	pointer to audio framebuffer The audio format for each device are uncompressed PCM audio data
ULONG	nFrameBufferLen	pointer to buffer length
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and its sound will be muted
<b>QCAP_RT_SKIP_RECORD_NUM_00..03</b>	To select which channel record number to skip the framebuffer
<b>QCAP_RT_SKIP_DISPLAY</b> <b>QCAP_RT_SKIP_CLONE_DISPLAY</b>	return value let the <b>PCM</b> audio uncompressed data and clone data will not be played on the window
<b>QCAP_RT_RESET_RECORD_NUM_00..03</b>	To select which channel to reset framebuffer

#### Examples

*Example: Register an audio preview callback to a user-defined function*

```
// PREVIEW AUDIO CALLBACK FUNCTION
QRETURN audio_preview_callback( PVOID pDevice,
                                double dSampleTime,
                                BYTE* pFrameBuffer,
                                ULONG nFrameBufferLen,
                                PVOID pUserData)
{
    if( pFrameBuffer==NULL || nFrameBufferLen==0 ) printf("NO DATA");

    //process audio data here
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_AUDIO_PREVIEW_CALLBACK pCB = audio_preview_callback;

    QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK( pDevice, pCB, pUserData );
}
```

### 3.2.2.3 QCAP\_REGISTER\_VIDEO\_HARDWARE\_ENCODER\_CALLBACK

#### Introduction

For those capture device has hardware encoder supported, you can register a **PF\_VIDEO\_HARDWARE\_ENCODER\_CALLBACK** function to receive hardware encoder event. This callback function will provide video hardware compressed **H.264** data stream to a user.

If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.



This function need to be called right after **QCAP\_CREATE()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the device handle
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters (starting from 0)
<b>PF_VIDEO_HARDWARE_ENCODER_CALLBACK</b>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_VIDEO\_HARDWARE\_ENCODER\_CALLBACK

type	parameter	callback descriptions
PVOID	pDevice	the device handle that callbacks
UINT	iRecNum	Indicates the recorder slot (starting from 0)
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	pointer to encoder <b>H.264</b> framebuffer
ULONG	nStreamBufferLen	pointer to buffer length
BOOL	bIsKeyFrame	Indicates for keyframe (I-frame), else P-frame
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The stream buffer of this frame will be dropped and will not send to the channel recorder

#### Examples

*Example: Register a callback function to receive hardware encoder event*

```
// VIDEO HARDWARE ENCODER CALLBACK
QRETURN hardware_encoder_callback( PVOID pDevice,
                                   UINT iRecNum,
                                   double dSampleTime,
                                   BYTE * pStreamBuffer,
                                   ULONG nStreamBufferLen,
                                   BOOL bIsKeyFrame,
                                   PVOID pUserData )
{
    if( pStreamBuffer==NULL || nStreamBufferLen==0 ) printf("NO DATA");

    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_HARDWARE_ENCODER_CALLBACK pCB = hardware_encoder_callback;

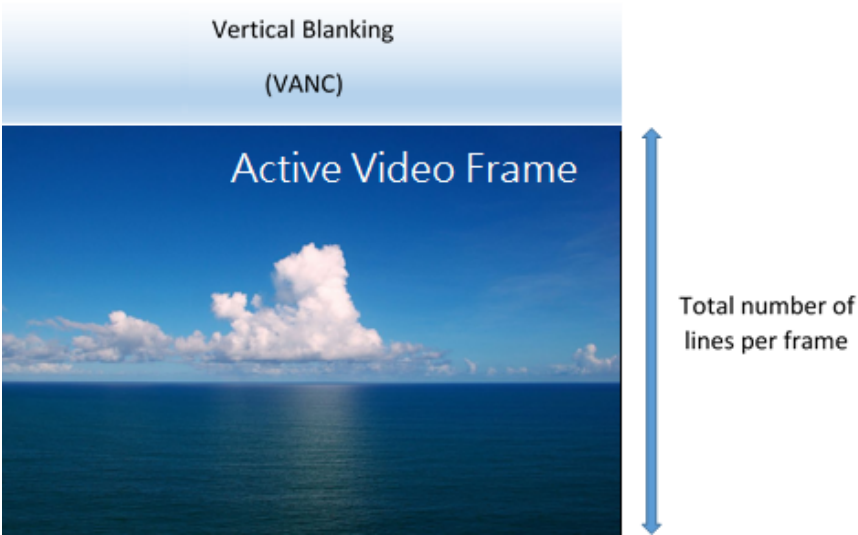
    QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK( pDevice, pCB, pUserData );
}
```

3.2.2.4 QCAP\_REGISTER\_VIDEO\_VERTICAL Ancillary Data Callback

Introduction

For those user wants to get VANC information before the video signal, you can register a **PF\_VIDEO\_VERTICAL Ancillary Data Callback** function. This callback function will retrieve current video vertical ancillary data grabbed lines. For example, if user called **QCAP\_SET\_VIDEO\_VERTICAL Ancillary Data Grabbed Lines()** to set grabbed nLines=8 in YUY2 video format, this callback function will provide ( 1920 \* 2 ) \* 8 Lines data for developer to use.

If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.



This function need to be called right after **QCAP\_CREATE()**.

Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Specify the device handle
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters (starting from 0)
<b>PF_VIDEO_VERTICAL Ancillary Data Callback</b>	pCB	IN	Specify the callback function
PVOID	pUserData	IN	Pointer to custom user data

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**PF\_VIDEO\_VERTICAL Ancillary Data Callback**

Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	The device that callbacks
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	pointer to video framebuffer
ULONG	nFrameBufferLen	pointer to buffer length
PVOID	pUserData	Pointer to custom user data



### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

### Examples

*Example: Register a callback function to retrieve video vertical ancillary data*

```
// VIDEO VERTICAL ANCILLARY DATA CALLBACK
QRETURN vertical_ancillary_data_callback( PVOID pDevice,
                                          double dSampleTime,
                                          BYTE * pFrameBuffer,
                                          ULONG nFrameBufferLen,
                                          PVOID pUserData )
{
    if( pFrameBuffer==NULL || nFrameBufferLen==0 ) printf("NO DATA");

    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_VERTICAL AncillaryDataCallback = vertical_ancillary_data_callback;

    QCAP_REGISTER_VIDEO_VERTICAL AncillaryDataCallback( pDevice, pCB, pUserData );
}
```

# 3.3 Data Access Functions

## Introduction

QCAP API has a lot of callback functions to retrieve the audio/video captured result, for languages without any callbacks mechanism like (e.g. LabView), this chapter provides functions that helps to retrieve the data buffers by polling them.

### 3.3.1 QCAP\_COPY\_VIDEO\_PREVIEW\_BUFFER

#### Introduction

For user don't use the callback to retrieve data, it can copy video preview framebuffer to a user area. It also returns the SampleTime of the frame copied. Then sample time is useful to know whether the frame is identical to last copy or not.



This function must be started after **QCAP\_RUN()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
double *	pSampleTime	OUT	Pointer to the frame sampling time
ULONG	nColorSpaceType	IN	Specify the color space type
VOID *	pFrameBuffer	IN	Pointer to the framebuffer
ULONG *	pFrameBufferLen	IN/OUT	Pointer to the input buffer length

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Copy the audio/video preview buffer by polling*

```
VOID *VideoBuffer = NULL, *AudioBuffer = NULL;

ULONG VideoBufferLen = 0, AudioBufferLen = 0;

double VideoTime = 0.0f, AudioTime = 0.0f;

QCAP_COPY_VIDEO_PREVIEW_BUFFER( pDevice, &VideoTime, QCAP_COLORSPACE_TYEP_YUY2, VideoBuffer, &VideoBufferLen );

QCAP_COPY_AUDIO_PREVIEW_BUFFER( pDevice, &AudioTime, AudioBuffer, &AudioBufferLen );
```

### 3.3.2 QCAP\_COPY\_AUDIO\_PREVIEW\_BUFFER

#### Introduction

For user don't use the callback to retrieve data, it can copy audio preview framebuffer to a user area. It also returns the SampleTime of the frame copied. Then sample time is useful to know whether the frame is identical to last copy or not.



This function must be started after **QCAP\_RUN()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
double *	pSampleTime	OUT	Pointer to the frame sampling time
VOID *	pFrameBuffer	IN	Pointer to the framebuffer
ULONG *	pFrameBufferLen	IN/OUT	Pointer to the input buffer length

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Copy the audio preview buffer by polling*

```
VOID *VideoBuffer = NULL, *AudioBuffer = NULL;

ULONG VideoBufferLen = 0, AudioBufferLen = 0;

double VideoTime = 0.0f, AudioTime = 0.0f;

QCAP_COPY_VIDEO_PREVIEW_BUFFER( pDevice, &VideoTime, QCAP_COLORSPACE_TYEP_YUY2, VideoBuffer, &VideoBufferLen );

QCAP_COPY_AUDIO_PREVIEW_BUFFER( pDevice, &AudioTime, AudioBuffer, &AudioBufferLen );
```

### 3.3.3 QCAP\_LOCK\_VIDEO\_PREVIEW\_BUFFER

### 3.3.4 QCAP\_LOCK\_AUDIO\_PREVIEW\_BUFFER

#### Introduction

For user don't use the callback to retrieve data, it locks the audio/video data and return the pointer of audio/video preview framebuffer for a user without copy it. It also returns the SampleTime of the frame. This function behavior is just like *PF\_VIDEO\_PREVIEW\_CALLBACK* / *PF\_AUDIO\_PREVIEW\_CALLBACK*. The user must call unlock function after it's done.



This function must be started after **QCAP\_RUN()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
double *	pSampleTime	OUT	Pointer to the sample time
VOID **	ppFrameBuffer	OUT	Pointer to retrieve framebuffer pointer
ULONG *	pFrameBufferLen	OUT	Pointer to the input buffer length

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Lock audio/video preview buffer*

```
VOID *VideoBuffer = NULL, *AudioBuffer = NULL;

ULONG VideoBufferLen = 0, AudioBufferLen = 0;

double VideoTime = 0.0f, AudioTime = 0.0f;

QCAP_LOCK_VIDEO_PREVIEW_BUFFER( pDevice, &VideoTime, &VideoBuffer, &VideoBufferLen );

QCAP_LOCK_AUDIO_PREVIEW_BUFFER( pDevice, &AudioTime, &AudioBuffer, &AudioFrameBufferLen );

//access the framebuffer directly

memset( VideoBuffer, 0, 100);

memset( AudioBuffer, 0, 100);

QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER( pDevice );

QCAP_UNLOCK_AUDIO_PREVIEW_BUFFER( pDevice );
```

### 3.3.5 QCAP\_UNLOCK\_VIDEO\_PREVIEW\_BUFFER

### 3.3.6 QCAP\_UNLOCK\_AUDIO\_PREVIEW\_BUFFER

#### Introduction

For user don't use the callback to retrieve data, can use this function to unlock audio/video data buffer.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Unlock audio/video preview buffer*

```
QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER( pDevice );

QCAP_UNLOCK_AUDIO_PREVIEW_BUFFER( pDevice );
```

# 3.4 Input Property Functions

## Introduction

This section contains device property functions to control audio/video capture inputs. For Video input, the sources can be either many of HDMI, SDI, COMPOSITE, DVI\_D, RGB, VGA...etc. with AUTO signal source detection for convenient. For Audio input, the source can be embedded audio within HDMI/SDI, or the system sound device to input the sound signal.

## 3.4.1 QCAP\_SET\_VIDEO\_INPUT

### Introduction

The user can use this function to set video input source. if *nInput* set to QCAP\_INPUT\_TYPE\_AUTO, video source will start auto detection of the signal source. For each input source, it takes 3 seconds to detect signal. The user can use QCAP\_SET\_SYSTEM\_CONFIGURATION to modify the duration of signal detection.



If the signal is unstable, we suggest to set the input source directly.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nInput	IN	Specify input video source: QCAP_INPUT_TYPE_COMPOSITE, QCAP_INPUT_TYPE_SVIDEO, QCAP_INPUT_TYPE_HDMI, QCAP_INPUT_TYPE_DVI_D, QCAP_INPUT_TYPE_COMPONENTS, QCAP_INPUT_TYPE_YCBCR, QCAP_INPUT_TYPE_DVI_A, QCAP_INPUT_TYPE_RGB, QCAP_INPUT_TYPE_VGA, QCAP_INPUT_TYPE_SDI, QCAP_INPUT_TYPE_DISPLAY_PORT, QCAP_INPUT_TYPE_AUTO

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the video input to HDMI source*

```
QCAP_SET_VIDEO_INPUT( pDevice, QCAP_INPUT_TYPE_HDMI );
```

## 3.4.2 QCAP\_GET\_VIDEO\_INPUT

### Introduction

The user can use this function to get current video input source.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pInput	OUT	Pointer to input video source

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get current input source*

```
ULONG nInput=0;

QCAP_GET_VIDEO_INPUT( pDevice, &nInput );
```

### 3.4.3 QCAP\_SET\_AUDIO\_INPUT

#### Introduction

The user can use this function to set audio input source. The embedded audio input means the HDMI or SDI audio via the capture device.

The sound card means the sound input device from the computer system. The user can use this function to change to sound card input, or create independent sound card device by QCAP\_CREATE( "SOUNDCARD", ... ).

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nInput	IN	Specify input audio input source: QCAP_INPUT_TYPE_EMBEDDED_AUDIO QCAP_INPUT_TYPE_LINE_IN QCAP_INPUT_TYPE_SOUNDCARD_MICROPHONE QCAP_INPUT_TYPE_SOUNDCARD_LINE_IN

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example 1: Select the embedded audio within the HDMI or SDI*

```
QCAP_SET_AUDIO_INPUT( pDevice, QCAP_INPUT_TYPE_EMBEDDED_AUDIO );
```

*Example 2: Create SOUNDCARD to be the audio input*

```
QCAP_CREATE( "SOUNDCARD", 0, hWnd, &pDevice, TRUE );

QCAP_SET_AUDIO_INPUT( pDevice, QCAP_INPUT_TYPE_SOUNDCARD_LINE_IN );

QCAP_RUN( pDevice );

QCAP_STOP( pDevice );

QCAP_DESTROY( pDevice );
```



### 3.4.4 QCAP\_GET\_AUDIO\_INPUT

#### Introduction

The user can use this function to get current audio input source.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pInput	OUT	Pointer of input audio source

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get the audio input source*

```
ULONG nInput;

QCAP_GET_AUDIO_INPUT( pDevice, &nInput );
```

# 3.5 Format Property Functions

## Introduction

This section contains device property functions to control audio/video capture data formats.

### 3.5.1 QCAP\_SET\_VIDEO\_STANDARD

#### Introduction

The user can use this function to set current video standard. If *Standard* parameter set to QCAP\_STANDARD\_TYPE\_AUTO, it will start auto video standard detection. This function only supports for SD capture card.



It is only for PAL / NTSC format. If your input signal is unstable, please disable auto video standard detection and set the correct standard directly.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nStandard	IN	Specify input video standard: QCAP_STANDARD_TYPE_NTSC_M QCAP_STANDARD_TYPE_NTSC_M_J QCAP_STANDARD_TYPE_NTSC_433 QCAP_STANDARD_TYPE_PAL_M QCAP_STANDARD_TYPE_PAL_60 QCAP_STANDARD_TYPE_PAL_B QCAP_STANDARD_TYPE_PAL_D QCAP_STANDARD_TYPE_PAL_G QCAP_STANDARD_TYPE_PAL_H QCAP_STANDARD_TYPE_PAL_I QCAP_STANDARD_TYPE_PAL_N QCAP_STANDARD_TYPE_PAL_N_COMBO QCAP_STANDARD_TYPE_AUTO (DEFAULT)

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set the video standard by auto detection*

```
QCAP_SET_VIDEO_STANDARD( pDevice, QCAP_STANDARD_TYPE_AUTO );
```

# 3.5.2 QCAP\_GET\_VIDEO\_STANDARD

## Introduction

This function is to get current video standard. This function only supports for SD capture card.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pStandard	OUT	Pointer to the video standard

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the current standard*

```
ULONG nStandard=0;

QCAP_GET_VIDEO_STANDARD( pDevice, &nStandard );
```



# 3.5.5 QCAP\_GET\_AUDIO\_CURRENT\_INPUT\_FORMAT

## Introduction

This function can get current input audio format information, the same information can retrieve via format changed callback.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get current audio input format*

```
ULONG nChannels=0, nBitsPerSample=0, nSampleFrequency=0;

QCAP_GET_AUDIO_CURRENT_INPUT_FORMAT( pDevice, &nChannels, &nBitsPerSample, &nSampleFrequency );
```

### 3.5.6 QCAP\_SET\_VIDEO\_DEFAULT\_OUTPUT\_FORMAT

#### Introduction

This function can change default video format output from a capture device. By default, the video output format from capture device is identical to input signal format. This function is for an advanced user who wants to fix the video output from the capture card. For example, the SC300 outputs **720x480i@60fps** at NTSC by default, can use this function to change it to output **704x480i@60fps** .



LabView developer can use this to set video output format.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify video output width
ULONG	nHeight	IN	Specify video output height
BOOL	bIsInterleaved	IN	Specify video output interleaved
double	dFrameRate	IN	Specify video output frame rate

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Change the default output format from capture device to 720x480i@60FPS*

```
QCAP_SET_VIDEO_DEFAULT_OUTPUT_FORMAT( pDevice, QCAP_COLORSPACE_TYEP_YUY2, 720, 480, 1, 60 );
```

### 3.5.7 QCAP\_GET\_VIDEO\_DEFAULT\_OUTPUT\_FORMAT

#### Introduction

This function to get current default video output format parameters from a capture device.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pColorSpaceType	OUT	Pointer to current color space type
ULONG *	pWidth	OUT	Pointer to current video output width
ULONG *	pHeight	OUT	Pointer to current video output height
BOOL *	pIsInterleaved	OUT	Pointer to current output Interleaved
double *	pFrameRate	OUT	Pointer to current output frame rate

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get default video output format from capture device*

```
ULONG nWidth=0, nHeight=0, nColorSpaceType=0;
BOOL nIsInterleaved=0;
double nFrameRate=0.0f;

QCAP_GET_VIDEO_DEFAULT_OUTPUT_FORMAT( pDevice,
                                     &nWidth, &nHeight,
                                     &nColorSpaceType,
                                     &nIsInterleaved, &nFrameRate );
```

### 3.5.8 QCAP\_SET\_AUDIO\_DEFAULT\_OUTPUT\_FORMAT

#### Introduction

This function can set current default audio output format parameters from capture device. The function is only available to all SD capture cards currently. The pSampleFrequency value must set as multiple of 8000.



LabView developer can use this to set audio output format.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nChannels	IN	Specify total audio channels
ULONG	nBitsPerSample	IN	Specify total audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

Example : Set the audio output format to 2 CH x 16 BITS x 48KHz

```
QCAP_SET_AUDIO_DEFAULT_OUTPUT_FORMAT( pDevice, 2, 16, 48000 );
```

### 3.5.9 QCAP\_GET\_AUDIO\_DEFAULT\_OUTPUT\_FORMAT

#### Introduction

The user can use this function to get current default audio output format parameters from a capture device.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pChannels	OUT	Pointer to total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

Example : Get the current audio output format

```
QCAP_GET_AUDIO_DEFAULT_OUTPUT_FORMAT( pDevice, &nChannel, &nBitsPerSample, &nSampleFrequency );
```



### 3.5.10 QCAP\_SET\_VIDEO\_VERTICAL Ancillary Data Grabbed Lines

#### Introduction

This function can set current video vertical data grabbed parameter in a number of lines when advanced user want to retrieve data from VANC area.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nLines	IN	Specify current lines, 0 is disable

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set VANC data to 8 lines*

```
QCAP_SET_VIDEO_VERTICAL Ancillary Data Grabbed Lines( pDevice, 8 );
```

### 3.5.11 QCAP\_GET\_VIDEO\_VERTICAL Ancillary Data Grabbed Lines

#### Introduction

This function can get current video vertical data grabbed lines.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pLines	OUT	Pointer to current lines

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

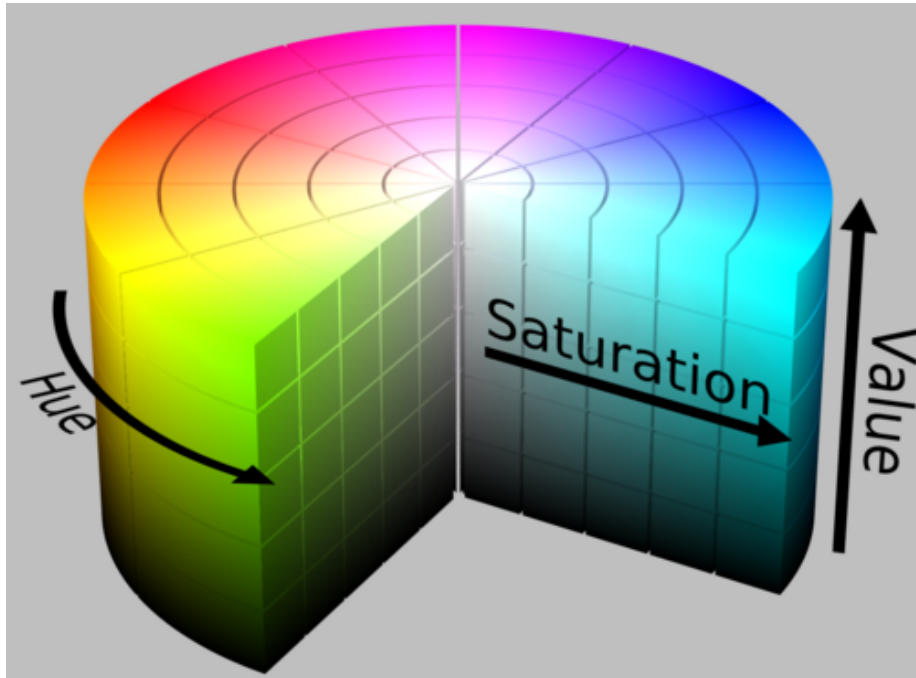
*Example : Get current video vertical data grabbed lines*

```
ULONG npLines=0;

QCAP_GET_VIDEO_VERTICAL Ancillary Data Grabbed Lines( pDevice, &npLines );
```

## 3.6 Video Property Functions

### Introduction



This section provides functions for other properties of a capture device. Including HDCP status, de-interlaced method, display region and mirror, adjust audio volume, sound renderer..etc. In the function that controls color space are following the HSV model. The relation between RGB model and HSV model is:

### 3.6.1 QCAP\_GET\_VIDEO\_CURRENT\_CONTENT\_PROTECTION\_STATUS

#### Introduction

This function can get current input source content protection status ( HDCP & MarcoVision ).

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pStatus	OUT	Pointer to current content protection status

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get content protection status*

```
ULONG nStatus=0;

QCAP_GET_VIDEO_CURRENT_CONTENT_PROTECTION_STATUS( pDevice, &nStatus );

If( Status == 1 ) Video input with HDCP encryption
    else Video input without HDCP encryption
```

### 3.6.2 QCAP\_SET\_VIDEO\_DEINTERLACE\_TYPE

#### Introduction

This function can select the de-interlace mode to the current video input.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nType	IN	Specify input video de-interlace mode: QCAP_SOFTWARE_DEINTERLACE_TYPE_BLENDING QCAP_SOFTWARE_DEINTERLACE_TYPE_MOTIONAD_APTIVE QCAP_SOFTWARE_DEINTERLACE_TYPE_FILTER_TRIANGLE QCAP_SOFTWARE_DEINTERLACE_TYPE_BOB QCAP_SOFTWARE_DEINTERLACE_TYPE_MEDICAL_RGB_REPACK

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set de-interlace mode to BLENDING*

```
QCAP_SET_VIDEO_DEINTERLACE_TYPE( pDevice, QCAP_SOFTWARE_DEINTERLACE_TYPE_BLENDING );
```

### 3.6.3 QCAP\_GET\_VIDEO\_DEINTERLACE\_TYPE

#### Introduction

This function can get current video de-interlace method.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pType	OUT	Pointer to the de-interlace method

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get current de-interlace method*

```
ULONG Type=0;

QCAP_GET_VIDEO_DEINTERLACE_TYPE( pDevice, &Type );
```

### 3.6.4 QCAP\_SET\_VIDEO\_DEINTERLACE

#### Introduction

This function is to enable/disable de-interlace function, and can only be enabled when interleaved video format, such as 1920x1080i@60fps or 720x480i@60fps is on. The default algorithm of de-interlace function is the Blending method. The user must use **QCAP\_SET\_VIDEO\_DEINTERLACE\_TYPE()** to set de-interlace type first and then can use **QCAP\_SET\_VIDEO\_DEINTERLACE()** to start de-interlaced.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
BOOL	bEnable	IN	Enable/Disable video de-interlace

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set de-Interlace mode to blending method and enable it*

```
QCAP_SET_VIDEO_DEINTERLACE_TYPE( pDevice, QCAP_SOFTWARE_DEINTERLACE_TYPE_BLENDING );

QCAP_SET_VIDEO_DEINTERLACE( pDevice, TRUE );
```

### 3.6.5 QCAP\_GET\_VIDEO\_DEINTERLACE

#### Introduction

This function can get the current status of de-interlace function.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
BOOL *	pEnable	OUT	Return the current status of this function

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get current de-interlace status*

```
BOOL Enable=0;

QCAP_GET_VIDEO_DEINTERLACE( pDevice, &bEnable );
```

### 3.6.6 QCAP\_SET\_VIDEO\_REGION\_DISPLAY

#### Introduction

This function can set the clipping area of an input video, the crop region will be scaled to capture device display output.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nCropX	IN	Specify the x-coordinate of the crop region display
ULONG	nCropY	IN	Specify the y-coordinate of the crop region display
ULONG	nCropW	IN	Specify the width of crop region display
ULONG	nCropH	IN	Specify the height of crop region display

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set crop region x,y(0,0) to w,h(1280x720) to be displayed*

```
QCAP_SET_VIDEO_REGION_DISPLAY( pDevice, 0, 0, 1280, 720 );
```

### 3.6.7 QCAP\_GET\_VIDEO\_REGION\_DISPLAY

#### Introduction

This function can get the current region clipping area parameters.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pCropX	OUT	Pointer to x-coordinate of the crop region display
ULONG *	pCropY	OUT	Pointer to y-coordinate of the crop region display
ULONG *	pCropW	OUT	Pointer to horizontal width of crop region display
ULONG *	pCropH	OUT	Pointer to vertical height of crop region display

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get the clipping area setting*

```
ULONG bCropX,bCropY,bCropW,bCropH;

QCAP_GET_VIDEO_REGION_DISPLAY( pDevice, &bCropX, &bCropY, &bCropW, &bCropH );
```

## 3.6.8 QCAP\_SET\_VIDEO\_MIRROR

### Introduction

This function can set the video flipping mode in both vertical and horizontal way.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
BOOL	bHorizontalMirror	IN	Enable/Disable horizontal of mirror
BOOL	bVerticalMirror	IN	Enable/Disable vertical of mirror

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Flipping video up-down & left-right*

```
QCAP_SET_VIDEO_MIRROR( pDevice, TRUE, TRUE );
```

## 3.6.9 QCAP\_GET\_VIDEO\_MIRROR

### Introduction

This function can get the flipping mode of a video display.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
BOOL *	pHorizontalMirror	OUT	Return the current horizontal of mirror
BOOL *	pVerticalMirror	OUT	Return the current vertical of mirror

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get the video flipping mode*

```
BOOL bHorizontalMirror, bVerticalMirror;

QCAP_GET_VIDEO_MIRROR( pDevice, &bHorizontalMirror, &bVerticalMirror );
```

### 3.6.10 QCAP\_SET\_VIDEO\_BRIGHTNESS

### 3.6.11 QCAP\_SET\_VIDEO\_BRIGHTNESS\_EX

#### Introduction

This function can set video brightness value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.



**physical value** means video input side on channel source, **display value** means video render side on graphics card.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nPhysicalValue	IN	Specify the input video brightness 0-255
ULONG	nDisplayValue	IN	Specify to output video brightness 0-255 <b>Only in QCAP_SET_VIDEO_BRIGHTNESS_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 3.6.12 QCAP\_GET\_VIDEO\_BRIGHTNESS

### 3.6.13 QCAP\_GET\_VIDEO\_BRIGHTNESS\_EX

#### Introduction

This function can get video brightness value range from 0 to 255. The neutral value is 128.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pPhysicalValue	OUT	pointer to input video brightness 0-255
ULONG *	pDisplayValue	OUT	pointer to output video brightness 0-255 <b>Only in QCAP_GET_VIDEO_BRIGHTNESS_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set/get the video brightness value*

```
QCAP_GET_VIDEO_BRIGHTNESS( pDevice, &pPhysicalValue );
QCAP_SET_VIDEO_BRIGHTNESS( pDevice, pPhysicalValue + 20 ); //DisplayValue plus by 20

QCAP_GET_VIDEO_BRIGHTNESS_EX( pDevice, &pPhysicalValue, &pDisplayValue );
QCAP_SET_VIDEO_BRIGHTNESS_EX( pDevice, pPhysicalValue, pDisplayValue + 30 ); //DisplayValue plus by 30
```



### 3.6.14 QCAP\_SET\_VIDEO\_CONTRAST

### 3.6.15 QCAP\_SET\_VIDEO\_CONTRAST\_EX

#### Introduction

This function can set video contrast value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting the new value.



**physical value** means video input side on channel source, **display value** means video render side on graphics card.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nPhysicalValue	IN	Specify the input video contrast 0-255
ULONG	nDisplayValue	IN	Specify the output video contrast 0-255 <b>Only in QCAP_SET_VIDEO_CONTRAST_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 3.6.16 QCAP\_GET\_VIDEO\_CONTRAST

### 3.6.17 QCAP\_GET\_VIDEO\_CONTRAST\_EX

#### Introduction

This function can get video contrast value range from 0 to 255. The neutral value is 128.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pPhysicalValue	OUT	pointer to input video contrast 0-255
ULONG *	pDisplayValue	OUT	pointer to output video contrast 0-255 <b>Only in QCAP_GET_VIDEO_CONTRAST_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set/get the video contrast value*

```
QCAP_GET_VIDEO_CONTRAST( pDevice, &pPhysicalValue );
QCAP_SET_VIDEO_CONTRAST( pDevice, pPhysicalValue + 20 ); //DisplayValue plus by 20

QCAP_GET_VIDEO_CONTRAST_EX( pDevice, &pPhysicalValue, &pDisplayValue );
QCAP_SET_VIDEO_CONTRAST_EX( pDevice, pPhysicalValue, pDisplayValue + 30 ); //DisplayValue plus by 30
```

### 3.6.18 QCAP\_SET\_VIDEO\_HUE

### 3.6.19 QCAP\_SET\_VIDEO\_HUE\_EX

#### Introduction

This function can set video hue value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.



**physical value** means video input side on channel source, **display value** means video render side on graphics card.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nPhysicalValue	IN	Specify the input video hue 0-255
ULONG	nDisplayValue	IN	Specify the output video hue 0-255 <b>Only in QCAP_SET_VIDEO_HUE_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 3.6.20 QCAP\_GET\_VIDEO\_HUE

### 3.6.21 QCAP\_GET\_VIDEO\_HUE\_EX

#### Introduction

This function can get video hue value range from 0 to 255. The neutral value is 128.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pPhysicalValue	OUT	pointer to input video hue 0-255
ULONG *	pDisplayValue	OUT	pointer to output video hue 0-255 <b>Only in QCAP_GET_VIDEO_HUE_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set/get the video hue value*

```
QCAP_GET_VIDEO_HUE( pDevice, &pPhysicalValue );
QCAP_SET_VIDEO_HUE( pDevice, pPhysicalValue + 20 ); //DisplayValue plus by 20

QCAP_GET_VIDEO_HUE_EX( pDevice, &pPhysicalValue, &pDisplayValue );
QCAP_SET_VIDEO_HUE_EX( pDevice, pPhysicalValue, pDisplayValue + 30 ); //DisplayValue plus by 30
```

### 3.6.22 QCAP\_SET\_VIDEO\_SATURATION

### 3.6.23 QCAP\_SET\_VIDEO\_SATURATION\_EX

#### Introduction

This function can set video saturation value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.



**physical value** means video input side on channel source, **display value** means video render side on graphics card.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nPhysicalValue	IN	Specify the input video saturation 0-255
ULONG	nDisplayValue	IN	Specify the output video saturation 0-255 <b>Only in QCAP_SET_VIDEO_SATURATION_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 3.6.24 QCAP\_GET\_VIDEO\_SATURATION

### 3.6.25 QCAP\_GET\_VIDEO\_SATURATION\_EX

#### Introduction

This function can get video saturation value range from 0 to 255. The neutral value is 128.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pPhysicalValue	OUT	pointer to input video saturation 0-255
ULONG *	pDisplayValue	OUT	pointer to output video saturation 0-255 <b>Only in QCAP_GET_VIDEO_SATURATION_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set/get the video saturation*

```
QCAP_GET_VIDEO_SATURATION( pDevice, &pPhysicalValue );
QCAP_SET_VIDEO_SATURATION( pDevice, pPhysicalValue + 20 ); //DisplayValue plus by 20

QCAP_GET_VIDEO_SATURATION_EX( pDevice, &pPhysicalValue, &pDisplayValue );
QCAP_SET_VIDEO_SATURATION_EX( pDevice, pPhysicalValue, pDisplayValue + 30 ); //DisplayValue plus by 30
```

### 3.6.26 QCAP\_SET\_VIDEO\_SHARPNESS

### 3.6.27 QCAP\_SET\_VIDEO\_SHARPNESS\_EX

#### Introduction

This function can set video sharpness value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.



**physical value** means video input side on channel source, **display value** means video render side on graphics card.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nPhysicalValue	IN	Specify the input video sharpness 0-255
ULONG	nDisplayValue	IN	Specify the output video sharpness 0-255 <b>Only in QCAP_SET_VIDEO_SHARPNESS_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 3.6.28 QCAP\_GET\_VIDEO\_SHARPNESS

### 3.6.29 QCAP\_GET\_VIDEO\_SHARPNESS\_EX

#### Introduction

This function can get video sharpness value range from 0 to 255. The neutral value is 128.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pPhysicalValue	OUT	pointer to input video sharpness 0-255
ULONG *	pDisplayValue	OUT	pointer to output video sharpness 0-255 <b>Only in QCAP_GET_VIDEO_SHARPNESS_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set/get the video sharpness*

```
QCAP_GET_VIDEO_SHARPNESS( pDevice, &pPhysicalValue );
QCAP_SET_VIDEO_SHARPNESS( pDevice, pPhysicalValue + 20 ); //DisplayValue plus by 20

QCAP_GET_VIDEO_SHARPNESS_EX( pDevice, &pPhysicalValue, &pDisplayValue );
QCAP_SET_VIDEO_SHARPNESS_EX( pDevice, pPhysicalValue, pDisplayValue + 30 ); //DisplayValue plus by 30
```

# 3.7 Audio Property Functions

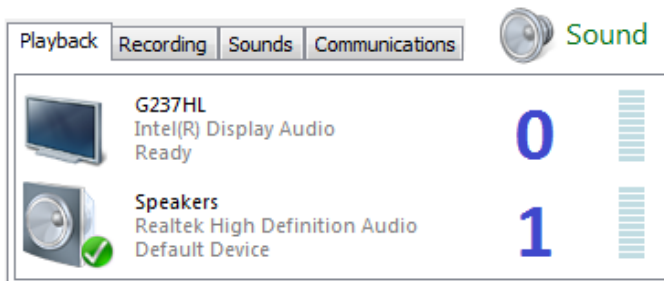
## Introduction

This is section contains the audio property to set in the device, such as sound renderer can decide which output device to play the sound, and it's volume setting.

## 3.7.1 QCAP\_SET\_AUDIO\_SOUND\_RENDERER

### Introduction

This function can set current sound renderer from an available sound output device. For example a system sound renderer list in a figure, each sound output device has its index number (count from the first one) as *sound number*. The number will remain the same after restart.



### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iSoundNum	IN	Sound Renderer, default Renderer is 0

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the current audio recording input device to Sound Device #1*

```
UINT nSounder = 1;

QCAP_SET_AUDIO_SOUND_RENDERER( pDevice, nSounder );
```

## 3.7.2 QCAP\_GET\_AUDIO\_SOUND\_RENDERER

### Introduction

This function can get current sound renderer of sound output device used.

Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT *	pSoundNum	OUT	Pointer to Sound number

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

*Example : Get current sound renderer*

```
UINT nSounder = 0;

QCAP_GET_AUDIO_SOUND_RENDERER( pDevice, &nSounder );
```

### 3.7.3 QCAP\_SET\_AUDIO\_VOLUME

#### Introduction

This function can set current audio volume value, range from 0 to 100.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nVolume	IN	Audio volume range 0-100. Set 0 to mute

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set current audio volume to half*

```
QCAP_SET_AUDIO_VOLUME( pDevice, 50 );
```

### 3.7.4 QCAP\_GET\_AUDIO\_VOLUME

#### Introduction

This function can get current audio volume value, range from 0 to 100.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pVolume	OUT	Pointer to audio volume value

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get current audio volume*

```
ULONG nVolume;

QCAP_GET_AUDIO_VOLUME( pDevice, &nVolume );
```

# 3.8 Advanced Property Functions

## Introduction

This section contains hardware properties setting for advanced users. The user can set custom parameters to hardware properties to change the recording result for encoding format, downscale mode, recording profile, bit rate, quality...etc. The detailed custom properties depend on the hardware capture card.

For more custom property programming, please reference Windows Media SDK and the product's Extra Programming Guide in SDK package.

## 3.8.1 QCAP\_SET\_VIDEO\_PREVIEW\_PROPERTY\_EX

### Introduction

The user can use this function to set video preview format size of SC590 capture card.



This function is only supported on SC590 & SC3B0 capture card.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nDownscaleMode	IN	Specify video preview downscale mode: QCAP_DOWNSCALE_MODE_OFF QCAP_DOWNSCALE_MODE_2_3 QCAP_DOWNSCALE_MODE_1_2 QCAP_DOWNSCALE_MODE_1_4

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the video preview downscale mode off (for example: use SC590 card )*

```
QCAP_SET_VIDEO_PREVIEW_PROPERTY_EX( pDevice, QCAP_DOWNSCALE_MODE_OFF );
```



# 3.8.2 QCAP\_GET\_VIDEO\_PREVIEW\_PROPERTY\_EX

## Introduction

The user can use this function to get video preview format size of SC590 capture card.



This function is only supported on SC590 & SC3B0 capture card.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG *	pDownscaleMode	OUT	Pointer to the downscale mode

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video preview downscale mode off (for SC590 card )*

```
ULONG DownscaleMode;

QCAP_GET_VIDEO_PREVIEW_PROPERTY_EX( pDevice, &DownscaleMode );
```

## QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_VIDEOCOMPRESSION\_PROPERTY

This function can get some custom device properties when a hardware encode capture card is used.

For more custom property programming, please reference Windows Media SDK and product's Extra Programming Guide in SDK package.#



type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters
ULONG	nProperty	IN	Specify the property to get from the device
ULONG	nValue	IN	Specify the property value. The range of value is dependent on its property

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

*Example : Set custom OSD color to hardware*

[illegible]

## QCAP\_GET\_VIDEO\_HARDWARE\_ENCODER\_VIDEOCOMPRESSION\_PROPERTY

This function can is used to get some custom device properties.

For more custom property programming, please reference Windows Media SDK and product's Extra Programming Guide in SDK package.



type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters
ULONG	nProperty	IN	Specify the property that will be gotten from the device.
ULONG *	pValue	OUT	Pointer the specify property value. The range of value is dependent on its property. It cannot be NULL.

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

*Example : To obtain device GPIO data from hardware*

[illegible]

### 3.8.5 QCAP\_SET\_DEVICE\_CUSTOM\_PROPERTY

### 3.8.6 QCAP\_SET\_DEVICE\_CUSTOM\_PROPERTY\_EX

#### Introduction

This function is used to set/change some custom device properties. All properties are designed to provide particular effects through hardware or software support.



For more custom property programming, please reference Windows Media SDK and product's Extra Programming Guide in SDK package.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nProperty	IN	Specify the property get from the device.
BYTE *	pValue	IN	Pointer the specify property value. The range of value is dependent on its property. It cannot be NULL.
ULONG	nBytes	IN	Specify the size of property value buffer. <b>Only in QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example 1: Set SC580 customer property for OSD text. (SC5A0 not supported)*

```
QCAP_SET_DEVICE_CUSTOM_PROPERTY( pDevice, 929, 0x00000001 );

QCAP_SET_DEVICE_CUSTOM_PROPERTY( pDevice, 920, 0x00000000 );

CHAR path_text[] = "C:/WINDOWS/FH8735/OSD.TXT";

QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice, 921, (BYTE *) (path_text), strlen(path_text) );
```

*Example 2: Set SC580 OSD picture in hardware encoder. (SC5A0 not supported)*

```
//Note SC580 allows software to load one 8 bits (256 colors) BMP file into its board memory.
ULONG params[ 4 ] = { 0, /*Picture index is 0 or 1 */
                     1, /*Picture start left position*/
                     1, /*Picture start top position*/
                     255 }; /*Picture transparent is from 0-255*/

CHAR path_picture[] = "C:/WINDOWS/FH8735/Slimmer_64.bmp";

QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice, 970, (BYTE *) params, sizeof(params) );

QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice, 971, (BYTE *) (path_picture), strlen(path_picture) );
```

### 3.8.7 QCAP\_GET\_DEVICE\_CUSTOM\_PROPERTY

### 3.8.8 QCAP\_GET\_DEVICE\_CUSTOM\_PROPERTY\_EX

#### Introduction

This function can get custom device properties from a capture device.

All properties are designed to provide particular effects through hardware or software support.



For more custom property programming, please reference Windows Media SDK and product's Extra Programming Guide in SDK package.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nProperty	IN	Specify the property to get from the device
BYTE *	pValue	OUT	Pointer to variable that stores the specify property value. The range of return value is dependent on its property. It cannot be NULL.
ULONG	nBytes	IN	Specify the size of property value in bytes. <b>Only in QCAP_GET_DEVICE_CUSTOM_PROPERTY_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get device custom property: 1.device serial number 2.analog video switch table*

```
//To obtain device GPIO data
QCAP_GET_DEVICE_CUSTOM_PROPERTY( pDevice,
                                KSPROPERTY_CUSTOM_XET_GPIO_DATA,
                                &nDeviceGPIOdata );

//To obtain analog video switch table
QCAP_GET_DEVICE_CUSTOM_PROPERTY_EX( pDevice,
                                    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_SWITCH_TABLE,
                                    &nDeviceVideoSwitchTable
                                    12 );
```

# 3.9 Device Helper Functions

## Introduction

Device helper functions are used to make your programs easier to query name, hardware capabilities, and device information. For NI® LabView users or language has no callback function, QCAP provides polling APIs can help to complete the data acquisition.

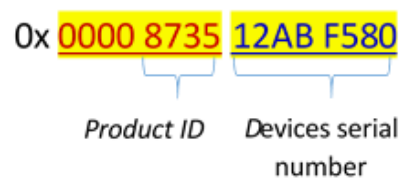
## 3.9.1 QCAP\_DEVICE\_ENUMERATION

### Introduction

This function retrieval device information such as device serial number, device name, and hardware capabilities installed on the system. To enumerate devices, you can query by *Device Name* or *Serial Number*. This function walks through every card attached on the system, check audio/video with capture/encoder capabilities, and return all devices data in a list. The enumerated data are useful to create the capture object by calling **QCAP\_CREATE()** for each device.

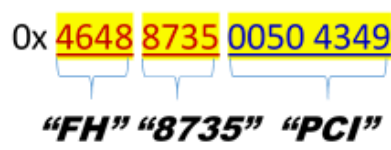
(a) To enumerate by **QCAP\_ENUM\_TYPE\_SERIAL\_NUMBER** to get a unique serial number:

For example, SC580 returns 8 bytes data, we get the product\_id is "8735", and devices serial number is "12ABF580".



(b) To enumerate by **QCAP\_ENUM\_TYPE\_DEVICE\_NAME** to get a device string:

For example, SC580 returns 8 bytes data, we get the device name string "FH8735 PCI".



### Parameters

type	parameter	I/O	descriptions
ULONGLONG **	ppVideoDeviceList	OUT	the pointer to array of video capture information
ULONG *	pVideoDeviceSize	OUT	The number of video capture devices exist
ULONGLONG **	ppVideoEncoderDeviceList	OUT	the pointer to array of video encoder information
ULONG *	pVideoEncoderDeviceSize	OUT	The number of hardware video encoder exist
ULONGLONG **	ppAudioDeviceList	OUT	the pointer to array of audio device information
ULONG *	pAudioDeviceSize	OUT	The number of audio device exist
ULONGLONG **	ppAudioEncoderDeviceList	OUT	the pointer to array of audio encoder information
ULONG *	pAudioEncoderDeviceSize	OUT	The number of hardware audio encoder exist
ULONG	nDeviceEnumType	IN	<b>default QCAP_ENUM_TYPE_DEVICE_NAME</b> Specific the way to enumerate hardware devices: 1. <b>QCAP_ENUM_TYPE_SERIAL_NUMBER</b> return hardware devices by unique serial number

			<b>2. QCAP_ENUM_TYPE_DEVICE_NAME</b> return hardware devices by unique device string
--	--	--	---

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Enumerate all available devices attached to the system*

```
ULONGLONG *pVideoDeviceList = NULL;
ULONGLONG *pVideoEncoderDeviceList = NULL;
ULONGLONG *pAudioDeviceList = NULL;
ULONGLONG *pAudioEncoderDeviceList = NULL;

ULONG nVideoDeviceList_size = 0;
ULONG nVideoEncoderDeviceList_size = 0;
ULONG nAudioDeviceList_size = 0;
ULONG nAudioEncoderDeviceList_size = 0;

HRESULT qr = QCAP_DEVICE_ENUMERATION( &pVideoDeviceList,
                                       &nVideoDeviceList_size,
                                       &pVideoEncoderDeviceList,
                                       &nVideoEncoderDeviceList_size,
                                       &pAudioDeviceList,
                                       &nAudioDeviceList_size,
                                       &pAudioEncoderDeviceList,
                                       &nAudioEncoderDeviceList_size,
                                       QCAP_ENUM_TYPE_SERIAL_NUMBER );

if( qr == QCAP_RS_SUCCESSFUL )
{
    for( ULONG i = 0 ; i < nVideoDeviceList_size ; i++ )

        printf( "Live#%02d = 0x%016l1X\n", i, pVideoDeviceList[ i ] );

    for( ULONG i = 0 ; i < nVideoEncoderDeviceList_size ; i++ )

        printf( "Encoder#%02d = 0x%016l1X\n", i, pVideoEncoderDeviceList[ i ] );
}
```

### 3.9.1.1 QCAP\_GET\_DEVICE\_ENUMERATION\_ITEM\_INFO

For C# developer user this function to enumerate device by index.

type	parameter	I/O	descriptions
UINT	iDevNum	IN	Device index specified according to pszDevName, start from 0
ULONGLONG *	pDeviceList	IN	Specify a device list
ULONG *	pDeviceInfoH	OUT	Pointer to device information high-word
ULONG *	pDeviceInfoL	OUT	Pointer to device information low-word

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Enumerate all available devices attached to the system*

```
uint pVideoDeviceList = 0;
uint pVideoEncoderDeviceList = 0;
uint pAudioDeviceList = 0;
uint pAudioEncoderDeviceList = 0;

uint nVideoDeviceSize = 0;
uint nVideoEncoderDeviceSize = 0;
uint nAudioDeviceSize = 0;
uint nAudioEncoderDeviceSize = 0;

uint pDeviceInfoH = 0;
uint pDeviceInfoL = 0;

// Enumerate all capture devices on the platform
EXPORTS.QCAP_DEVICE_ENUMERATION( ref pVideoDeviceList,
                                ref nVideoDeviceSize,
                                ref pVideoEncoderDeviceList,
                                ref nVideoEncoderDeviceSize,
                                ref pAudioDeviceList,
                                ref nAudioDeviceSize,
                                ref pAudioEncoderDeviceList,
                                ref nAudioEncoderDeviceSize );

// Get the device item info of the index 0
EXPORTS.QCAP_GET_DEVICE_ENUMERATION_ITEM_INFO( 0,
                                                pVideoDeviceList,
                                                ref pDeviceInfoH,
                                                ref pDeviceInfoL );
```

C#



# 3.9.2 QCAP\_QUERY\_DEVICE\_CAP

## Introduction

This function can query the number of main/sub hardware encoders of a device handle. This is called after a device handle is created.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Pointer to Device handle
DWORD *	pDeviceSerialNumber	OUT	Pointer to device serial number e.g. 0x12ABF580
BOOL *	pHasHardwareMainEncoder	OUT	Indicate hardware main encoder available
BOOL *	pHasHardwareSubEncoder	OUT	Indicate hardware sub encoder available

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : query the number of hardware encoders*

```
DWORD DeviceSerialNumber = 0;
BOOL HasHardwareMainEncoder = 0;
BOOL HasHardwareSubEncoder = 0;

QCAP_QUERY_DEVICE_CAP( hVideoDevice,
                        &DeviceSerialNumber,
                        &HasHardwareMainEncoder,
                        &HasHardwareSubEncoder );

//example result: "serial number=0x12ABF580, main_encoder=1 sub_encoder=1"

printf( "Serial number=%X, main_encoder=%d sub_encoder=%d\n",
        DeviceSerialNumber,
        HasHardwareMainEncoder,
        HasHardwareSubEncoder );
```

# 4 Snapshot Function API

---

## Introduction



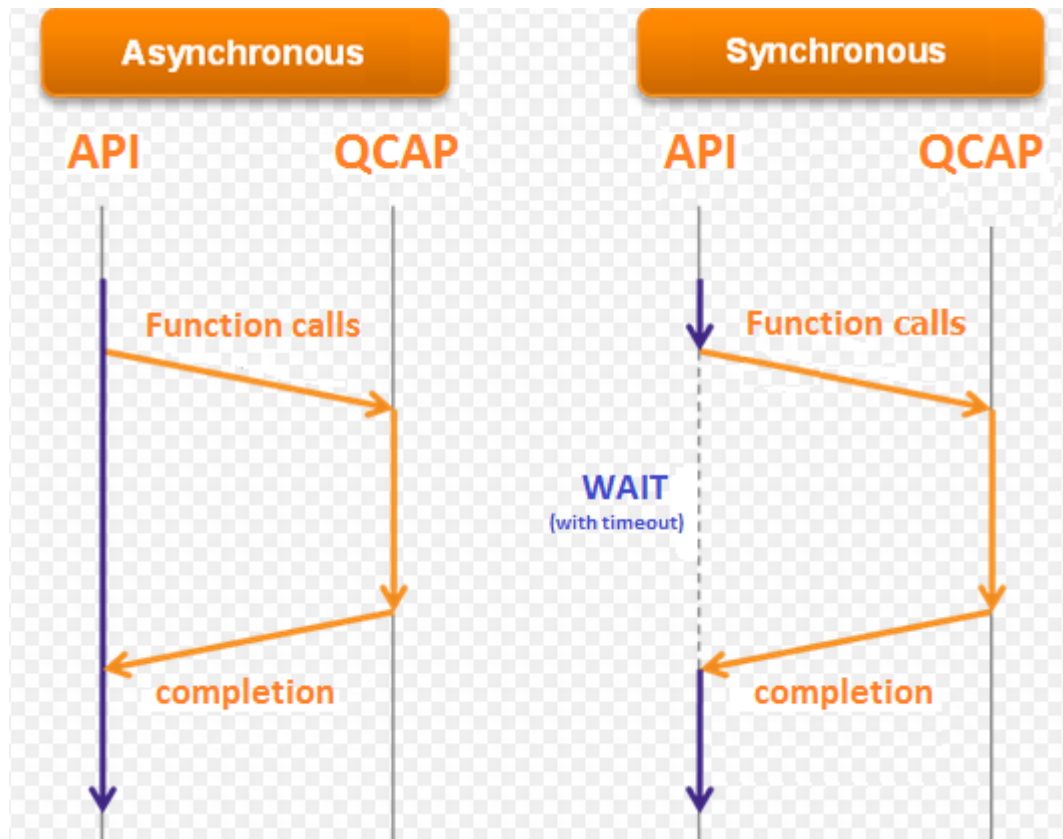
This chapter will help to take a snapshot of your current capture video. Follow this guide to take a snapshot of your whole video display, or any section of the video you want. The user can save a snapshot to a **BMP/JPG**, or to trigger a snapshot to get the image stream buffer from a callback function without saving it to disk.

## 4.1 QCAP\_SNAPSHOT\_BMP

## 4.2 QCAP\_SNAPSHOT\_BMP\_EX

### Introduction

This function takes a snapshot of video and saves to **BMP** 24bit or 32bit file.



This function is designed for both synchronous and asynchronous operations. While snapshot is taking, user can decide to wait until completion (sync mode) or run snapshot in background (async mode):

1. In asynchronous mode( *blsAsync* = TRUE ):
  - function returns immediately.
  - QCAP will use a counter for snapshot image. For example, if you call 100 times, you will generate 100 different and continuous images in sequence,
2. In Synchronous mode ( *blsAsync* = FALSE ):
  - if *nMillisecond* is INFINITE, the function will be blocked until the snapshot is completed.
  - If *nMillisecond* is 0, the function returns immediately and generates image file when next frame is coming.



For .NET developer, the INFINITE is defined as 0xFFFFFFFF.

The user can use this function to set cropping parameters of snapshot a picture to destination **BMP** image format. If the destination rectangle is not same as the cropping rectangle, the snapshot image will be scaled to destination rectangle.

The parameter *pszFilePathName* does specify the filename of the snapshot you want to save to disk. For a user who wants to get the snapshot in buffer without saving to a file, it could be done by passing *pszFilePathName* with file extension only. For example, use **"BMP"** instead of "Filename.BMP" as the parameter, then snapshot stream data can be retrieved in **PF\_SNAPSHOT\_STREAM\_CALLBACK** callback function.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.BMP24" → save to 24bit <b>BMP</b> "Filename.BMP32 or BMP" → save to 32bit <b>BMP</b> "BMP24" → To snapshot stream in callback only (no save to file.) "BMP32"/"BMP" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SNAPSHOT_BMP_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SNAPSHOT_BMP_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SNAPSHOT_BMP_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SNAPSHOT_BMP_EX()</b>
ULONG	nDstW	IN	The width of the snapshot rectangle to scale <b>Only in QCAP_SNAPSHOT_BMP_EX()</b>
ULONG	nDstH	IN	The height of the snapshot rectangle to scale <b>Only in QCAP_SNAPSHOT_BMP_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> For users who use asynchronous mode,
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

### Example 1: Take a snapshot and save to file

```
QCAP_SNAPSHOT_BMP( pDevice, "C:/PICTURE1.BMP" );

QCAP_SNAPSHOT_BMP( pDevice, "C:/PICTURE2.BMP24" );

QCAP_SNAPSHOT_BMP_EX( pDevice, "C:/PICTURE3.BMP", 10, 40, 1900, 1000, 720, 480 );

QCAP_SNAPSHOT_BMP_EX( pDevice, "C:/PICTURE4.BMP23", 10, 40, 1900, 1000, 720, 480 );
```

### Example 2: Take a snapshot without saving to file

```
QRETURN snapshot_stream_callback( PVOID pDevice,
                                   CHAR *pszFilePathName,
                                   BYTE *pStreamBuffer,
                                   LONG nStreamBufferLen,
                                   PVOID pUserData )
{
    if( pStreamBuffer==NULL || nStreamBufferLen==0 ) printf("NO DATA");

    //Get snapshot stream data here

    return QCAP_RT_OK;
}

QCAP_SNAPSHOT_BMP( pDevice, "BMP" );

QCAP_SNAPSHOT_BMP_EX( pDevice, "BMP24", 10, 40, 1900, 1000, 720, 480 );
```

## 4.3 QCAP\_SNAPSHOT\_JPG

## 4.4 QCAP\_SNAPSHOT\_JPG\_EX

### Introduction

This function takes a snapshot of video and saves to **JPEG** image file format. The **JPEG** is a lossy image compression file format, so it has a *nQuality* parameter can control the quality of **JPEG** file(range from 0-100). The lower quality values the smaller image file size. a typically value 80 can get the better picture result and good file size.

This function is designed for both synchronous and asynchronous operations, for detailed *blsAsync* parameter please refers to **QCAP\_SNAPSHOT\_BMP()**.

The user can use this function to set cropping parameters of snapshot a picture to destination **BMP** image format. If the destination rectangle is not same as the cropping rectangle, the snapshot image will be scaled to destination rectangle.

The parameter *pszFilePathName* does specify the filename of the snapshot you want to save to disk. For a user who wants to get the snapshot in buffer without saving to a file, it could be done by passing *pszFilePathName* with file extension only. For example, use "JPG" instead of "Filename.JPG" as the parameter, then snapshot stream data can be retrieved in **PF\_SNAPSHOT\_STREAM\_CALLBACK** callback function.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.JPG" → To snapshot to <b>JPEG</b> image file "JPG" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SNAPSHOT_JPG_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SNAPSHOT_JPG_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SNAPSHOT_JPG_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SNAPSHOT_JPG_EX()</b>
ULONG	nDstW	IN	The width of the snapshot rectangle to scale <b>Only in QCAP_SNAPSHOT_JPG_EX()</b>
ULONG	nDstH	IN	The height of the snapshot rectangle to scale <b>Only in QCAP_SNAPSHOT_JPG_EX()</b>
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example 1: Take a snapshot, and cropping from x=20,y=80 with rectangle 1900x1000 and scale to 720x480 in **JPEG***

```
QCAP_SNAPSHOT_JPG( pDevice, "C:/PICTURE1.JPG", 80 );

QCAP_SNAPSHOT_JPG_EX( pDevice, "C:/PICTURE2.JPG", 20, 80, 1900, 1000, 720, 480, 100 );
```

*Example 2: Take a snapshot without saving to **JPEG** file*

```
QRETURN snapshot_stream_callback( PVOID pDevice,
                                   CHAR *pszFilePathName,
                                   BYTE *pStreamBuffer,
                                   LONG nStreamBufferLen,
                                   PVOID pUserData )
{
    if( pStreamBuffer==NULL || nStreamBufferLen==0 ) printf("NO DATA");

    //Get snapshot stream data here

    return QCAP_RT_OK;
}

QCAP_SNAPSHOT_JPG( pDevice, "JPG", 80 );

QCAP_SNAPSHOT_JPG_EX( pDevice, "JPG", 10, 40, 1900, 1000, 720, 480, 80 );
```

## 4.5 QCAP\_SNAPSHOT\_BUFFER\_TO\_BMP\_EX

## 4.6 QCAP\_SNAPSHOT\_BUFFER\_TO\_JPG\_EX

### Introduction

This function saves a video stream buffer to **BMP/JPEG** image file format.

For more detailed parameters information, please refers to **QCAP\_SNAPSHOT\_BMP()/QCAP\_SNAPSHOT\_JPG()**.

### Parameters

type	parameter	I/O	descriptions
CHAR *	pszFilePathName	IN	Specify the filename to store image
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pSrcFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nSrcWidth	IN	Specify the width of frame contained in a buffer
ULONG	nSrcHeight	IN	Specify the height of frame contained in a buffer
ULONG	nSrcPitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate by width and color space format.
UINT	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle
UINT	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle
UINT	nCropW	IN	The width of the crop rectangle
UINT	nCropH	IN	The height of the crop rectangle
UINT	nDstW	IN	The width of the snapshot rectangle to scale
UINT	nDstH	IN	The height of the snapshot rectangle to scale
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100 <b>Only in QCAP_SNAPSHOT_BUFFER_TO_JPG_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.



## Examples

*Example: Take a snapshot in video preview callback*

```
QRETURN video_preview_callback( PVOID pDevice, double dSampleTime, BYTE * pFrameBuffer, ULONG
nFrameBufferLen, PVOID pUserData )
{
    //Get snapshot stream data here

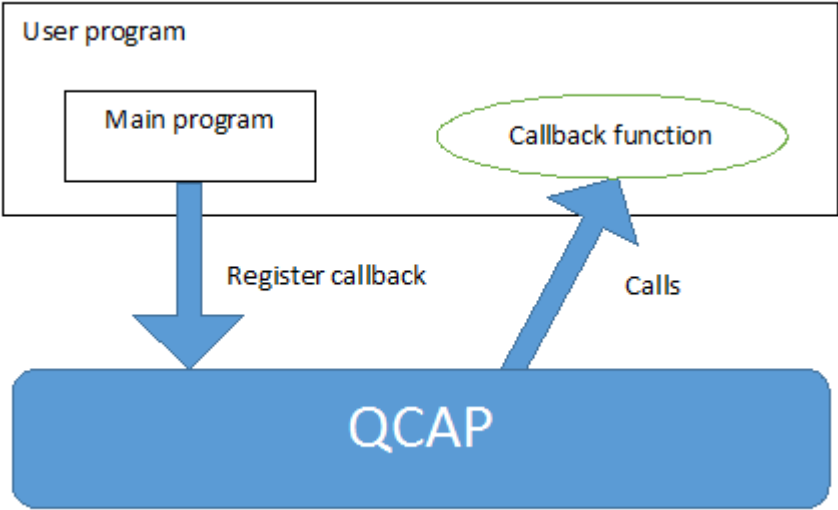
    QCAP_SNAPSHOT_BUFFER_TO_BMP_EX( "C:/PICTURE1.BMP", QCAP_COLORSPACE_TYEP_YV12, pFrameBuffer,
Width, Height, 0, 0, 0, Width, Height, Width, Height );

    QCAP_SNAPSHOT_BUFFER_TO_JPG_EX( "C:/PICTURE1.JPG", QCAP_COLORSPACE_TYEP_YV12, pFrameBuffer,
Width, Height, 0, 0, 0, Width, Height, Width, Height, 80 );

    return QCAP_RT_OK;
}
```

# 4.7 Snapshot Callback Functions

## Introduction



The snapshot callback function is a pointer to a user-defined the function and will be called by the QCAP library. There are 2 callback functions (and its register functions) for different purposes. For each callback function to work, you need to register it with a user defined function before **QCAP\_SNAPSHOT\_BMP/JPG()**. For example, user can get notification when snapshot completed or retrieve snapshot image stream in buffer directly.

this section contains how to register callback functions and its functionalities for:

Register functions	Callback functions
QCAP_REGISTER_SNAPSHOT_DONE_CALLBACK	<i>PF_SNAPSHOT_DONE_CALLBACK</i>
QCAP_REGISTER_SNAPSHOT_STREAM_CALLBACK	<i>PF_SNAPSHOT_STREAM_CALLBACK</i>



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

# 4.7.1 QCAP\_REGISTER\_SNAPSHOT\_DONE\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_BMP/JPG()** is completed. The user can get the path filename of the snapshot in the callback.

When uses asynchronous snapshot (*bIsAsync* = TRUE), it is useful to know when snapshot file is ready.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
<b>PF_SNAPSHOT_DONE_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	user defined data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_SNAPSHOT\_DONE\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
CHAR *	pszFilePathName	Pointer to path filename of image
PVOID	pUserData	user defined data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example: Register a callback function for snapshot done*

```
// SNAPSHOT DONE CALLBACK
QRETURN on_snapshot_done_callback( PVOID pDevice,
                                   CHAR *pszFilePathName,
                                   PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SNAPSHOT_STREAM_CALLBACK pCB = on_snapshot_done_callback;

    QCAP_REGISTER_SNAPSHOT_DONE_CALLBACK( pDevice, pCB, pUserData );
}
```

# 4.7.2 QCAP\_REGISTER\_SNAPSHOT\_STREAM\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_BMP/JPG()** image stream is generated, then user can get image stream in buffer directly.

For the users who want a snapshot without saving to a file, call **QCAP\_SNAPSHOT\_BMP/JPG()** by passing *pszFilePathName* file extension only (e.g. "BMP"), then a user can use this callback to retrieve the snapshot image in the buffer.



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
<b><i>PF_SNAPSHOT_STREAM_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	user defined data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# ***PF\_SNAPSHOT\_STREAM\_CALLBACK***

## *Parameters of Callback*

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
CHAR *	pszFilePathName	Pointer to path filename of image
BYTE *	pStreamBuffer	pointer to image framebuffer
ULONG	nStreamBufferLen	Indicates the image framebuffer length
PVOID	pUserData	user defined data

## *Return value of Callback*

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example: Register a callback function for snapshot stream completion*

```
// SNAPSHOT_STREAM_CALLBACK
QRETURN on_snap_stream_callback( PVOID pDevice,
                                CHAR *pszFilePathName,
                                BYTE *pStreamBuffer,
                                LONG nStreamBufferLen,
                                PVOID pUserData )
{
    if( pStreamBuffer==NULL || nStreamBufferLen==0 ) printf("NO DATA");

    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SNAPSHOT_STREAM_CALLBACK pCB = on_snap_stream_callback;

    QCAP_REGISTER_SNAPSHOT_STREAM_CALLBACK( pDevice, pCB, pUserData );
}
```

# 5 Recording Function API

## Introduction



The channel recording API is to provide a simple mechanism for a user to record captured streams from the input devices. The recording properties for audio/video can be manually performed by set record property functions. For example, the user can select the encoder type, video recording bit rate and quality, video file format, and adjust the audio/video synchronize time... etc. There are also recording callbacks when the recording completion or when audio/video data available.

More complex applications can use by setting the record properties for encoding engine on the specified hardware/platform, to gain the advantages from the hardware accelerations. To use a hardware compression capture card will get the higher performance as possible.

*Table 1. The list of supported encoder types:*

Encoder type	descriptions
QCAP_ENCODER_TYPE_SOFTWARE	pure software encoding
QCAP_ENCODER_TYPE_HARDWARE	For <b>Hardware</b> compression capture card only!
QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK	Intel® Media SDK with Intel® platform
QCAP_ENCODER_TYPE_AMD_STREAM	AMD® SDK with AMD® platform
QCAP_ENCODER_TYPE_NVIDIA_CUDA	NVIDIA® CUDA SDK with NVIDIA® graphic card
QCAP_ENCODER_TYPE_NVIDIA_NVENC	NVIDIA® NVENC encoder with NVIDIA® graphic card



To use CUDA SDK, user must copy the CUDA's Dynamically Link Library (DLL) to project folder manually to prevent **DLL** runtime error.

The recording video can save to many popular video format depends on user choice. The current supported audio/video encoding format are list below:

	AVI	MP4	ASF	WMV	FLV	TS	M3U8	SCF	WAV	MP3
<i>Video</i>										
<b>MPEG2</b>						V	V			
<b>H264</b>	V	V	V	V	V	V	V	V		
<b>H264_3D</b>						V	V			
<b>H264_VC</b>	V	V	V	V	V	V	V	V		
<b>RAW</b>	V									
<b>H265</b>						V	V			
<i>Audio</i>										
<b>PCM</b>	V	V	V	V				V	V	
<b>AAC_RAW</b>		V	V	V				V		
<b>AAC_ADTS</b>					V	V	V	V		
<b>MP2</b>		V	V	V		V	V	V		
<b>MP3</b>		V	V	V		V	V	V		V

The default downscale mode is **QCAP\_DOWNSCALE\_MODE\_OFF** and the resolution is 1920x1080 Full HD. The user can set *downscale mode* to scale down to desired resolution. For each device has many recorder slots (**RecNum**) can also be used to record different resolutions or variety bit rates video-streams simultaneously.

Table 2. The list of supported downscale modes:

downscale mode	resolution
QCAP_DOWNSCALE_MODE_OFF	1920x1080 <b>Full HD</b> ( <i>original</i> )
QCAP_DOWNSCALE_MODE_2_3	1280x720 ( <i>downscale by 2/3</i> )
QCAP_DOWNSCALE_MODE_1_2	960x540 ( <i>downscale by 1/2</i> )
QCAP_DOWNSCALE_MODE_1_4	480x270 ( <i>downscale by 1/4</i> )

## Hardware Compression Capture cards

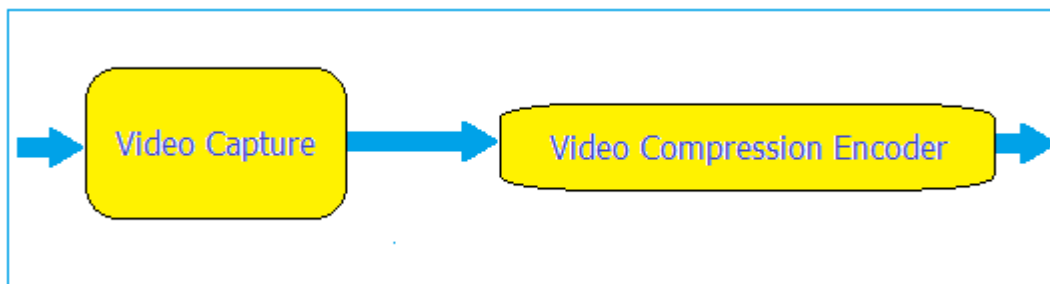
A hardware compression capture card usually delivers many independent output streams for each video input:

- **Capture-only card:**

- one raw uncompressed stream

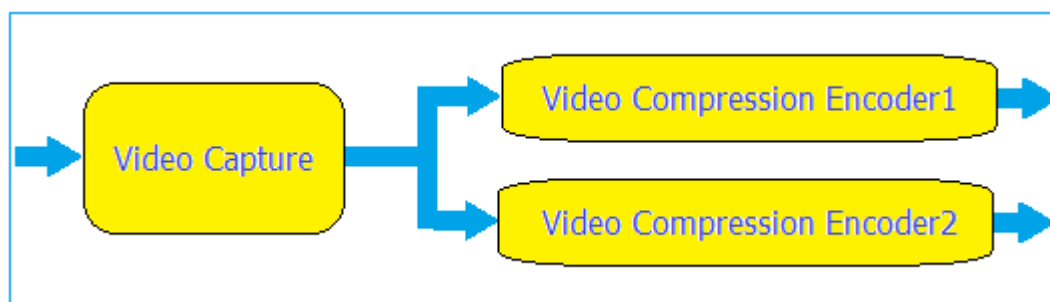
- **Single-stream encoder card:**

- one **H.264** compressed streams for the main encoder (*for single-stream card*)
- Products List: **SC290, SC390, SC3C0, SC5C0, SC580, SC590, SC5A0, SC5B0, SC5C0** and **UB658G**



- **Dual-stream encoder card:**

- one **H.264** compressed streams for the main encoder, and
- one **H.264** compressed streams for sub encoder (*for dual-stream card*)
- Products List: **SC2B0, SC3B0, SC3A0, SC2A0, SC580, SC590, SC5A0, SC5B0, SC5C0**





## Encoder and Recorder Number (RecNum)

Each capture device has many recorder slots can record different resolution or variety bit rates within a capture device simultaneously. The **Recorder Number (RecNum)** is the index to select the i-th recorder slot for recording. The use cases for each **Record Number** (index)\_ for different hardware encoders has little differences:

type	Software encoder	Hardware Single-stream encoder	Hardware Dual-stream encoder
RecNum 0	SW Encode	HW Encode (main) QCAP_ENCODER_TYPE_HARDWARE QCAP_DOWNSCALE_MODE_OFF	HW Encode (main) QCAP_ENCODER_TYPE_HARDWARE QCAP_DOWNSCALE_MODE_OFF
RecNum 1	SW Encode	SW Encode	HW Encode (sub) QCAP_ENCODER_TYPE_HARDWARE QCAP_DOWNSCALE_MODE_1_2
RecNum 2	SW Encode	SW Encode	SW Encode
RecNum 3	SW Encode	SW Encode	SW Encode

A. For capture-only card:

1. All *RecNum* can be used.

B. For single-stream hardware encoder card:

1. *EncoderType* must be **QCAP\_ENCODER\_TYPE\_HARDWARE**, and always use *RecNum* 0 to encode video
2. and *nDownscaleMode* must be **QCAP\_DOWNSCALE\_MODE\_OFF**.

C. For the dual-stream hardware encoder card:

1. *EncoderType* must be **QCAP\_ENCODER\_TYPE\_HARDWARE**, and always use *RecNum* 0 & 1 to encode video
2. *RecNum* 0 is used by main hardware encoder, and *nDownscaleMode* must be **QCAP\_DOWNSCALE\_MODE\_OFF**
3. *RecNum* 1 is used by sub hardware encoder, and *nDownscaleMode* must be **QCAP\_DOWNSCALE\_MODE\_1\_2**

## 5.1 QCAP\_START\_RECORD

### Introduction

The user can use this function to start channel recording from a capture device.

The property setting functions **QCAP\_SET\_AUDIO\_RECORD\_PROPERTY()**, **QCAP\_SET\_VIDEO\_RECORD\_PROPERTY()** must be called to specified recording audio/video properties such as encoder type, video recording bit rate and quality...etc. in prior to **QCAP\_START\_RECORD()** calls.

QCAP support the multi-streams recording at the same time by **Recorder Number (RecNum)**. It means you can select different recorder slots to generate many recording files with different resolutions (downscaled) or variety bit rates within a capture device:

*A example use-case, by using same capture device, you can*

- record 1920x1080 at Recorder Number 0, and
- record 720x480 at Recorder Number 1, at the same time.

The file extension of *pszFilePathName* string directly decides which video format to output the recording stream. For example, "filename.MP4" records in **MP4** video format. If user want to use start/stop recording time as parts of filename, supported input field descriptors listed below:

start_record time	stop_record time	descriptions
%Y	\$Y	year (for example 2016)
%M	\$M	month (1-12)
%D	\$D	day (1-31)
%h	\$h	hours (0-23)
%m	\$m	minutes (0-59)
%s	\$s	seconds (0-59)
%i	\$i	milliseconds (0-999)

If user wants to split the video recording file in video time/file size in recording process,

*the segment duration parameters can be useful (this 2 parameter are exclusive):*

- *dSegmentDurationTime* - to split video recording file by time (*in seconds*)
- *nSegmentDurationSizeKB* - to split video recording file by file size (*in Kilo-Bytes*)

If A/V synchronization timing adjustment is needed, you can fine-tune audio/video delay time by *dVideoDelayTime* & *dAudioDelayTime* parameters. T

The *dwFlags* parameter allows you to toggle internal components in the recording process.

For example:

- To display audio/video stream | `QCAP_RECORD_FLAG_FILE` \ `QCAP_RECORD_FLAG_ENCODE`.
- To display video stream from compressed buffer| `QCAP_RECORD_FLAG_DISPLAY` \ `QCAP_RECORD_FLAG_DECODE`.



These property setting functions:

**QCAP\_SET\_AUDIO\_RECORD\_PROPERTY()**,

**QCAP\_SET\_VIDEO\_RECORD\_PROPERTY()** must be called before **QCAP\_START\_RECORD()** calls.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to start recording, start from 0 range is 0-3.
CHAR *	pszFilePathName	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3</b>
DWORD	dwFlags	IN	<b>default QCAP_RECORD_FLAG_FULL</b> Specify the action flag of channel recording, can be combinations of the following value: QCAP_RECORD_FLAG_FULL QCAP_RECORD_FLAG_FILE QCAP_RECORD_FLAG_ENCODE QCAP_RECORD_FLAG_DISPLAY QCAP_RECORD_FLAG_DECODE QCAP_RECORD_FLAG_VIDEO_ONLY QCAP_RECORD_FLAG_AUDIO_ONLY QCAP_RECORD_FLAG_VIDEO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_AUDIO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_IGNORE_FORMAT_CHANGED QCAP_RECORD_FLAG_SYNCHRONIZED_RECORD QCAP_RECORD_FLAG_VIDEO_USE_MEDIA_TIMER QCAP_RECORD_FLAG_AUDIO_USE_MEDIA_TIMER
double	dVideoDelayTime	IN	<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time
double	dSegmentDurationTime	IN	<b>default 0.0</b> Specify the video segment duration time to split in recording file (in seconds)
ULONG	nSegmentDurationSizeKB	IN	<b>default 0</b> Specify the video segment duration to split in recording file (in Kilo-Bytes)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example 1: Start recording video stream to 1280x720 MP4. Use "stop recording time" as a filename.*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_AAC ); //MP4 use AAC audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                12*1024*1024, //bit rate=12M
                                30,
                                0,
                                0,
                                QCAP_DOWNSCALE_MODE_OFF ); //downscale off (1920x1080)

QCAP_START_RECORD( pDevice, 0,
                  "C:/REC_%Y_%M_%D_%h_%m_%s_%i.MP4" //the start recording time as filename
                  );
```

*Example 2: Start recording a video stream in recorder number 1 and split each video file in 10 seconds duration.*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 1, //recorder number 1
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_PCM ); //AVI use PCM audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 1, //recorder number 1
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                9*1024*1024, //bit rate=9M
                                30,
                                0,
                                0,
                                QCAP_DOWNSCALE_MODE_2_3 ); //downscale to 2/3 (1280x720)

QCAP_START_RECORD( pDevice,
                  1, //recorder number 1
                  "C:/REC_%Y_%M_%D_%h_%m_%s_%i.AVI", //the stop recording time as filename
                  QCAP_RECORD_FLAG_FULL,
                  0.0, 0.0, 10.0, 0 );
```

## 5.2 QCAP\_START\_CLONE\_RECORD

### Introduction

The user can use this function to start a clone recording. The "CLONE" means two identical videos will be recorded in two different locations simultaneously. This is useful for users who has high-security concerns and have to clone a video recording constantly to another safe/backup place.

For more detailed parameters descriptions, please refer to **QCAP\_START\_RECORD()**.



These property setting functions:

**QCAP\_SET\_AUDIO\_RECORD\_PROPERTY()**,

**QCAP\_SET\_VIDEO\_RECORD\_PROPERTY()** must be called before **QCAP\_START\_CLONE\_RECORD()** calls.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to start recording, start from 0. Range is 0-31
CHAR *	pszFilePathNameA	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3</b>
CHAR *	pszFilePathNameB	IN	Specify the clone file name for recording. (please refer to <i>pszFilePathNameA</i> )
DWORD	dwFlags	IN	<b>default QCAP_RECORD_FLAG_FULL</b> Specify the action flag of channel recording, can be combinations of the following value: QCAP_RECORD_FLAG_FULL QCAP_RECORD_FLAG_FILE QCAP_RECORD_FLAG_ENCODE QCAP_RECORD_FLAG_DISPLAY QCAP_RECORD_FLAG_DECODE QCAP_RECORD_FLAG_VIDEO_ONLY QCAP_RECORD_FLAG_AUDIO_ONLY QCAP_RECORD_FLAG_VIDEO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_AUDIO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_IGNORE_FORMAT_CHANGED QCAP_RECORD_FLAG_SYNCHRONIZED_RECORD QCAP_RECORD_FLAG_VIDEO_USE_MEDIA_TIMER QCAP_RECORD_FLAG_AUDIO_USE_MEDIA_TIMER
double	dVideoDelayTime	IN	<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time
double	dSegmentDurationTime	IN	<b>default 0.0</b> Specify the video segment duration time to split in recording file (in seconds)
ULONG	nSegmentDurationSizeKB	IN	

**default 0**

Specify the video segment duration to split in recording file (in Kilo-Bytes)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example 1: start clone recording to 2 different locations from the same video stream*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_AAC ); //MP4 use AAC audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                12*1024*1024, //bit rate=12M
                                30,
                                0,
                                0,
                                QCAP_DOWNSCALE_MODE_OFF ); //downscale off (1920x1080)

QCAP_START_CLONE_RECORD( pDevice, 0, "C:/CH01.AVI", "D:/CH01.AVI" );

QCAP_START_CLONE_RECORD( pDevice,
                          0,
                          "C:/REC_%Y_%M_%D_%h_%m_%s%i.mp4", //recording
                          "D:/REC_%Y_%M_%D_%h_%m_%s%i.mp4", //recording (clone)
                          QCAP_RECORD_FLAG_FULL,
                          0.0, //dVideoDelayTime
                          0.0, //dAudioDelayTime
                          0.0, //segment duration time
                          0 ); //segment duration size
```

## 5.3 QCAP\_START\_TIMESHIFT\_RECORD

### Introduction

The user can use this function to start time-shifting recording. That means a user can continue recording one video while playing back the same video recording file. In another word, a user can playback the video in recording on-the-fly, just like time-shifted.

The *ppPhysicalFileWriter* parameter returns a file handle in which a time-shifting recording is in progress. This file handle could be used by **QCAP\_OPEN\_TIMESHIFT\_FILE\_EX()** and other playback functions.

This time-shifted functionality current supported **MP4** format only.

For more detailed parameters descriptions, please refer to:

\* **QCAP\_START\_RECORD()**

\* **QCAP\_OPEN\_TIMESHIFT\_FILE\_EX()**



These property setting functions:

**QCAP\_SET\_AUDIO\_RECORD\_PROPERTY()**,

**QCAP\_SET\_VIDEO\_RECORD\_PROPERTY()** must be called before **QCAP\_START\_TIMESHIFT\_RECORD()** calls.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to start recording, start from 0. Range is 0-3
CHAR *	pszFilePathName	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3
PVOID *	ppPhysicalFileWriter	OUT	Handle of Physical File Writer object
DWORD	dwFlags	IN	<b>default QCAP_RECORD_FLAG_FULL</b> Specify the action flag of channel recording, can be combinations of the following value: QCAP_RECORD_FLAG_FULL QCAP_RECORD_FLAG_FILE QCAP_RECORD_FLAG_ENCODE QCAP_RECORD_FLAG_DISPLAY QCAP_RECORD_FLAG_DECODE QCAP_RECORD_FLAG_VIDEO_ONLY QCAP_RECORD_FLAG_AUDIO_ONLY QCAP_RECORD_FLAG_VIDEO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_AUDIO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_IGNORE_FORMAT_CHANGED QCAP_RECORD_FLAG_SYNCHRONIZED_RECORD QCAP_RECORD_FLAG_VIDEO_USE_MEDIA_TIMER QCAP_RECORD_FLAG_AUDIO_USE_MEDIA_TIMER
double	dVideoDelayTime	IN	

			<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : start a time-shifted recording for **MP4**, and playback the video (still in recording)*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_AAC ); //MP4 use AAC audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                12*1024*1024, //bit rate=12M
                                30,
                                0,
                                0,
                                QCAP_DOWNSCALE_MODE_OFF ); //downscale off (1920x1080)

PVOID pPhysicalFileWriter = NULL;

PVOID pFile = NULL;

QCAP_START_TIMEShift_RECORD( pDevice, 0,
                             "D:/CH01.MP4",
                             &pPhysicalFileWriter );

QCAP_OPEN_TIMEShift_FILE_EX( pPhysicalFileWriter,
                             &pFile, QCAP_DECODER_TYPE_SOFTWARE, &nVideoFormat,
                             &nVideoWidth, &nVideoHeight, &dVideoFrameRate,
                             &nAudioFormat, &nAudioChannels,
                             &nAudioBitsPerSample, &nAudioSampleFrequency,
                             &dFileTotalDuationTimes,
                             &nFileTotalVideoFrames, &nFileTotalAudioFrames,
                             &nFileTotalMetadataFrames,
                             hWnd, 1 );

QCAP_PLAY_FILE( m_pFile ); //playback the video (still in time-shifted recording)
```



## 5.2 QCAP\_START\_FAILSAFE\_RECORD

### Introduction

This function can do failsafe recording to save video stream to file while time-shift recording is in progress. The user can use this function to record any duration of time-shift video streams into a new video file to ensure the video data is well saved and closed.

For more detailed parameters descriptions, please refer to **QCAP\_START\_TIMESHIFT\_RECORD()**.



These property setting functions:

**QCAP\_SET\_AUDIO\_RECORD\_PROPERTY()**,

**QCAP\_SET\_VIDEO\_RECORD\_PROPERTY()** must be called before **QCAP\_START\_FAILSAFE\_RECORD()** calls.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to start failsafe recording, start from 0. Range is 0-31
CHAR *	pszFilePathNameA	IN	Specify the new file name for failsafe recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3</b>
UINT	iLinkRecNum	IN	The recorder slot used in time-shift recording
PVOID	pLinkPhysicalFileWriter	IN	Handle of Physical File Writer object of time-shift recording
double	dPreRecordTime	IN	<b>default 0.0</b> Specify the last N seconds from now to do failsafe recording

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example 1: start a fail safe video recording*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_AAC ); //MP4 use AAC audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                12*1024*1024, //bit rate=12M
                                30,
                                0,
                                0,
                                QCAP_DOWNSCALE_MODE_OFF ); //downscale off (1920x1080)

QCAP_START_FAILSAFE_RECORD( pDevice,
                            1, //the new record number for failsafe recording
                            "C:/REC_%Y_%M_%D_%h_%m_%s_%i.mp4", //recording
                            0,
                            pPhysicalFileWriter,
                            0.0 );
```

## 5.4 QCAP\_PAUSE\_RECORD

### Introduction

The user can use this function to pause channel recording.

If a user starts many recording in multiple recorder slots, the *iRecNum* parameter can use to choose which recorder slot to pause. To resume channel recording for a recorder slot please call **QCAP\_RESUME\_RECORD()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to pause recording, start from 0. Range is 0-3

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

---

## 5.5 QCAP\_RESUME\_RECORD

### Introduction

The user can use this function to resume a previously paused channel recording.

If the user starts many recording in multiple recorder slots, the *iRecNum* parameter can use to choose which recorder slot to resume.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to resume recording, start from 0. Range is 0-3

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Pause/Resume a previously paused channel recording in recorder slot 0*

```
QCAP_PAUSE_RECORD( pDevice, 0 );

QCAP_RESUME_RECORD( pDevice, 0 );
```

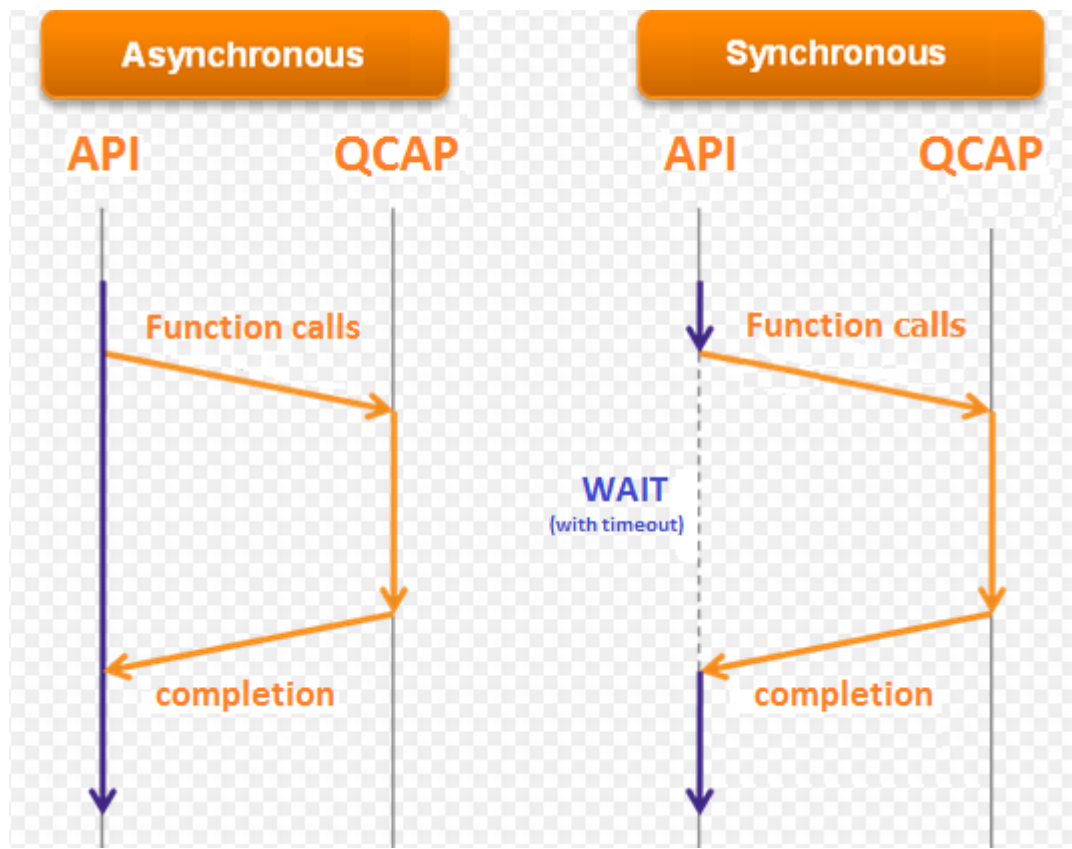
# 5.6 QCAP\_STOP\_RECORD

## Introduction

The user can use this function to stop channel recording from a capture device.

If a user starts many recording in multiple recorder slots, the *iRecNum* parameter can be used to choose which recorder slot to stop.

For multiple recorder slot in the process, a user need to call **QCAP\_STOP\_RECORD()** for each recorder slot in order to stop all recordings.



This function is designed for both synchronous and asynchronous operations. The user can decide to wait until completion (sync mode) or stop recording in background (async mode):

1. In Asynchronous mode( *blsAsync* = TRUE ):
  - function returns immediately.
2. In Synchronous mode ( *blsAsync* = FALSE ):
  - if *nMillisecond* is INFINITE, the function will be blocked until the recording is completed.
  - If *nMillisecond* is 0, the function returns immediately and generates video file when next frame is coming.



For .Net developer, the INFINITE is defined as 0xFFFFFFFF.

## Parameters

type	parameter	I/O	descriptions

PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to stop recording, start from 0. Range is 0-3
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Stop a channel recording at record\_index 0 & 1*

```
QCAP_STOP_RECORD( pDevice, 0 );

QCAP_STOP_RECORD( pDevice, 1 );
```

## 5.7 Channel Record Data Functions

### 5.7.1 QCAP\_SET\_METADATA\_RECORD\_DATA\_BUFFER

#### Introduction

While channel recording each frame can send a user-defined information, or meta-data, that could output pre-frames to the video file. Then the meta-data can be retrieved by calling **QCAP\_GET\_METADATA\_FILE\_DATA\_BUFFER()** for different applications. This function can be called any time (*in a video callback, timer handlers*) while recording is in progress.



Meta-data buffer can only set to frame while recording is in progress.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recording
BYTE *	pDataBuffer	IN	pointer to user-define meta-data buffer
ULONG	nDataBufferSize	IN	Specify the length of user-define meta-data buffer
double	dSampleTime	IN	<b>default 0.0</b> the sampling time in seconds

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set user-defined meta-data buffer to the beginning of video frame while recording*

```
char *user_buffer = "this is the user-defined string to store in the first video frame.";

QCAP_START_RECORD( pDevice, 0, "C:/REC_1080_12M.MP4" );

QCAP_SET_METADATA_RECORD_DATA_BUFFER( pDevice, 0,
                                     user_buffer,
                                     strlen(user_buffer),
                                     0 );
```

### 5.7.1 QCAP\_SET\_METADATA\_RECORD\_HEADER

#### Introduction

This function can set the header information in file record header.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0

CHAR *	ppszTitle	OUT	The title information in file header
CHAR *	ppszArtist	OUT	The artist information in file header
CHAR *	ppszComments	OUT	The comment information in file header
CHAR *	ppszGenre	OUT	The genre information in file header
CHAR *	ppszComposer	OUT	The composer information in file header

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : To set the meta-data information to file header.*

```
QCAP_SET_METADATA_RECORD_HEADER( pDevice, 0,
                                title, artist, comment, genre, composer );
```

# 5.8 Channel Record Property Functions

## Introduction

Before you want to create a video recording file, you will want to apply properties/setting so that the recording functions will output the effect you want. The functions to access software / hardware encoder properties for **Channel Recording** were designed *in different sets of functions*.

To access the audio/video channel recording properties functions:

mode	functions
For Software Encoder	
Set encoder properties	QCAP_SET_VIDEO_RECORD_PROPERTY()
	QCAP_SET_VIDEO_RECORD_PROPERTY_EX()
	QCAP_SET_AUDIO_RECORD_PROPERTY()
	QCAP_SET_AUDIO_RECORD_PROPERTY_EX()
Get encoder properties	QCAP_SET_VIDEO_RECORD_DYNAMIC_PROPERTY_EX()
	QCAP_GET_VIDEO_RECORD_PROPERTY()
	QCAP_GET_VIDEO_RECORD_PROPERTY_EX()
	QCAP_GET_AUDIO_RECORD_PROPERTY()
Get encoder properties	QCAP_GET_AUDIO_RECORD_PROPERTY_EX()
	QCAP_GET_VIDEO_RECORD_DYNAMIC_PROPERTY_EX()
For Hardware Encoder	
Set encoder properties	QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY()
	QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()
Get encoder properties	QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY()
	QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()



## 5.8.1 QCAP\_SET\_VIDEO\_RECORD\_PROPERTY

## 5.8.2 QCAP\_SET\_VIDEO\_RECORD\_PROPERTY\_EX

### Introduction

The property setting functions **QCAP\_SET\_AUDIO\_RECORD\_PROPERTY()**, **QCAP\_SET\_VIDEO\_RECORD\_PROPERTY()** must be called in prior to **QCAP\_START\_RECORD()** calls. The user can use this function to set channel video recording parameters, such as *encoder type, video recording bit rate, and quality, H.264 encoder setting, Profile, Level, Entropy, Complexity, and B-Frames...etc.*

In multi-threaded application, QCAP will balance CPU loading when *bMultiThread* is true.

When using **INTEL\_MEDIA\_SDK** as the encoder, the *MBBRC* and *ExtBRC* parameters could set to true to get better video quality in low bit rate mode on Intel® platform.



In channel recording, this function is For **Software encoder** only!

For hardware encoder must use **QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX()**

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters, start from 0. Range is 0-3
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE QCAP_ENCODER_TYPE_HARDWARE QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK QCAP_ENCODER_TYPE_AMD_STREAM QCAP_ENCODER_TYPE_NVIDIA_CUDA QCAP_ENCODER_TYPE_NVIDIA_NVENC <b>Note: For Hardware encoder must use</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nEncoderFormat	IN	Specify video encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800
ULONG	nRecordProfile	IN	<b>default QCAP_RECORD_PROFILE_BASELINE</b> Specify recording profile: QCAP_RECORD_PROFILE_BASELINE QCAP_RECORD_PROFILE_MAIN QCAP_RECORD_PROFILE_HIGH

			QCAP_RECORD_PROFILE_CONSTRAINED_BASELINE QCAP_RECORD_PROFILE_CONSTRAINED_HIGH <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nRecordLevel	IN	<b>default QCAP_RECORD_LEVEL_41</b> Specify recording levels: QCAP_RECORD_LEVEL_1 QCAP_RECORD_LEVEL_1B QCAP_RECORD_LEVEL_11 QCAP_RECORD_LEVEL_12 QCAP_RECORD_LEVEL_13 QCAP_RECORD_LEVEL_2 QCAP_RECORD_LEVEL_21 QCAP_RECORD_LEVEL_22 QCAP_RECORD_LEVEL_3 QCAP_RECORD_LEVEL_31 QCAP_RECORD_LEVEL_32 QCAP_RECORD_LEVEL_4 QCAP_RECORD_LEVEL_41 QCAP_RECORD_LEVEL_42 QCAP_RECORD_LEVEL_50 QCAP_RECORD_LEVEL_51 QCAP_RECORD_LEVEL_52 QCAP_RECORD_LEVEL_60 QCAP_RECORD_LEVEL_61 QCAP_RECORD_LEVEL_62 <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nRecordEntropy	IN	<b>default QCAP_RECORD_ENTROPY_CAVLC</b> Specify recording entropy: QCAP_RECORD_ENTROPY_CAVLC QCAP_RECORD_ENTROPY_CABAC <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nRecordComplexity	IN	<b>default QCAP_RECORD_COMPLEXITY_0</b> Specify recording complexity: QCAP_RECORD_COMPLEXITY_0 (Best Speed) QCAP_RECORD_COMPLEXITY_1 QCAP_RECORD_COMPLEXITY_2 QCAP_RECORD_COMPLEXITY_3 QCAP_RECORD_COMPLEXITY_4 QCAP_RECORD_COMPLEXITY_5 QCAP_RECORD_COMPLEXITY_6 (Best Quality) <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify recording quality, from 0-10000. It is only for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify recording bit rate. It is only for <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording GOP size, from 0-255
ULONG	nBFrames	IN	

			<b>default 0</b> Specify recording B-Frames <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL	bIsInterleaved	IN	<b>default FALSE</b> Specify recording interleaved <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nSlices	IN	<b>default 0</b> Specify recording slices <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nLayers	IN	<b>default 0</b> Specify recording layers <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nSceneCut	IN	<b>default 0</b> Specify recording Scene Cut, recommended value is 40. Set 0 to turn off. <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL	bMultiThread	IN	<b>default TRUE</b> Enable/Disable the multi-threaded CPU loading balance support <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL	bMBBRC	IN	<b>default FALSE</b> Enable/Disable the mbbrc <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL	bExtBRC	IN	<b>default FALSE</b> Enable/Disable the extbrc <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nMinQP	IN	<b>default 0</b> Specify the value of x264 Minimum quantizer settings <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nMaxQP	IN	<b>default 0</b> Specify the value of x264 Maximum quantizer settings <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nVBVMaxRate	IN	<b>default 0</b> Specify the value that x264 fills the buffer at (up to) the max rate <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nVBVBufSize	IN	<b>default 0</b> Specify the size that x264 fills the buffer <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nAspectRatioX	IN	Specify the aspect ratio X axis, 0 to turned off
ULONG	nAspectRatioY	IN	Specify the aspect ratio Y axis, 0 to turned off
ULONG	nDownscaleMode	IN	Specify recording downscale mode: QCAP_DOWNSCALE_MODE_OFF QCAP_DOWNSCALE_MODE_2_3 QCAP_DOWNSCALE_MODE_1_2 QCAP_DOWNSCALE_MODE_1_4 <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY()</b>
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>

ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nDstW	IN	Specify the width of the downscaled frame <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nDstH	IN	Specify the height of the downscaled frame <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nPostSkipFrameRate	IN	<b>default 0</b> Specify the post skip frame rate <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG	nPostAvgFrameRate	IN	<b>default 0</b> Specify the post average frame rate <b>Only in QCAP_SET_VIDEO_RECORD_PROPERTY_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example 1: Set the audio/video recording properties before video recording*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_AAC ); //MP4 use AAC audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                12*1024* 1024,
                                30,
                                4,
                                3,
                                QCAP_DOWNSCALE_MODE_OFF ); //downscale mode

QCAP_START_RECORD( pDevice, 0,
                   "C:/REC_%Y_%M_%D_%h_%m_%s_%i.MP4" //the start recording time as filename
                   );
```

Example 2: Set the audio/video recording properties before video recording (use extended function)

```
QCAP_SET_AUDIO_RECORD_PROPERTY_EX( pDevice, 0,
                                   QCAP_ENCODER_TYPE_SOFTWARE,
                                   QCAP_ENCODER_FORMAT_PCM,
                                   12*1024 );

QCAP_SET_VIDEO_RECORD_PROPERTY_EX( pDevice, 0,
                                   QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                   QCAP_ENCODER_FORMAT_H264,
                                   nRecordProfile,
                                   nRecordLevel,
                                   nRecordEntropy,
                                   1,
                                   QCAP_RECORD_MODE_CBR,
                                   8000,
                                   12*1024* 1024,
                                   30,
                                   0,
                                   FALSE,
                                   0,
                                   0,
                                   40,
                                   0,
                                   0,
                                   0,
                                   0,0,0,0,           //MinQP,MaxQP, VBVmaxRate,VBVBufSize
                                   4, 3,               //AspectRatioX,Y
                                   10, 40, 1900, 1000, //source cropping X,Y,W,H
                                   720, 480 );         //destination W,H

QCAP_START_RECORD( pDevice, 0,
                  "C:/REC_%Y_%M_%D_%h_%m_%s_%i.AVI" ); //the start recording time as filename
```

## 5.8.3 QCAP\_GET\_VIDEO\_RECORD\_PROPERTY

## 5.8.4 QCAP\_GET\_VIDEO\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to get video channel recording parameters.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_VIDEO\_RECORD\_PROPERTY\_EX()**.



In channel recording, this function is For **Software encoder** only!

For hardware encoder must use **QCAP\_GET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX()**

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to get recording parameters, start from 0. Range is 0-3
ULONG *	pEncoderType	OUT	Pointer to the encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the encoder format
ULONG *	pRecordProfile	OUT	Pointer to the recording profile <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordLevel	OUT	Pointer to the recording level <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordEntropy	OUT	Pointer to the recording entropy <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordComplexity	OUT	Pointer to the recording complexity <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordMode	OUT	Pointer to the recording mode
ULONG *	pQuality	OUT	Pointer to the quality
ULONG *	pBitRate	OUT	Pointer to the bit rate
ULONG *	pGOP	OUT	Pointer to the GOP size
ULONG *	pBFrames	OUT	Pointer to the current B-Frames <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL *	pIsInterleaved	OUT	Pointer to the current interleave <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pSlices	OUT	Pointer to the current slices <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pLayers	OUT	Pointer to the current layers <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pSceneCut	OUT	Pointer to the current scene cut <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL *	pMultiThread	OUT	Pointer to the current multi-threaded CPU loading balance status <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
BOOL *	pMBBRC	OUT	Pointer to the current mbbrc status <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>

BOOL *	pExtBRC	OUT	Pointer to the current extbrc status <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pMinQP	OUT	Pointer to the value of x264 Minimum quantizer settings <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pMaxQP	OUT	Pointer to the value of x264 Maximum quantizer settings <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pVBVMaxRate	OUT	Pointer to the value that x264 fills the buffer at (up to) the max rate <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pVBVBufSize	OUT	Pointer to the size that x264 fills the buffer <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pAspectRatioX	OUT	Pointer to the aspect ratio X axis
ULONG *	pAspectRatioY	OUT	Pointer to the aspect ratio Y axis
ULONG *	pDownscaleMode	OUT	Pointer to the downscale mode <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY()</b>
ULONG *	pCropX	OUT	Pointer to the x-coordinate of the crop <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pCropY	OUT	Pointer to the y-coordinate of the crop <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pCropW	OUT	Pointer to the horizontal width of crop <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pCropH	OUT	Pointer to the vertical height of crop <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pDstW	OUT	Pointer to the destination horizontal width <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pDstH	OUT	Pointer to the destination vertical height <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pPostSkipFrameRate	OUT	Pointer to post skip frame rate <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>
ULONG *	pPostAvgFrameRate	OUT	Pointer to post average frame rate <b>Only in QCAP_GET_VIDEO_RECORD_PROPERTY_EX()</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get the current video recording properties in channel recording*

[illegible]



## 5.8.5 QCAP\_SET\_AUDIO\_RECORD\_PROPERTY

## 5.8.6 QCAP\_SET\_AUDIO\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to set audio recording parameters in the current channel.

For the most popular **MP4** file recording, we suggested to use **H.264 + AAC** audio format. It is default file format in Microsoft® Windows 7. Since the **AAC** codec is licensed, if a customer wants to avoid **AAC** licenses, the QCAP SDK also support **H.264 + PCM** audio format for general purpose recording.

For developer uses **TS, FLV, RTMP** and **HLS**, must set audio encoder format to **QCAP\_ENCODER\_FORMAT\_AAC\_ADTS**.



All audio data use **software encoder** in recording.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to get recording parameters, start from 0. Range is 0-3
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE
ULONG	nEncoderFormat	IN	Specify audio encoder format: QCAP_ENCODER_FORMAT_PCM QCAP_ENCODER_FORMAT_AAC QCAP_ENCODER_FORMAT_AAC_RAW QCAP_ENCODER_FORMAT_AAC_ADTS QCAP_ENCODER_FORMAT_MP2 QCAP_ENCODER_FORMAT_MP3 QCAP_ENCODER_FORMAT_OPUS
ULONG	nBitRate	IN	<b>default 128K</b> Specify audio bit rate, the maximum value is 496K <b>Only in QCAP_SET_AUDIO_RECORD_PROPERTY_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set audio recording properties in channel recording*

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice,  
                                0,  
                                QCAP_ENCODER_TYPE_SOFTWARE,  
                                QCAP_ENCODER_FORMAT_AAC);  
  
QCAP_SET_AUDIO_RECORD_PROPERTY_EX( pDevice,  
                                    0,  
                                    QCAP_ENCODER_TYPE_SOFTWARE,  
                                    QCAP_ENCODER_FORMAT_PCM,  
                                    12*1024 );
```

## 5.8.7 QCAP\_GET\_AUDIO\_RECORD\_PROPERTY

## 5.8.8 QCAP\_GET\_AUDIO\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to get audio record property parameters in the current channel.



All audio data use **software encoder** in recording.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to get recording parameters, start from 0. Range is 0-3
ULONG *	pEncoderType	OUT	Pointer to the audio encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the audio encoder format
ULONG *	pBitRate	OUT	Pointer to the audio bit rate
			<b>Only in QCAP_GET_AUDIO_RECORD_PROPERTY_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Retrieve current audio recording properties in channel recording*

```
QCAP_GET_AUDIO_RECORD_PROPERTY( pDevice, 0, &nEncoderType, &nEncoderFormat );

QCAP_GET_AUDIO_RECORD_PROPERTY_EX( pDevice, 0, &nEncoderType, &nEncoderFormat, &nBitRate );
```

## 5.8.9 QCAP\_SET\_VIDEO\_RECORD\_DYNAMIC\_PROPERTY\_EX

### Introduction

This function is only for software encoders to set current channel recording properties **dynamically**, such as *bit rate* & *GOP reconfiguration... etc.*



In channel recording, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters, start from 0 Range is 0-3
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify recording software encoder quality, from 0-10000. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify recording software encoder bit rate. It is used for <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording software encoder GOP size, from 0-255

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the current setting of channel recording properties in run-time*

```
QCAP_SET_VIDEO_RECORD_DYNAMIC_PROPERTY_EX( pDevice,  
                                           0,  
                                           1,  
                                           8000,  
                                           12*1024*1024,  
                                           30 );
```

C



## 5.8.11 QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY

## 5.8.12 QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX

### Introduction

If a user is using hardware encode capture card, then **QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY()** function can be used to set basic video hardware recording parameters.

The extended function get supports the cropping function and the various upscale/downscale function, advanced **H.264** Encoder setting, such as *Profile, Level, Entropy, B-Frames, and SceneCut... etc.*



This function is For **Hardware encoder** only!

For **Software encoder** must use **QCAP\_SET\_VIDEO\_RECORD\_PROPERTY\_EX()**

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to set recording parameters
ULONG	nEncoderFormat	IN	Specify video encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800
ULONG	nRecordProfile	IN	Specify recording profile: QCAP_RECORD_PROFILE_BASELINE QCAP_RECORD_PROFILE_MAIN QCAP_RECORD_PROFILE_HIGH QCAP_RECORD_PROFILE_CONSTRAINED_BASELINE. QCAP_RECORD_PROFILE_CONSTRAINED_HIGH <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nRecordLevel	IN	Specify recording levels: QCAP_RECORD_LEVEL_1 QCAP_RECORD_LEVEL_1B QCAP_RECORD_LEVEL_11 QCAP_RECORD_LEVEL_12 QCAP_RECORD_LEVEL_13 QCAP_RECORD_LEVEL_2 QCAP_RECORD_LEVEL_21 QCAP_RECORD_LEVEL_22 QCAP_RECORD_LEVEL_3 QCAP_RECORD_LEVEL_31

			QCAP_RECORD_LEVEL_32 QCAP_RECORD_LEVEL_4 QCAP_RECORD_LEVEL_41 QCAP_RECORD_LEVEL_42 QCAP_RECORD_LEVEL_50 QCAP_RECORD_LEVEL_51 QCAP_RECORD_LEVEL_52 QCAP_RECORD_LEVEL_60 QCAP_RECORD_LEVEL_61 QCAP_RECORD_LEVEL_62 <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nRecordEntropy	IN	default QCAP_RECORD_ENTROPY_CAVLC Specify recording entropy: QCAP_RECORD_ENTROPY_CAVLC QCAP_RECORD_ENTROPY_CABAC <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify recording quality, from 0-10000. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify recording bit rate. It is used for <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording GOP size, range from 0-255
ULONG	nBFrames	IN	Specify recording B-Frames <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL	bIsInterleaved	IN	Specify recording interleaved <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nSlices	IN	Specify recording slices <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nLayers	IN	Specify recording layers <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nSceneCut	IN	Specify recording Scene Cut, recommended value is 40, 0 is function turned off <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL	bMultiThread	IN	Enable/Disable the multi-threaded CPU loading balance support For <b>Software encoder</b> only, not useful in hardware encoder. <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL	bMBBRC	IN	Enable/Disable the mbbrc For <b>Software encoder</b> only, not useful in hardware encoder. <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL	bExtBRC	IN	

			Enable/Disable the extbrc For <b>Software encoder</b> only, not useful in hardware encoder. <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nMinQP	IN	<b>default 0</b> Specify the value of x264 Minimum quantizer settings <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nMaxQP	IN	<b>default 0</b> Specify the value of x264 Maximum quantizer settings <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nVBVMaxRate	IN	<b>default 0</b> Specify the value that x264 fills the buffer at (up to) the max rate <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nVBVBufSize	IN	<b>default 0</b> Specify the size that x264 fills the buffer <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nAspectRatioX	IN	Specify the aspect ratio X axis, set 0 to turn off
ULONG	nAspectRatioY	IN	Specify the aspect ratio Y axis, set 0 to turn off
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nDownscaleMode	IN	Specify recording downscale mode: QCAP_DOWNSCALE_MODE_OFF QCAP_DOWNSCALE_MODE_2_3 QCAP_DOWNSCALE_MODE_1_2 QCAP_DOWNSCALE_MODE_1_4 <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY()</b>
ULONG	nDstW	IN	Specify the width of the downscaled frame <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nDstH	IN	Specify the height of the downscaled frame <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG	nPostSkipFrameRate	IN	<b>default 0</b> Specify the post skip frame rate
ULONG	nPostAvgFrameRate	IN	<b>default 0</b> Specify the post average frame rate



## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the hardware encoding properties for capturing video*

```
QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY( pDevice, 0,
    QCAP_ENCODER_FORMAT_H264, //nEncoderFormat
    QCAP_RECORD_MODE_CBR, //nRecordMode
    8000, //nQuality
    12 x 1024 x 1024, //nBitRate
    30, //nGOP
    0, //nAspectRatioX
    0, //nAspectRatioY
    QCAP_DOWNSCALE_MODE_1_2, //nDownscaleMode
    0, //nPostSkipFrameRate
    0 //nPostAvgFrameRate
);

QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX( pDevice, 0,
    QCAP_ENCODER_FORMAT_H264, //nEncoderFormat
    QCAP_RECORD_PROFILE_HIGH, //nRecordProfile,
    QCAP_RECORD_LEVEL_41, //nRecordLevel,
    QCAP_RECORD_ENTROPY_CAVLC, //nRecordEntropy,
    QCAP_RECORD_MODE_CBR, //nRecordMode
    8000, //nQuality
    12 x 1024 x 1024, //nBitRate
    30, //nGOP
    0, //nBFrames
    FALSE, //nIsInterleaved
    0, //nSlices
    0, //nLayers
    0, //nSceneCut
    FALSE, //nMultiThread
    FALSE, //nMBBRC
    FALSE, //nExtBRC
    0, //nMinQP
    0, //nMaxQP
    0, //nVBVMaxRate
    0, //nVBVBufSize
    0, //nAspectRatioX
    0, //nAspectRatioY
    0,0,0,0, //nCropX, nCropY, nCropW, nCropH
    720, 540, //nDstW, nDstH
    0, //nPostSkipFrameRate
    0 //nPostAvgFrameRate
);
```

## 5.8.13 QCAP\_GET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY

## 5.8.14 QCAP\_GET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX

### Introduction

If a user is using hardware encoder capture card, a user can use this function to get video hardware recording parameters setting.

For detailed parameters definition please refer to **QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX()** function.



This function is For **Hardware encoder** only!

For software encoder must use **QCAP\_GET\_VIDEO\_RECORD\_PROPERTY\_EX()**

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot to get recording parameters, start from 0
ULONG *	pEncoderFormat	OUT	Pointer to video encoder format
ULONG *	pRecordProfile	OUT	Pointer to recording profile <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pRecordLevel	OUT	Pointer to recording level <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pRecordEntropy	OUT	Pointer to recording entropy <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pRecordMode	OUT	Pointer to recording mode
ULONG *	pQuality	OUT	Pointer to quality
ULONG *	pBitRate	OUT	Pointer to bit rate
ULONG *	pGOP	OUT	Pointer to GOP size
ULONG *	pBFrames	OUT	Pointer to B-Frames <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL *	pIsInterleaved	OUT	Pointer to Interleaved flag <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pSlices	OUT	Pointer to slices <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pLayers	OUT	Pointer to layers <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pSceneCut	OUT	Pointer to screen cut <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL *	pMultiThread	OUT	

			Pointer to current multi-threaded CPU loading balance status <b>For Software encoder only, not useful in hardware encoder.</b> <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL *	pMBBRC	OUT	Pointer to current mbbrc status <b>For Software encoder only, not useful in hardware encoder.</b> <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
BOOL *	pExtBRC	OUT	Pointer to current extbrc status <b>For Software encoder only, not useful in hardware encoder.</b> <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pMinQP	OUT	Pointer to the value of x264 Minimum quantizer settings <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pMaxQP	OUT	Pointer to the value of x264 Maximum quantizer settings <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pVBVMaxRate	OUT	Pointer to the value that x264 fills the buffer at (up to) the max rate <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pVBVBufSize	OUT	Pointer to the size that x264 fills the buffer <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pAspectRatioX	OUT	Pointer to aspect ratio X axis
ULONG *	pAspectRatioY	OUT	Pointer to aspect ratio Y axis
ULONG *	pCropX	OUT	Pointer to x-coordinate of the crop <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pCropY	OUT	Pointer to y-coordinate of the crop <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pCropW	OUT	Pointer to horizontal width of crop <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pCropH	OUT	Pointer to vertical height of crop <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pDownscaleMode	OUT	Pointer to downscale mode <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY()</b>
ULONG *	pDstW	OUT	Pointer to the destination horizontal width <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pDstH	OUT	Pointer to the destination vertical height <b>Only in</b> <b>QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()</b>
ULONG *	pPostSkipFrameRate	OUT	Pointer to post skip frame rate
ULONG *	pPostAvgFrameRate	OUT	Pointer to post average frame rate

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

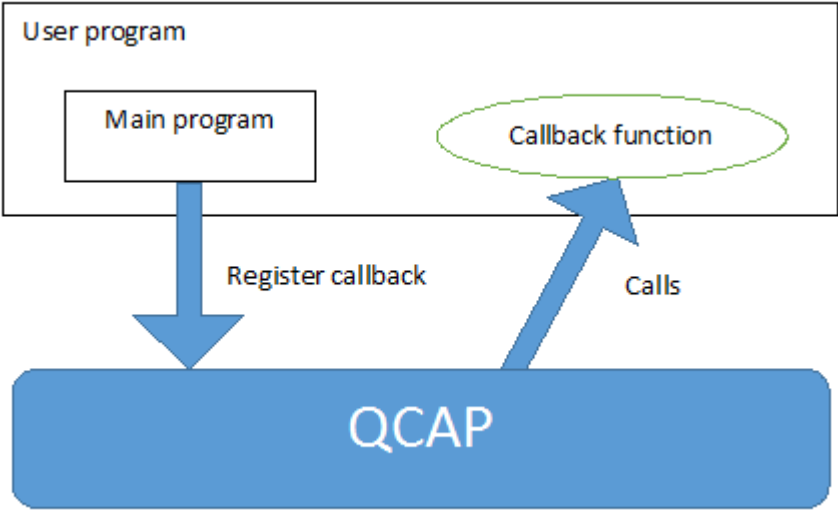
*Example : Get the hardware encoding properties for capturing video*

```
QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY( pDevice, 0,
                                           &pEncoderFormat,
                                           &pRecordMode,
                                           &pQuality,
                                           &pBitRate,
                                           &pGOP,
                                           &pAspectRatioX,
                                           &pAspectRatioY,
                                           &pDownscaleMode,
                                           &pPostSkipFrameRate,
                                           &pPostAvgFrameRate,
                                           );

QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX( pDevice, 0,
                                              &pEncoderFormat,
                                              &pRecordProfile, //extended
                                              &pRecordLevel, //extended
                                              &pRecordEntropy, //extended
                                              &pRecordMode,
                                              &pQuality,
                                              &pBitRate,
                                              &pGOP,
                                              &pBFrames, //extended
                                              &pIsInterleaved, //extended
                                              &pSlices, //extended
                                              &pLayers, //extended
                                              &pSceneCut, //extended
                                              &pMultiThread, //extended
                                              &pMBBRC, //extended
                                              &pExtBRC, //extended
                                              &pMinQP,
                                              &pMaxQP,
                                              &pVBVMaxRate,
                                              &pVBVBufSize, //extended
                                              &pAspectRatioX,
                                              &pAspectRatioY,
                                              &pCropX, //extended
                                              &pCropY, //extended
                                              &pCropW, //extended
                                              &pCropH, //extended
                                              &pDstW, &nDstH, //extended
                                              &pPostSkipFrameRate,
                                              &pPostAvgFrameRate,
                                              );
```

# 5.9 Channel Record Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_RECORD_DONE_CALLBACK	PF_RECORD_DONE_CALLBACK
QCAP_REGISTER_RECORD_FAIL_CALLBACK	PF_RECORD_FAIL_CALLBACK
QCAP_REGISTER_VIDEO_RECORD_CALLBACK	PF_VIDEO_RECORD_CALLBACK
QCAP_REGISTER_AUDIO_RECORD_CALLBACK	PF_AUDIO_RECORD_CALLBACK
QCAP_REGISTER_MEDIA_RECORD_CALLBACK	PF_MEDIA_RECORD_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

# 5.9.1 QCAP\_REGISTER\_RECORD\_DONE\_CALLBACK

## Introduction

The user can register a *PF\_RECORD\_DONE\_CALLBACK* function to get notification when the recording process is completed. When **QCAP\_STOP\_RECORD()** have done video files processing, then user-defined callback function will be called.

When calling **QCAP\_STOP\_RECORD()** asynchronously, or when some video format (e.g. **MP4**) may take a while to complete, this callback can help to know when the video file is ready.



This callback function need to be registered before recording starts.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<i>PF_RECORD_DONE_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# *PF\_RECORD\_DONE\_CALLBACK*

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
CHAR *	pszFilePathName	The video recorded filename
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example: Register a callback function for channel recording to notify completion*

```
QRETURN channel_record_done( PVOID pDevice,
                             UINT iRecNum,
                             CHAR * pszFilePathName,
                             PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_RECORD_DONE_CALLBACK pCB = channel_record_done;

    QCAP_REGISTER_RECORD_DONE_CALLBACK( pDevice, pCB, pUserData );
}
```

# 5.9.1 QCAP\_REGISTER\_RECORD\_FAIL\_CALLBACK

## Introduction

The user can register a **PF\_RECORD\_DONE\_CALLBACK** function to get notification when the recording process is fail for some reason. When an error occurred during recording process, then user-defined callback function will be called.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<b>PF_RECORD_FAIL_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_RECORD\_FAIL\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
CHAR *	pszFilePathName	The video recorded filename
QRESULT	nErrorStatus	IN
the error status of record fail	PVOID	pUserData

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example: Register a callback function for channel recording to notify completion*

```
QRETURN channel_record_fail( PVOID pDevice,
                             UINT iRecNum,
                             CHAR * pszFilePathName,
                             QRESULT nErrorStatus,
                             PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_RECORD_FAIL_CALLBACK pCB = channel_record_fail;

    QCAP_REGISTER_RECORD_FAIL_CALLBACK( pDevice, pCB, pUserData );
}
```



# 5.9.2 QCAP\_REGISTER\_VIDEO\_RECORD\_CALLBACK

## Introduction

The user can register a *PF\_VIDEO\_RECORD\_CALLBACK* function to process the video compressed data for every frame in a recording. When each video frame is recorded, then user-defined callback function will be called.



This callback function need to be registered before recording starts.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<i>PF_VIDEO_RECORD_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# *PF\_VIDEO\_RECORD\_CALLBACK*

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
BOOL	blsKeyFrame	Specify the input source's frame is keyframe or not
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The stream buffer of this frame will be dropped and will not be saved to the file

## Examples

*Example: Register a callback function for channel video frame in recording*

```
QRETURN channel_video_record( PVOID pDevice,
                              UINT iRecNum,
                              double dSampleTime,
                              BYTE * pStreamBuffer,
                              ULONG nStreamBufferLen,
                              BOOL bIsKeyFrame,
                              PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_RECORD_CALLBACK pCB = channel_video_record;

    QCAP_REGISTER_VIDEO_RECORD_CALLBACK( pDevice, pCB, pUserData );
}
```

# 5.9.3 QCAP\_REGISTER\_AUDIO\_RECORD\_CALLBACK

## Introduction

The user can register a *PF\_AUDIO\_RECORD\_CALLBACK* function to process the **PCM / AAC** audio data buffer for every frame in a recording. When each audio frame is recorded, then user-defined callback function will be called.



This callback function need to be registered before recording starts.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<i>PF_AUDIO_RECORD_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# *PF\_AUDIO\_RECORD\_CALLBACK*

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The stream buffer of this frame will be dropped and will not be saved to the file

## Examples

*Example: Register a callback function for channel audio frame in recording*

```
QRETURN channel_audio_record( PVOID pDevice,
                               UINT iRecNum,
                               double dSampleTime,
                               BYTE * pStreamBuffer,
                               ULONG nStreamBufferLen,
                               PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_AUDIO_RECORD_CALLBACK pCB = channel_audio_record;

    QCAP_REGISTER_AUDIO_RECORD_CALLBACK( pDevice, pCB, pUserData );
}
```

## 5.9.4 QCAP\_REGISTER\_MEDIA\_RECORD\_CALLBACK

### Introduction

The user can register a **PF\_MEDIA\_RECORD\_CALLBACK** function to process the media stream content (both audio/video) for every frame in a recording. Since media record buffer is used for streaming purpose, so it only supports for **TS** and **FLV** video format. When each streaming content is ready, then user-defined callback function will be called.



This callback function need to be registered before recording starts.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<b>PF_MEDIA_RECORD_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_MEDIA\_RECORD\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
PVOID	pDevice	Handle of the capture card object
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The streaming buffer of this frame will be dropped and will not be saved to the file

### Examples

*Example: Register a callback function for channel streaming frame in recording*

```
QRETURN channel_media_record( PVOID pDevice,
                              UINT iRecNum,
                              double dSampleTime,
                              BYTE * pStreamBuffer,
                              ULONG nStreamBufferLen,
                              PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_MEDIA_RECORD_CALLBACK pCB = channel_media_record;

    QCAP_REGISTER_MEDIA_RECORD_CALLBACK( pDevice, pCB, pUserData );
}
```

# 6 Synchronized Recording Function API

---

## Introduction

When a capture system has many sources, in a previous chapter the channel recording can record videos in each channel separately, and save independently along with their sample time. The main purpose for this chapter is to record live videos in each channel simultaneously and maintain their global sample time. Although each channel video saves to separately files, it is useful to check the video of scenes / environments in the synchronized timeline.



## 6.1 QCAP\_START\_SYNCHRONIZED\_RECORD

### Introduction

The user can use this function to start synchronized recording. At this function, QCAP will give unified sample time to start multiple recording. For example, QCAP will provide same start sample time for multiple recordings. The user can use *nFileArgs* parameter to set the total devices number to synchronized recording.



If user using this function to start synchronized recording of **share recording**, *pDevice* must be 0x01.#

### Parameters

type	parameter	I/O	descriptions
ULONG	nFileArgs	IN	Set the total devices number to sync record
CHAR *	pszFilePathName1	IN	1st device: Specify the file name for recording: <b>MP4,FLV,TS,ASF,SCF,WMV</b> <b>Note: Synchronized recording does not support AVI format.</b>
PVOID	pDevice1	IN	1st device: Handle of the capture 1st card object
UINT	iRecNum1	IN	1st device: Specify the recorder slot to start synchronized recording
CHAR *	pszFilePathName2	IN	2nd device: Specify the file name for recording.
PVOID	pDevice2	IN	2nd device:Handle of the 2nd capture card object
UINT	iRecNum2	IN	2nd device:Specify the recorder slot to start synchronized recording
...	...	...	<b>Variable arguments</b> (to support for 3rd, 4th, 5th, 6th... devices etc.)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Synchronize recording for 4 channels uses the same time-stamp*

```
QCAP_START_SYNCHRONIZED_RECORD( 4, //total channels to sync record
    "SR_CH01.MP4", pDevices[ 0 ], 0,
    "SR_CH02.MP4", pDevices[ 1 ], 0,
    "SR_CH03.MP4", pDevices[ 2 ], 0,
    "SR_CH04.MP4", pDevices[ 3 ], 0 );
```



## 6.2 QCAP\_PAUSE\_SYNCHRONIZED\_RECORD

### Introduction

The user can use this function to pause synchronized recording for multiple channels. The user can use *nFileArgs* parameter to set the total devices number to synchronized recording.

### Parameters

type	parameter	I/O	descriptions
ULONG	nFileArgs	IN	Set the total devices number to sync record
PVOID	pDevice1	IN	Handle of the capture card object
UINT	iRecNum1	IN	Specify the recorder slot to pause synchronized recording
PVOID	pDevice2	IN	Handle of the capture card object
UINT	iRecNum2	IN	Specify the recorder slot to pause synchronized recording
...	...	...	Variable arguments (to support for 3rd, 4th, 5th, 6th... devices etc.)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Pause synchronized recording for 4 channels*

```
QCAP_PAUSE_SYNCHRONIZED_RECORD( 4, //total channels to sync record
                                pDevices[ 0 ], 0,
                                pDevices[ 1 ], 0,
                                pDevices[ 2 ], 0,
                                pDevices[ 3 ], 0 );
```

# 6.3 QCAP\_RESUME\_SYNCHRONIZED\_RECORD

## Introduction

The user can use this function to resume synchronized recording from previous pause status. The user can use *nFileArgs* parameter to set the total devices number to synchronized recording.

## Parameters

type	parameter	I/O	descriptions
ULONG	nFileArgs	IN	Set the total devices number to sync record
PVOID	pDevice1	IN	Handle of the capture card object
UINT	iRecNum1	IN	Specify the recorder slot to resume synchronized recording
PVOID	pDevice2	IN	Handle of the capture card object
UINT	iRecNum2	IN	Specify the recorder slot to resume synchronized recording
...	...	...	Variable arguments (to support for 3rd, 4th, 5th, 6th... devices etc.)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Resume synchronized recording for 4 channels*

```
QCAP_RESUME_SYNCHRONIZED_RECORD( 4, //total channels to sync record
                                pDevices[ 0 ], 0,
                                pDevices[ 1 ], 0,
                                pDevices[ 2 ], 0,
                                pDevices[ 3 ], 0 );
```

# 6.4 QCAP\_STOP\_SYNCHRONIZED\_RECORD

## Introduction

The user can use this function to stop synchronized recording.

This function is designed for both synchronous and asynchronous operations.

The user can use *nFileArgs* parameter to set the total devices number to synchronized recording.



For more detailed parameters descriptions, please refer to **QCAP\_STOP\_RECORD()**.

## Parameters

type	parameter	I/O	descriptions
BOOL	blsAsync	IN	Set the asynchronous operation flag
ULONG	nMilliseconds	IN	Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.
ULONG	nFileArgs	IN	Set the total devices number to sync record
PVOID	pDevice1	IN	Handle of the capture card object
UINT	iRecNum1	IN	Specify the recorder slot to stop synchronized recording
PVOID	pDevice2	IN	Handle of the capture card object
UINT	iRecNum2	IN	Specify the recorder slot to stop synchronized recording
...	...	...	<b>Variable arguments</b> (to support for 3rd, 4th, 5th, 6th... devices etc.)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Stop synchronized recording for 4 channels*

```
QCAP_STOP_SYNCHRONIZED_RECORD( TRUE, 0,
                                4, //total channels to sync record
                                pDevices[ 0 ], 0,
                                pDevices[ 1 ], 0,
                                pDevices[ 2 ], 0,
                                pDevices[ 3 ], 0 );
```

# 7 OSD Function API

## Introduction



An on-screen display (OSD) are control functions on a video screen that allows you to draw text fields, overlapped pictures, play sequences of images for animation, or put customer image buffer with blending or color key effect. For multiple object on the video can also control their output layer. The timing to the OSD output is important when capture device is in pipe-line processes such as recording or streaming...etc. A *nSequenceStyle* parameter is used to adjust the OSD output timing Before/After in different stages(it depends on the applications). For example, Draw OSD after preview callback and before Recording application starts.

Here are the list of the OSD Sequence Styles:

Application	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
GENERAL DEVICE	<b>FOREMOST</b>	PREVIEW CALLBACK	<b>BEFORE_ENCODE</b>	RECORDING (encode)	<b>AFTERMOST</b>
SHARE RECORDING	<b>FOREMOST</b>	RECORDING (encode)	<b>AFTERMOST</b>	—	—
BROADCAST SERVER	<b>FOREMOST</b>	STREAMING (encode)	<b>AFTERMOST</b>	—	—
BROADCAST CLIENT	PLAYBACK (decode)	<b>FOREMOST</b>	DECODER CALLBACK	<b>AFTERMOST</b>	—
VIRTUAL CAMERA	<b>FOREMOST</b>	<b>AFTERMOST</b>	SHARING	—	—
FILE PLAYBACK	PLAYBACK (decode)	<b>FOREMOST</b>	DECODER CALLBACK	<b>AFTERMOST</b>	—



1. Each OSD layer can set only 1 object, the later added OSD object will remove previous one.
2. To remove an OSD object from OSD layer, simply call **QCAP\_SET\_OSD\_BUFFER()** with buffer set to **NULL**.



## 7.1 QCAP\_SET\_OSD\_TEXT

## 7.2 QCAP\_SET\_OSD\_TEXT\_EX

## 7.3 QCAP\_SET\_OSD\_TEXT\_W

## 7.4 QCAP\_SET\_OSD\_TEXT\_EX\_W

### Introduction

The user can use this function to create a text field objects used for on-screen display. If the user set width/height parameter to 0, then QCAP will calculate actual width and height automatically. You also can adjust the OSD output layer by *iOsdNum* on video-streams.

For blending and transparent effects, here are some notices:

- The Color space of *FontColor* is **ARGB**, don't forget to set its alpha value.
- The Color space of *BackgroundColor* is **ARGB**, don't forget to set its alpha value.
- The *Transparent* parameter is a global alpha value for all colors. Set 255 means no transparent.

By calling **QCAP\_SET\_OSD\_TEXT\_[EX]\\_W()**, this function can support Wide-String in parameters: *pwszString*, *pszFontFamilyName*.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output. Set 0 to auto calculate width.
INT	h	IN	Specify the height of OSD output. Set 0 to auto calculate height.
CHAR * <a href="#">WSTRING</a>	pszString <a href="#">pwszString</a>	IN	Specify to display the text of OSD output <a href="#">Support wide character string</a>
CHAR * <a href="#">WSTRING</a>	pszFontFamilyName <a href="#">pwszFontFamilyName</a>	IN	Specify the font name used to display the text of OSD output <a href="#">Support wide character string</a>
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output: QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT

ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
DWORD	dwFontColor	IN	Specify the font color <b>ARGB</b> used to display the text of OSD output
DWORD	dwBackgroundColor	IN	Specify the background color <b>ARGB</b> used to display the text of OSD output
DWORD	dwBorderColor	IN	Specify the border color of the text <b>Only in QCAP_SET_OSD_TEXT_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width in pixel, set 0 to disable border. <b>Only in QCAP_SET_OSD_TEXT_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
INT	nTextStartPosX	IN	<b>default 0</b> Specify the text position X of the upper-left corner of OSD text
INT	nTextStartPosY	IN	<b>default 0</b> Specify the text position Y of the upper-left corner of OSD text
ULONG	nStringAlignmentStyle	IN	<b>default QCAP_STRING_ALIGNMENT_STYLE_LEFT</b> The alignment styles are: QCAP_STRING_ALIGNMENT_STYLE_LEFT QCAP_STRING_ALIGNMENT_STYLE_NEAR QCAP_STRING_ALIGNMENT_STYLE_CENTER QCAP_STRING_ALIGNMENT_STYLE_RIGHT QCAP_STRING_ALIGNMENT_STYLE_FAR <b>Only in QCAP_SET_OSD_TEXT_EX()</b>
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Put a string "CH01" on the top of preview video*

```
QCAP_SET_OSD_TEXT( hHandle,  
    0,  
    0, 0, 0, 0,  
    "CH01", "Arial",  
    QCAP_FONT_STYLE_BOLD, 12,  
    0xFFFF0000,  
    0xFFFFFFFF,  
    128, 0, 0,  
    QCAP_SEQUENCE_STYLE_FOREMOST );  
  
QCAP_SET_OSD_TEXT_EX( hHandle,  
    0,  
    0, 0, 0, 0,  
    "CH01", "Arial",  
    QCAP_FONT_STYLE_BOLD, 12,  
    0xFFFF0000,  
    0xFFFFFFFF,  
    0,0,  
    128, 0, 0,  
    QCAP_STRING_ALIGNMENT_STYLE_LEFT,  
    QCAP_SEQUENCE_STYLE_FOREMOST );
```



# 7.5 QCAP\_GET\_OSD\_TEXT\_BOUNDARY

# 7.6 QCAP\_GET\_OSD\_TEXT\_BOUNDARY\_W

## Introduction

The user can use this function to get a size of OSD string. For example, the user can use boundary information to set OSD string location on a display window.

By calling **QCAP\_GET\_OSD\_TEXT\_BOUNDARY\_W()**, this function can support Wide-String in parameters: *pwszString*, *pszFontFamilyName*.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
CHAR * WSTRING	pszString pwszString	IN	Specify to display the text of OSD output Support wide character string
CHAR * WSTRING	pszFontFamilyName pwszFontFamilyName	IN	Specify the font name used to display the text of OSD output Support wide character string
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output: QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
ULONG *	pBoundaryWidth	OUT	Pointer to the boundary width
ULONG *	pBoundaryHeight	OUT	Pointer to the boundary height

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the boundary width/height of the OSD text*

```
QCAP_GET_OSD_TEXT_BOUNDARY( hHandle,
                             0,
                             "CH01", "Arial",
                             QCAP_FONT_STYLE_BOLD, 12,
                             &BoundaryWidth,
                             &BoundaryHeight );
```

## 7.7 QCAP\_SET\_OSD\_PICTURE

### Introduction

This OSD function displays one or more **BMP/JPG/PNG/GIF/EDL.INI** on top of a video stream. The user can call this function to place a picture from an image file for on-screen display. If the user set width/height parameter to 0, then QCAP will calculate actual width and height automatically. The user also can adjust the OSD output layer by *iOsdNum* on video-streams.

If user wants to allow various images (frames) to be painted with time delays by OSD picture, here are 2 ways to do so:

- use **GIF** animated image format as input
- use **EDL.INI** to describe the number of frames to display in succession.

The **EDL.INI** script describes how the image sequences will be played one-by-one with a time delay, and how many times to replay the animation. Here is the example of the script:

```
[OUTLINE]
LOOP=0           ; 0 = INFINITE, > 1 = TIMES to loop-play
SPEED=1          ; IN UNITS / FRAME is images playing speed
LENGTH=5        ; FILE COUNTS is total images count
FILENAME.0=TEST0.PNG
FILENAME.1=TEST1.PNG
FILENAME.2=TEST2.PNG
FILENAME.3=TEST3.PNG
FILENAME.4=TEST4.PNG
```

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set to -1 to use original picture width as default.
INT	h	IN	Specify the height of OSD output. Set to -1 to use original picture height as default.
CHAR *	pszFilePathName	IN	Specify the image file name to display on OSD. Supported extension are <b>JPG, PNG, BMP</b> (24/32 bit) Supported animation by <b>GIF,EDL.INI</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	

			<b>default 0.0</b>
			Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example 1: Place a half-transparent PNG on the top of captured video stream*

```
QCAP_SET_OSD_PICTURE( hHandle,
    0,
    0, 0, 0, 0,
    "C:/SAMPLE.PNG",
    128,
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

*Example 2: Place a series of PNG images by EDL.INI to animate pictures on OSD*

```
QCAP_SET_OSD_PICTURE( hHandle,
    0,
    0, 0, 0, 0,
    "C:/EDL.INI",
    128,
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

# 7.8 QCAP\_SET\_OSD\_BUFFER

# 7.8 QCAP\_SET\_OSD\_BUFFER\_EX

## Introduction

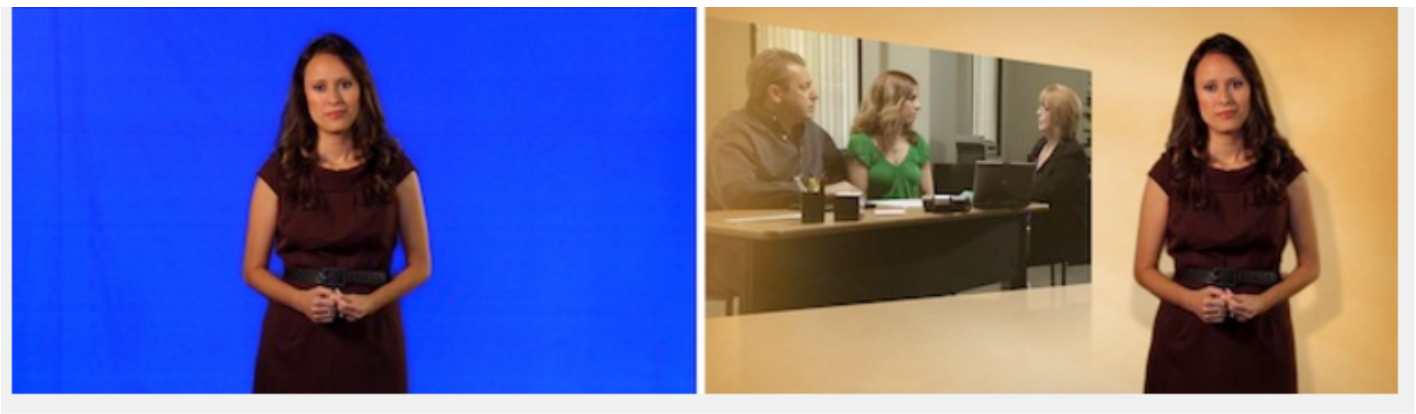
The user can use this function to create a framebuffer object used for on-screen display. It can directly use a framebuffer whose entire pixel data is stored in available color space, and display it on the screen. You also can adjust the width and height parameters to scale the OSD output.

The *pMaskBuffer* parameter is to provide a mask bitmap area which width/height is same as OSD buffer, each pixel use a byte (0/1) to represent a mask bit - when a value is 1 the pixel won't be shown. It will be used to mask the buffer when OSD output.

The *dwKeyColor* parameter is used to select color key mode: Green / Blue Screen mode and RGB mode. In **ARGB** color type the bytes arrangement in memory is **[R][G][B][A]**, and in an **ULONG** variable: **0xAABBGGRR** (A is for alpha-blending). For Blue screen the *dwKeyColor* can set to **0x000000FF** as shown below:

- 1. 0xFFFFFFFF (NO COLORKEY)
- 2. 0x00FF0000 (MASK **BLUE**)
- 3. 0x0000FF00 (MASK **GREEN**)

To remove an OSD object from OSD layer, simply call **QCAP\_SET\_OSD\_BUFFER()** with buffer set to **NULL**.



ColorKey: Alpha=255, Red=0, Green=0, Blue=255, Threshold=16



The performance of **RGB** type is faster than **BGR** in our alpha blending algorithm.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object

UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output
INT	h	IN	Specify the height of OSD output
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pFrameBuffer	IN	Specify a raw data of framebuffer
ULONG	nFrameWidth	IN	Specify the width of framebuffer
ULONG	nFrameHeight	IN	Specify the height of framebuffer
ULONG	nFramePitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate it by width and color space format.
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_BUFFER_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_BUFFER_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SET_OSD_BUFFER_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SET_OSD_BUFFER_EX()</b>
DWORD	dwBorderColor	IN	Specify the border color <b>Only in QCAP_SET_OSD_BUFFER_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width <b>Only in QCAP_SET_OSD_BUFFER_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	<b>default 0xFFFFFFFF</b> Specify the key color of OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY) 2. 0x00FF0000 (MASK BLUE) 3. 0x0000FF00 (MASK GREEN)
ULONG	nKeyColorThreshold	IN	<b>default 25</b> Specify the threshold of key color, the range from 0 to 128
ULONG	nKeyColorBlurLevel	IN	<b>default 2</b> Specify the blur level, range 0-2
BOOL	bKeyColorSpillSuppress	IN	<b>default TRUE</b> Specify the color spill suppress value

ULONG	nKeyColorSpillSuppressThreshold	IN	<b>default 22</b> Specify the threshold value of color spill suppress
BYTE *	pMaskBuffer	IN	<b>default NULL</b> Specify a mask bitmap in bytes for OSD buffer, value 1 is masked.
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example 1: Place a picture directly from a framebuffer on the video display*

```
QCAP_SET_OSD_BUFFER( hHandle,  
    0,  
    0, 0,  
    1920, 1080,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    pFramebuffer,  
    800, 600, 0,  
    128,  
    0xFFFFFFFF,  
    25,  
    2,  
    NULL,  
    QCAP_SEQUENCE_STYLE_FOREMOST );  
  
QCAP_SET_OSD_BUFFER_EX( hHandle,  
    0,  
    0, 0,  
    1920, 1080,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    pFramebuffer,  
    800, 600, 0,  
    0,0,0,0, 0, 0,  
    128,  
    0xFFFFFFFF,  
    25, 2, FALSE,  
    NULL,  
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

*Example 2: To remove an OSD object form OSD layer 0*

```
QCAP_SET_OSD_BUFFER( hHandle,  
    0,  
    0, 0,  
    0, 0,  
    0,  
    NULL,  
    800, 600, 0,  
    0,  
    0xFFFFFFFF,  
    0,  
    0,  
    NULL,  
    0 );
```

## 7.9 QCAP\_MOVE\_OSD\_OBJECT

# Introduction

The user can use this function to move the OSD object around the video window. It is useful to scroll the text string or picture on the video display window.



For more detailed parameters descriptions, please refer to **QCAP\_MOVE\_OSD\_OBJECT()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To scroll the OSD object from left to right*

```
for ( int i = 0; i< 1920; i++ )
{
    QCAP_MOVE_OSD_OBJECT( 0,
                           i, 0,
                           QCAP_SEQUENCE_STYLE_FOREMOST );
}
```



# 8 Clone Function API

---

## Introduction



This chapter can help the user to clone the preview video to multiple windows from the same capture source. The clone preview windows is another separate windows from the main window. By move the windows position, it can overlapped with the main windows like a Picture-In-Picture (PIP).+

## 8.1 QCAP\_CREATE\_CLONE

### Introduction

The user can use this function to create a clone device from current capture device object. The cloning device can then render preview in another window. The clone preview windows size will decide the clone video size. The user can use property functions to access the clone device, such as **QCAP\_SET\_VIDEO\_INPUT()**. QCAP auto-mapping those properties to its physical device object. However, every clone device can be accessed independently.

The Main device and the clone device typical work flow are:

#	Main Device	Clone Device	Function calls
1	Create Main Device		QCAP_CREATE( &pDevice )
2	Start Main Capturing		QCAP_RUN( pDevice )
3		<i>Create Clone Device</i>	<b>QCAP_CREATE_CLONE(pDevice, &amp;pClone)</b>
4		<i>Start Clone Preview</i>	<b>QCAP_RUN( pClone )</b>
5		<i>Access Clone Property</i>	<b>QCAP_GET_VOLUME() / QCAP_SET_VOLUME()</b>
6		<i>Stop Clone Preview</i>	<b>QCAP_STOP( pClone )</b>
7		<i>Delete Clone Device</i>	<b>QCAP_DESTROY( pClone )</b>
8	Stop Main Capturing		QCAP_STOP( pDevice )
9	Delete Main Device		QCAP_DESTROY( pDevice )

The **ThumbDraw** function can reduce the render loading of video display, especially when display window size is small than the input video size. We suggest turning on **bThumbDraw** in order to obtain the best rendering performance and save CPU loadings.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDevice	IN	Handle of the capture card object
HWND	hAttachedWindow	IN	Handle of the window to show the preview clone video
PVOID *	ppCloneDevice	OUT	Handle of the clone capture device
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the Maintain aspect ratio

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : create clone device from pDevice and preview it in the second window*

```
PVOID pDevice; //Main Device
PVOID pClone;  //Clone device

HWND Main_hWnd;    //Main Preview window
HWND Clone_hWnd;   //Clone Preview window

QCAP_CREATE( "QP0203 PCI", 0, Main_hWnd, &pDevice, true, true);

QCAP_RUN( pDevice );

    QCAP_CREATE_CLONE( pDevice, Clone_hWnd, &pClone );

    QCAP_RUN( pClone );

    //Clone Preview starting

    QCAP_SET_AUDIO_VOLUME( pClone, 80 );

    //.....

    QCAP_STOP( pClone );

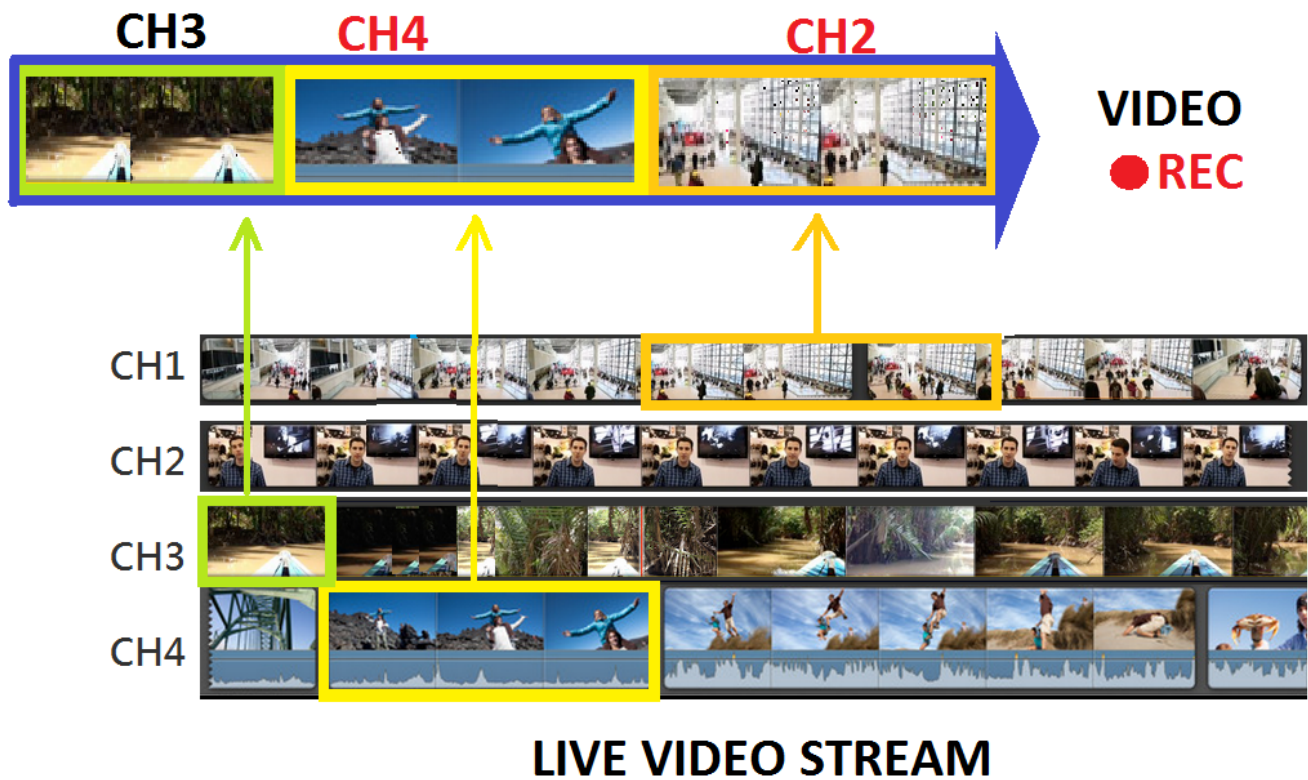
    QCAP_DESTROY( pClone );

QCAP_STOP( pDevice );

QCAP_DESTROY( pDevice );
```

# 9 Share Recording Function API

## Introduction



This chapter provides *share recording* functions that can help you to create one customized recording file easily and are different from *channel recording*. The **share recording** means you can switch to any live channel to be recorded in any recording time. The software developer can be free to combine any audio and video frame buffers into one recording engine.

*For a single recording tasks, the differences between APIs:*

- **Channel Recording:** You can record one channel at a time, save into one video file.
- **Synchronize Recording:** You can record many channels in a synchronized time, save into many video files.
- **Share Recording:** You can record one channel a time, you can choose which channel to record in any time, then save into one video file.
- **Broadcast Client Recording:** You can record one server session at a time, save into one video file.

The share recording engine will hide all complex operations under the hood. To switching multiple channel encoding and save to one video file, you just need to decide which channel's frame buffer to push into the recording engine. In result, you will have only one video file, with video from each channel in different timeline, as shown in figure:

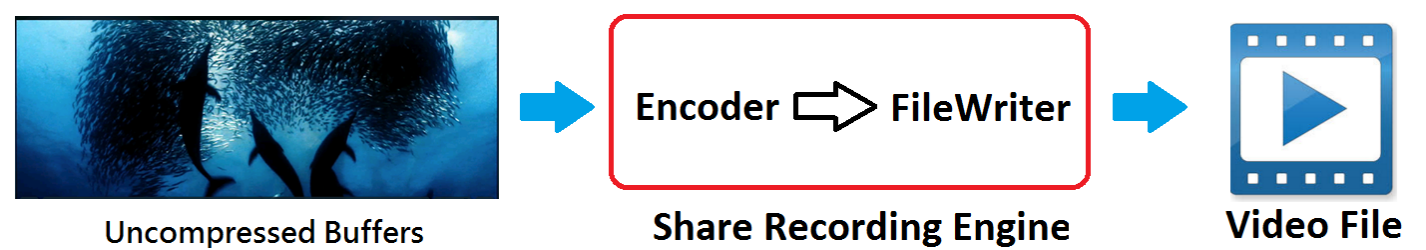
# 9.1 Share Record Major Functions

## Introduction

This section provides *share recording* major functions that can start a shared record from created captured objects, to pause/resume share recording, and to stop share recording when your task is done. Because share recording is using existing capturing object already created, so there is no create/destroy functions in share recording.

### 9.1.1 QCAP\_START\_SHARE\_RECORD

#### Introduction



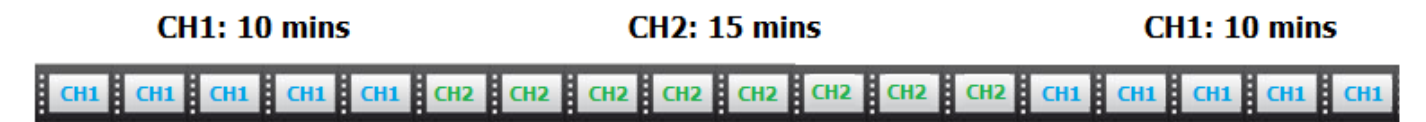
The share recording API is to provide a simple mechanism for a user to record captured streams from different input devices at a time.

In **Channel Recording / Synchronized Recording**, a user doesn't have to push any audio/video frame buffer to record engine, it's automatically done no matter hardware or software encoder were used.

In **Share Recording** user have to decide which channel's frame buffer to push into share recording engine so that channel selection in recording time can be archived. The push audio/video frame buffer operations usually be done at capture stream callbacks:

#	Push video frames to Share Recording engine	Push audio frames to Share Recording engine
1	<b>Software Encoder</b> QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK	<b>Software Encoder</b> QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK
2	<b>Hardware Encoder</b> QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK	

Just like channel recording, the share recording properties for audio/video can be manually performed by set record property functions. For example, a user can select the encoder type, video recording bit rate and quality, video file format, and adjust the audio/video synchronize time... etc. There are also share recording callbacks when the recording completion or when audio/video data available.



For instance, we can record the first 10 min video from CH1 and the 15 mins from CH2, and the rest video from CH1, into one **H.264** MP4 file.

## Practical Share Recording flow

Steps	Programming Step	API Function calls
1	Set Video Record Property	<b>QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( idx, ... )</b>
2	Set Audio Record Property	<b>QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( idx, ... )</b>
3	Start Share Recording	<b>QCAP_START_SHARE_RECORD( idx, ... )</b>
4	Push the i-th Video Frame Buffer into Recorder	<b>QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( idx, ... )</b> or <b>QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER( idx, ... )</b>
5	Push the i-th Audio Frame Buffer into Recorder	<b>QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( idx, ... )</b>
6	To continue share recording, <b>Go to Step 4</b>	
7	Stop Share Recording	<b>QCAP_STOP_SHARE_RECORD( idx, ... )</b>



These property setting functions:

**QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY()**,

**QCAP\_SET\_AUDIO\_SHARE\_RECORD\_PROPERTY()** must be called before **QCAP\_START\_SHARE\_RECORD()** calls.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters  start from 0 Max RecNum is 0-63
CHAR *	pszFilePathName	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3</b>
DWORD	dwFlags	IN	<b>default QCAP_RECORD_FLAG_FULL</b> Specify the action flag of share recording, can be combinations of the following value: QCAP_RECORD_FLAG_FULL QCAP_RECORD_FLAG_FILE QCAP_RECORD_FLAG_ENCODE QCAP_RECORD_FLAG_DISPLAY QCAP_RECORD_FLAG_DECODE QCAP_RECORD_FLAG_VIDEO_ONLY QCAP_RECORD_FLAG_AUDIO_ONLY QCAP_RECORD_FLAG_VIDEO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_AUDIO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_IGNORE_FORMAT_CHANGED QCAP_RECORD_FLAG_SYNCHRONIZED_RECORD QCAP_RECORD_FLAG_VIDEO_USE_MEDIA_TIMER. QCAP_RECORD_FLAG_AUDIO_USE_MEDIA_TIMER.
double	dVideoDelayTime	IN	<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time
double	dSegmentDurationTime	IN	<b>default 0.0</b> Specify the video segment duration time to split in recording file (in seconds)
ULONG	nSegmentDurationSizeKB	IN	

**default 0**

Specify the video segment duration to split into recording file (in Kilo-Bytes)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

#### *Example 1: Start a shared recording by means of software encoding*

```
PVOID Selected_Device = NULL;

QRETURN video_preview_callback( PVOID pDevice,
                                double dSampleTime,
                                BYTE * pFrameBuffer,
                                ULONG nFrameBufferLen,
                                PVOID pUserData )
{
    if( pDevice == Selected_Device )
        QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, MAKEFOURCC('Y', 'V', '1', '2'), W[ ch ], H[ ch ],
        pFrameBuffer, nFrameBufferLen );

    return QCAP_RT_OK;
}

QRETURN audio_preview_callback( PVOID pDevice,
                                double dSampleTime,
                                BYTE * pFrameBuffer,
                                ULONG nFrameBufferLen,
                                PVOID pUserData )
{
    if( pDevice == Selected_Device )
        QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, pFrameBuffer, nFrameBufferLen );

    return QCAP_RT_OK;
}

void software_share_recording()
{
    QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( 0,
        QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
        QCAP_ENCODER_FORMAT_H264,
        QCAP_COLORSPACE_TYEP_YV12,
        1920, 1080, 30,
        QCAP_RECORD_MODE_CBR,
        8000,
        12 * 1024 * 1024,
        30,
        0, 0, 0,
        1 );
}
```

```

QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( 0,
                                       QCAP_ENCODER_TYPE_SOFTWARE,
                                       QCAP_ENCODER_FORMAT_AAC,
                                       2, 16, 48000 );

QCAP_START_SHARE_RECORD( 0,
                        "D:/SHARE_RECORD_$Y_$M_$D_$h_$m_$s_$i_MP4",
                        QCAP_RECORD_FLAG_FULL );
}

//To select which channel to record in share recording at run-time

void select_channel_in_share_recording( PVOID pDevice )
{
    Selected_Device = pDevice;
}

```

*Example 2: Start a shared recording by means of hardware encoding (with a hardware compression card only)*

```

PVOID Selected_Device = NULL;

QRETURN video_hardware_encoder_callback( PVOID pDevice,
                                         UINT iRecNum,
                                         double dSampleTime,
                                         BYTE * pStreamBuffer,
                                         ULONG nStreamBufferLen,
                                         BOOL bIsKeyFrame,
                                         PVOID pUserData )
{
    //...
    if( pDevice == Selected_Device )
        QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER( 0,
                                                         pStreamBuffer,
                                                         nStreamBufferLen,
                                                         bIsKeyFrame,
                                                         dSampleTime);

    //...
    return QCAP_RT_OK;
}

QRETURN audio_preview_callback( PVOID pDevice,
                               double dSampleTime,
                               BYTE * pFrameBuffer,
                               ULONG nFrameBufferLen,
                               PVOID pUserData )
{

```



```

//...
if( pDevice == Selected_Device )
    QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0,
                                                        pFrameBuffer,
                                                        nFrameBufferLen );

//...
return QCAP_RT_OK;
}

void hardware_share_recording()
{

    QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( 0,
                                           QCAP_ENCODER_TYPE_HARDWARE,
                                           QCAP_ENCODER_FORMAT_H264,
                                           QCAP_COLORSPACE_TYEP_YV12,
                                           1920, 1080, 30,
                                           QCAP_RECORD_MODE_CBR,
                                           8000,
                                           12 * 1024 * 1024,
                                           30,
                                           0, 0, 0,
                                           1 );

    QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( 0,
                                           QCAP_ENCODER_TYPE_SOFTWARE,
                                           QCAP_ENCODER_FORMAT_AAC,
                                           2, 16, 48000 );

    QCAP_START_SHARE_RECORD( 0,
                             "D:/SHARE_RECORD_$Y_$M_$D_$h_$m_$s_$i_MP4",
                             QCAP_RECORD_FLAG_FULL );

}

//To select which channel to record in share recording at run-time

void select_channel_in_share_recording( PVOID pDevice )
{
    Selected_Device = pDevice;
}

```

## 9.1.2 QCAP\_START\_TIMESHIFT\_SHARE\_RECORD

### Introduction

The user can use this function to start time-shifting share recording. That means a user can continue share recording one video while playing back the same video recording file. In another word, a user can playback the video in share recording on-the-fly, just like time-shifted.

The *ppPhysicalFileWriter* parameter returns a file handle in which a time-shifting recording is in progress. This file handle could be used by **QCAP\_OPEN\_TIMESHIFT\_FILE\_EX()** and other playback functions.

This time-shifted functionality current supported **MP4** format only.

For more detailed parameters descriptions, please refer to:

\* **QCAP\_START\_TIMESHIFT\_RECORD()**

\* **QCAP\_OPEN\_TIMESHIFT\_FILE\_EX()**



These property setting functions:

**QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY()**.

**QCAP\_SET\_AUDIO\_SHARE\_RECORD\_PROPERTY()** must be called before **QCAP\_START\_SHARE\_RECORD()** calls.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
CHAR *	pszFilePathName	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3</b>
PVOID *	ppPhysicalFileWriter	OUT	Handle of Physical File Writer object
DWORD	dwFlags	IN	<b>default QCAP_RECORD_FLAG_FULL</b> Specify the action flag of share recording, can be combinations of the following value: QCAP_RECORD_FLAG_FULL QCAP_RECORD_FLAG_FILE QCAP_RECORD_FLAG_ENCODE QCAP_RECORD_FLAG_DISPLAY QCAP_RECORD_FLAG_DECODE QCAP_RECORD_FLAG_VIDEO_ONLY QCAP_RECORD_FLAG_AUDIO_ONLY QCAP_RECORD_FLAG_VIDEO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_AUDIO_USE_IDEAL_TIMESTAMP QCAP_RECORD_FLAG_IGNORE_FORMAT_CHANGED QCAP_RECORD_FLAG_SYNCHRONIZED_RECORD QCAP_RECORD_FLAG_VIDEO_USE_MEDIA_TIMER. QCAP_RECORD_FLAG_AUDIO_USE_MEDIA_TIMER.
double	dVideoDelayTime	IN	<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

Example : Start a time-shifted share recording for **MP4**, and playback the video (still in recording)

```
QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( 0,
                                       QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                       QCAP_ENCODER_FORMAT_H264,
                                       QCAP_COLORSPACE_TYEP_YV12,
                                       1920, 1080, 30,
                                       QCAP_RECORD_MODE_CBR,
                                       8000,
                                       12 * 1024 * 1024,
                                       30,
                                       0, 0, 0, 1 );

QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( 0,
                                       QCAP_ENCODER_TYPE_SOFTWARE,
                                       QCAP_ENCODER_FORMAT_AAC,
                                       2, 16, 48000 );

PVOID pPhysicalFileWriter = NULL;

PVOID pFile = NULL;

QCAP_START_TIMESHIFT_SHARE_RECORD( 0,
                                    "D:/SHARE_RECORD_01.MP4",
                                    &pPhysicalFileWriter );

QCAP_OPEN_TIMESHIFT_FILE_EX( pPhysicalFileWriter,
                             &pFile, QCAP_DECODER_TYPE_SOFTWARE, &nVideoFormat,
                             &nVideoWidth, &nVideoHeight, &dVideoFrameRate,
                             &nAudioFormat, &nAudioChannels,
                             &nAudioBitsPerSample, &nAudioSampleFrequency,
                             &dFileTotalDuationTimes,
                             &nFileTotalVideoFrames, &nFileTotalAudioFrames,
                             &nFileTotalMetadataFrames,
                             hWnd, 1 );

QCAP_PLAY_FILE( pFile ); //playback the video (still in time-shifted share recording)
```

### 9.1.3 QCAP\_PAUSE\_SHARE\_RECORD

**Introduction**

The user can use this function to pause share recording.  
If a user starts many recording in multiple recorder slots, the *iRecNum* parameter can use to choose which recorder slot to pause. To resume share recording for a recorder slot please call **QCAP\_RESUME\_SHARE\_RECORD()**.

**Parameters**

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 9.1.4 QCAP\_RESUME\_SHARE\_RECORD

**Introduction**

The user can use this function to resume a previously paused share recording.  
If a user starts many recording in multiple recorder slots, the *iRecNum* parameter can use to choose which recorder slot to resume.

**Parameters**

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : Pause/Resume a previously paused share recording in recorder slot 0*

```
QCAP_PAUSE_SHARE_RECORD( 0 );

QCAP_RESUME_SHARE_RECORD( 0 );
```

# 9.1.5 QCAP\_STOP\_SHARE\_RECORD

## Introduction

The user can use this function to stop share recording from a capture device.  
If the user starts many share recording in multiple recorder slots, the *iRecNum* parameter can use to choose which recorder slot to stop. For multiple recorder slot in a process, a user need to call **QCAP\_STOP\_SHARE\_RECORD()** for each recorder slot in order to stop all share recordings.

This function is designed for both synchronous and asynchronous operations.



For more detailed parameters descriptions, please refer to **QCAP\_STOP\_RECORD()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Stop a share recording at record\_index 0 & 1*

```
QCAP_STOP_SHARE_RECORD( 0 );  
  
QCAP_STOP_SHARE_RECORD( 1 );
```

# 9.1.6 QCAP\_GET\_SHARE\_RECORD\_STATUS

## Introduction

The user can use this function to monitor the share recorder's resources. For example, to see if a share record *RecNum* index is available to start a new share record.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BOOL	pIsValid	OUT	Pointer to the availability of this <i>iRecNum</i> .

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : See if share record number 0 is in-used or not.*

```
QCAP_GET_SHARE_RECORD_STATUS( 0, &pIsValid );
```

# 9.2 Share Record Property Functions

## Introduction

Before you want to create a video recording file, you will want to apply properties/setting so that the recording functions will output the effect you want. Like *Channel Recording*, the functions to access software / hardware encoder properties for **Share Recording** were designed in **two sets of functions**.

To access the audio/video share recording properties functions:

mode	functions
For <b>Software Encoder</b>	
Set encoder properties	QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY()
	QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()
	QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY()
	QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY_EX()
Get encoder properties	QCAP_SET_VIDEO_SHARE_RECORD_DYNAMIC_PROPERTY_EX()
	QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY()
	QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()
	QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY()
	QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY_EX()
	QCAP_GET_VIDEO_SHARE_RECORD_DYNAMIC_PROPERTY_EX()
For <b>Hardware Encoder</b> - please refer to <a href="#">Chapter 05 Recording Functions API</a>	
Set encoder properties	QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY()
	QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()
Get encoder properties	QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY()
	QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY_EX()

## 9.2.1 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY

## 9.2.2 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to set share video recording parameters, such as *encoder type*, *video recording bit rate*, and *quality*, **H.264** *encoder setting*, *Profile*, *Level*, *Entropy*, *Complexity*, and *B-Frames*...etc.

The property setting functions **QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY()**, **QCAP\_SET\_AUDIO\_SHARE\_RECORD\_PROPERTY()** must be called in prior to **QCAP\_START\_SHARE\_RECORD()** calls.

In multi-threaded application, QCAP will balance CPU loading when *bMultiThread* is true. When using **INTEL\_MEDIA\_SDK** as the encoder, the *MBBRC* and *ExtBRC* parameters could set to true to get better video quality in low bit rate mode on Intel® platform.

By using the attached window, QCAP can help you to real-time display the share recording's result. It is suggested to turn on *bThumbDraw* to obtain the best rendering performance.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_VIDEO\_RECORD\_PROPERTY\_EX()**.



In share recording, this function is For **Software encoder** only!

For hardware encoder must use **QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX()**

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE QCAP_ENCODER_TYPE_HARDWARE QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK QCAP_ENCODER_TYPE_AMD_STREAM QCAP_ENCODER_TYPE_NVIDIA_CUDA QCAP_ENCODER_TYPE_NVIDIA_NVENC <b>Note: QCAP_ENCODER_TYPE_HARDWARE Only for hardware encoder capture card.</b>
ULONG	nEncoderFormat	IN	Specify video encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12



			QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify encoder width.
ULONG	nHeight	IN	Specify encoder height.
double	dFrameRate	IN	Specify encoder frame rate.
ULONG	nRecordProfile	IN	<b>default QCAP_RECORD_PROFILE_BASELINE</b> Specify recording profile: QCAP_RECORD_PROFILE_BASELINE QCAP_RECORD_PROFILE_MAIN QCAP_RECORD_PROFILE_HIGH QCAP_RECORD_PROFILE_CONSTRAINED_BASELINE QCAP_RECORD_PROFILE_CONSTRAINED_HIGH <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nRecordLevel	IN	<b>default 41</b> Specify recording level QCAP_RECORD_LEVEL_1 QCAP_RECORD_LEVEL_1B QCAP_RECORD_LEVEL_11 QCAP_RECORD_LEVEL_12 QCAP_RECORD_LEVEL_13 QCAP_RECORD_LEVEL_2 QCAP_RECORD_LEVEL_21 QCAP_RECORD_LEVEL_22 QCAP_RECORD_LEVEL_3 QCAP_RECORD_LEVEL_31 QCAP_RECORD_LEVEL_32 QCAP_RECORD_LEVEL_4 QCAP_RECORD_LEVEL_41 QCAP_RECORD_LEVEL_42 QCAP_RECORD_LEVEL_50 QCAP_RECORD_LEVEL_51 QCAP_RECORD_LEVEL_52 QCAP_RECORD_LEVEL_60 QCAP_RECORD_LEVEL_61 QCAP_RECORD_LEVEL_62 <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nRecordEntropy	IN	<b>default QCAP_RECORD_ENTROPY_CAVLC</b> Specify recording entropy: QCAP_RECORD_ENTROPY_CAVLC QCAP_RECORD_ENTROPY_CABAC <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nRecordComplexity	IN	<b>default 0</b> Specify recording complexity: QCAP_RECORD_COMPLEXITY_0 (Best Speed) QCAP_RECORD_COMPLEXITY_1 QCAP_RECORD_COMPLEXITY_2 QCAP_RECORD_COMPLEXITY_3 QCAP_RECORD_COMPLEXITY_4 QCAP_RECORD_COMPLEXITY_5 QCAP_RECORD_COMPLEXITY_6 (Best Quality) <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR

			QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify recording quality, from 0-10000. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify recording bit rate. It is used for <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording GOP size, from 0-255
ULONG	nBFrames	IN	<b>default 0</b> Specify recording B-Frame <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL	bIsInterleaved	IN	<b>default FALSE</b> Enable/Disable the Interleaved <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nSlices	IN	<b>default 0</b> Specify recording slices, default 0 <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nLayers	IN	<b>default 0</b> Specify recording layers, default 0 <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nSceneCut	IN	<b>default 0</b> Specify recording Scene Cut, recommended value is 40. 0 is function turned off <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL	bMultiThread	IN	<b>default TRUE</b> Enable/Disable the multi-threaded CPU loading balance support <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL	bMBBRC	IN	<b>default FALSE</b> Enable/Disable the mbbrc <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL	bExtBRC	IN	<b>default FALSE</b> Enable/Disable the extbrc <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nMinQP	IN	<b>default 0</b> Specify the value of x264 Minimum quantizer settings <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nMaxQP	IN	<b>default 0</b> Specify the value of x264 Maximum quantizer settings <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nVBVMaxRate	IN	<b>default 0</b> Specify the value that x264 fills the buffer at (up to) the max rate <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nVBVBufSize	IN	<b>default 0</b> Specify the size that x264 fills the buffer <b>Only in QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG	nAspectRatioX	IN	Specify the aspect ratio X axis, 0 to turned off
ULONG	nAspectRatioY	IN	Specify the aspect ratio Y axis, 0 to turned off
HWND	hAttachedWindow	IN	<b>default NULL</b> Handle of the window to show the preview video
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the video properties before video share recording*

```
QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY(0,
                                     QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                     QCAP_ENCODER_FORMAT_H264,
                                     QCAP_COLORSPACE_TYEP_YV12,
                                     1280, 720, 60,
                                     QCAP_RECORD_MODE_CBR,
                                     8000,
                                     12*1024*1024,
                                     30,
                                     4, 3,
                                     hAttachedWindow,
                                     FALSE, FALSE );

QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX( 0,
                                     QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                     QCAP_ENCODER_FORMAT_H264,
                                     QCAP_COLORSPACE_TYEP_YUY2,
                                     1280, 720, 60,
                                     nRecordProfile,
                                     nRecordLevel,
                                     nRecordEntropy,
                                     QCAP_RECORD_COMPLEXITY_1,
                                     QCAP_RECORD_MODE_CBR,
                                     8000,
                                     12*1024*1024,
                                     30,
                                     0,
                                     FALSE,
                                     0,0,0,
                                     FALSE,
                                     0, 0,
                                     0, 0, 0, 0,
                                     4, 3,
                                     hAttachedWindow,
                                     FALSE, FALSE);
```

## 9.2.3 QCAP\_GET\_VIDEO\_SHARE\_RECORD\_PROPERTY

## 9.2.4 QCAP\_GET\_VIDEO\_SHARE\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to get video share recording parameters.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY\_EX()**.



In share recording, this function is For **Software encoder** only!

For hardware encoder must use **QCAP\_GET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY\_EX()**

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG *	pEncoderType	OUT	Pointer to the video encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the video encoder format
ULONG *	pColorSpaceType	OUT	Pointer to the video color space type
ULONG *	pWidth	OUT	Pointer to the video width
ULONG *	pHeight	OUT	Pointer to the video height
double *	pFrameRate	OUT	Pointer to the video frame rate
ULONG *	pRecordProfile	OUT	Pointer to the recording profile <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordLevel	OUT	Pointer to the recording level <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordEntropy	OUT	Pointer to the recording entropy <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordComplexity	OUT	Pointer to the recording complexity <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pRecordMode	OUT	Pointer to the recording mode
ULONG *	pQuality	OUT	Pointer to the quality
ULONG *	pBitRate	OUT	Pointer to the bit rate
ULONG *	pGOP	OUT	Pointer to the GOP size
ULONG *	pBFrames	OUT	Pointer to the B-Frames <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL *	pIsInterleaved	OUT	Pointer to the Interleaved <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pSlices	OUT	Pointer to the slices <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pLayers	OUT	Pointer to the layers <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pSceneCut	OUT	Pointer to the screen cut <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL *	pMultiThread	OUT	Pointer to the current multi-threaded CPU loading balance status <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL *	pMBBRC	OUT	Pointer to the current mbbrc status <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
BOOL *	pExtBRC	OUT	Pointer to the current extbrc status <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pMinQP	OUT	Pointer to the value of x264 Minimum quantizer settings <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pMaxQP	OUT	Pointer to the value of x264 Maximum quantizer settings <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pVBVMaxRate	OUT	

			Pointer to the value that x264 fills the buffer at (up to) the max rate <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pVBVBufSize	OUT	Pointer to the size that x264 fills the buffer <b>Only in QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX()</b>
ULONG *	pAspectRatioX	OUT	Pointer to the aspect ratio X axis
ULONG *	pAspectRatioY	OUT	Pointer to the aspect ratio Y axis
HWND *	pAttachedWindow	OUT	Pointer to the window to show the recording video
BOOL *	pThumbDraw	OUT	Pointer to the enable flag of thumb draw renderer
BOOL *	pMaintainAspectRatio	OUT	Pointer to the enable flag of maintaining aspect ratio

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the current video recording properties in share recording*

```

QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY( 0,
    &nEncoderType,
    &nEncoderFormat,
    &nColorSpaceType,
    &nWidth,
    &nHeight,
    &nFrameRate,
    &nRecordMode,
    &nQuality,
    &nBitRate,
    &nGOP,
    &nAspectRatioX,
    &nAspectRatioY,
    &hAttachedWindow,
    &nThumbDraw,
    &nMaintainAspectRatio );

QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX( 0,
    &nEncoderType,
    &nEncoderFormat,
    &nColorSpaceType,
    &nWidth,
    &nHeight,
    &nFrameRate,
    &nRecordProfile,
    &nRecordLevel,
    &nRecordEntropy,
    &nRecordComplexity,
    &nRecordMode,
    &nQuality,
    &nBitRate,
    &nGOP,
    &nBframe,
    &nIsInterleaved,
    &nSlices,
    &nLayers,
    &nSceneCut,
    &nMultiThread,
    &nMBBRC,
    &nExtBRC,
    &nMinQP,
    &nMaxQP,
    &nVBVMaxRate,
    &nVBVBufSize,
    &nAspectRatioX,
    &nAspectRatioY,
    &hAttachedWindow,
    &nThumbDraw,
    &nMaintainAspectRatio );

```

## 9.2.5 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_PROPERTY

## 9.2.6 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to set audio recording parameters in share recording.

For the most popular **MP4** file recording, we suggested using **H.264 + AAC** encoder. It is default format in Microsoft® Windows. Since the **AAC** codec is licensed, if a customer wants to avoid **AAC** licenses, the QCAP SDK also support **H.264 + PCM** audio format for general purpose recording.

For developer uses **TS, FLV, RTMP** and **HLS**, must set audio encoder format to **QCAP\_ENCODER\_FORMAT\_AAC\_ADTS**.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_AUDIO\_RECORD\_PROPERTY\_EX()**.



All audio data use **software encoder** in recording.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE
ULONG	nEncoderFormat	IN	Specify audio encoder format: QCAP_ENCODER_FORMAT_PCM QCAP_ENCODER_FORMAT_AAC QCAP_ENCODER_FORMAT_AAC_RAW QCAP_ENCODER_FORMAT_AAC_ADTS QCAP_ENCODER_FORMAT_MP2 QCAP_ENCODER_FORMAT_MP3 QCAP_ENCODER_FORMAT_OPUS
ULONG	nChannels	IN	Specify total audio channels
ULONG	nBitsPerSample	IN	Specify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency
ULONG	nBitRate	IN	Specify audio bit rate. default 128k and max is 496k <b>Only in QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set current audio recording properties in share recording*

[illegible]



## 9.2.7 QCAP\_GET\_AUDIO\_SHARE\_RECORD\_PROPERTY

## 9.2.8 QCAP\_GET\_AUDIO\_SHARE\_RECORD\_PROPERTY\_EX

### Introduction

The user can use this function to get audio record property parameters in share recording.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_AUDIO\_RECORD\_PROPERTY\_EX()**.



All audio data use **software encoder** in recording.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG *	pEncoderType	OUT	Pointer to the audio encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the audio encoder format
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency
ULONG *	pBitRate	OUT	Pointer to the audio bit rate <b>Only in QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Retrieve current audio recording properties in share recording*

```
QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY( 0, &nEncoderType, &nEncoderFormat,
&nChannels, &nBitsPerSample, &nSampleFrequency );

QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY_EX( 0, &nEncoderType, &nEncoderFormat,
&nChannels, &nBitsPerSample, &nSampleFrequency,
&nBitRate );
```

# 9.2.9 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_DYNAMIC\_PROPERTY\_EX

## Introduction

This function is only for software encoders to set current share recording properties *dynamically*, such as *bit rate* & *GOP reconfiguration... etc.*



In share recording, this function is For **Software encoder** only!

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify recording software encoder quality, from 0-10000. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify recording software encoder bit rate. It is used for <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording software encoder GOP size, from 0-255

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Change the current bit rate properties to share recording in run-time*

```
QCAP_SET_VIDEO_SHARE_RECORD_DYNAMIC_PROPERTY_EX( 0,
                                                QCAP_RECORD_MODE_CBR,
                                                8000,
                                                12 * 1024 * 1024,
                                                30 );
```

### 9.2.10 QCAP\_GET\_VIDEO\_SHARE\_RECORD\_DYNAMIC\_PROPERTY\_EX

## Introduction

This function is only for software encoders to retrieve current channel recording properties **dynamically**, such as *bit rate* & *GOP reconfiguration... etc.*



In share recording, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG *	pRecordMode	OUT	Pointer to the record mode
ULONG *	pQuality	OUT	Pointer to the quality
ULONG *	pBitRate	OUT	Pointer to the bit rate
ULONG *	pGOP	OUT	Pointer to the GOP size

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

.Example : Retrieve current bit rate setting of share recording properties in run-time

[illegible]

# 9.2.9 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_COPP

## Introduction

This function is used to set the share record COPP.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BOOL	Enable	OUT	Specify the value to the COPP

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the share record COPP*

```
QCAP_SET_VIDEO_SHARE_RECORD_COPP( 0, TRUE );
```

# 9.2.9 QCAP\_GET\_VIDEO\_SHARE\_RECORD\_COPP

## Introduction

This function is used to get the share record COPP.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BOOL *	pEnable	OUT	The pointer to the value of COPP

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the share record COPP*

```
QCAP_GET_VIDEO_SHARE_RECORD_COPP( 0, &Flag );
```

# 9.3 Share Record Data Functions

## Introduction

When share recording properties are set correctly, the next step is to push the real video data into the recording engine. Unlike *channel recording* user doesn't care about how the frame data are set into the recording process. In share recording, User can either push the uncompressed data from the capture device or push the compressed data comes from hardware encoder directly. For every data, frames pushed into the share recording engine, hence the video file output.



Figure 1. example of cropping video frame and scale in result.

Functions to push the audio/video frames to share recording engine:

mode	functions
For <b>Software Encoder</b>	QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER()
	QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()
	QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER()
	QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()
For <b>Hardware Encoder</b>	QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER()
	QCAP_SET_AUDIO_SHARE_RECORD_COMPRESSION_BUFFER()
	QCAP_SET_METADATA_SHARE_RECORD_DATA_BUFFER()

## 9.3.1 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER

## 9.3.2 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER\_EX

### Introduction

The user can call this function to push an uncompressed video buffer into share recording engine. In channel recording, the user doesn't need to know how the frame-buffer goes to recording engine. But in share recording, because a user can select which channel to record, it should be done by pushing the desired video buffer manually to recording engine.

This function is typically called in **QCAP\_REGISTER\_VIDEO\_PREVIEW\_CALLBACK()**, the example flow is:

- User can use *pDevice* to distinguish different channels and decide which one to use.
- The *bForceKeyFrame* parameter means to push a keyframe.
- The cropping parameters will decide new frame buffer resolution
- If the result frame buffer resolution is different from the recording frame, QCAP will enable video scaler to resize size it before encoding.
- Then the buffer will be sent to the encoder engine and generate its video bitstream.
- Finally, save to disk by the file writer.



In share recording, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG	nColorSpaceType	IN	Specify the input source buffer's color space type, : QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the input source buffer's width
ULONG	nHeight	IN	Specify the input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
ULONG	nCropX	IN	Specify the input source buffer's width <b>Only in</b> <b>QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropY	IN	Specify the input source buffer's height <b>Only in</b> <b>QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in</b> <b>QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in</b> <b>QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>





### 9.3.3 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER

### 9.3.4 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER\_EX

#### Introduction

The user can call this function to push an uncompressed audio buffer into share recording engine. In channel recording, the user doesn't need to know how the frame-buffer goes to recording engine. But in share recording, because the user can select which channel to record, it should be done by pushing the desired video buffer manually to recording engine.

This function is typically called in **QCAP\_REGISTER\_AUDIO\_PREVIEW\_CALLBACK()**, the example flow is:

- User can use *pDevice* to distinguish different channels and decide which one to use.
- the input audio format can be set in *nChannels nBitsPerSample nSampleFrequency* without re-sampling.
- The cropping parameters will decide new frame buffer resolution
- Then the buffer will be sent to the encoder engine and generate its audio bitstream.
- Finally, save to disk by the file writer.



In share recording, this function is For **Software encoder** only!

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG	nChannels	IN	Specify the total audio channels <b>Only in</b> <b>QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nBitsPerSample	IN	Specify the audio bits per sample <b>Only in</b> <b>QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nSampleFrequency	IN	Specify the audio sample frequency <b>Only in</b> <b>QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX()</b>
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push the desired channel's uncompressed audio buffer to share recording engine*

```
#define SET_AUDIO_FORMAT    1

QRETURN audio_preview_callback( PVOID pDevice, double dSampleTime, BYTE * pFrameBuffer, ULONG nFrameBufferLen,
PVOID pUserData )
{

    //select which channel to push video data
    if( Selected_Device = pDevice )
    {

#if SET_AUDIO_FORMAT

        QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER_EX( 0,
                                                                2, 16, 48000,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

#else

        QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0,
                                                            pFrameBuffer,
                                                            nFrameBufferLen );

#endif

    }

    return QCAP_RT_OK;
}
```

# 9.3.5 QCAP\_SET\_VIDEO\_SHARE\_RECORD\_COMPRESSION\_BUFFER

## Introduction

The user can call this function to push a compressed video buffer into share recording engine. In channel recording, a user doesn't need to know how the frame-buffer goes to recording engine. But in share recording, because a user can select which channel to record, it should be done by pushing the desired video buffer manually to recording engine.

This function is typically called in **QCAP\_REGISTER\_VIDEO\_HARDWARE\_ENCODER\_CALLBACK()**, the example flow is:

- User can use *pDevice* to distinguish different channels and decide which one to use.
- The *bForceKeyFrame* parameter means to push a keyframe.
- The buffer is already a compressed video stream, no software encoder involves.
- Finally, save to disk by the file writer.



In share recording, this function is For **Hardware encoder** only!

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BYTE *	pStreamBuffer	IN	Specify the input source buffer
ULONG	nStreamBufferLen	IN	Specify the input source buffer's size
ULONG	bIsKeyFrame	IN	Specify the input source's frame is keyframe or not
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push the desired channel's compressed video buffer to share recording engine*

```
QRETURN video_hardware_encoder_callback( PVOID pDevice, UINT iRecNum, double dSampleTime, BYTE * pStreamBuffer,
ULONG nStreamBufferLen, BOOL bIsKeyFrame, PVOID pUserData )
{
    //...
    if( pDevice == Selected_Device )
    {
        QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER( 0,
                                                         pStreamBuffer,
                                                         nStreamBufferLen,
                                                         bIsKeyFrame, dSampleTime);
    }
    //...
    return QCAP_RT_OK;
}
```

### 9.3.6 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_COMPRESSION\_BUFFER

## Introduction

The user can call this function to push a compressed audio buffer into share recording engine.

Currently there is no audio hardware encoder compressed stream support!



In share recording, this function is For **Hardware encoder** only!

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BYTE *	pStreamBuffer	IN	Specify the input source buffer
ULONG	nStreamBufferLen	IN	Specify the input source buffer's size
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push the desired channel's compressed audio buffer to share recording engine*

[illegible]

## 9.3.7 QCAP\_SET\_METADATA\_SHARE\_RECORD\_DATA\_BUFFER

### Introduction

Just like meta-data in channel recording, while share recording each frame also can send a user-defined information, or meta-data, that could output pre-frames to the video file. Then the meta-data can be retrieved by calling **QCAP\_GET\_METADATA\_FILE\_DATA\_BUFFER()** for different applications. This function can be called any time ( in a video callback, timer handlers) while share recording is in progress.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
BYTE *	pDataBuffer	IN	pointer to user-define meta-data buffer
ULONG	nDataBufferLen	IN	Specify the length of user-define meta-data buffer
double	dSampleTime	IN	<b>default 0.0</b> the sampling time in seconds

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set user-defined meta-data buffer to the beginning of video frame while share recording*

```
char *user_buffer = "this is the user-defined string to store in the first video frame.";

QCAP_START_SHARE_RECORD(0, "D:/CH01.AVI" );

QCAP_SET_METADATA_SHARE_RECORD_DATA_BUFFER( 0,
                                             user_buffer,
                                             strlen,
                                             0 );
```

## 9.3.7 QCAP\_SET\_METADATA\_SHARE\_RECORD\_HEADER

### Introduction

This function can set the header information in share record header.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to start recording, start from 0 Max RecNum is 0-63
CHAR *	ppszTitle	OUT	The title information in file header
CHAR *	ppszArtist	OUT	The artist information in file header
CHAR *	ppszComments	OUT	The comment information in file header
CHAR *	ppszGenre	OUT	The genre information in file header
CHAR *	ppszComposer	OUT	The composer information in file header

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To set the meta-data information to file header.*

```
QCAP_SET_METADATA_SHARE_RECORD_HEADER( 0,  
                                         title, artist, comment, genre, composer );
```

# 9.4 Share Record Audio Functions

## Introduction

This is section contains the audio property to set in the device, such as sound renderer can decide which output device to play the sound, and it's volume setting in share recording.

## 9.4.1 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_SOUND\_RENDERER

### Introduction

This function can set current sound renderer from available sound output device in share recording.



For more detailed parameters descriptions, please refer to `QCAP_SET_AUDIO_SOUND_RENDERER()`.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
UINT	iSoundNum	IN	Sound Renderer, default Renderer is 0

### Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

## 9.4.2 QCAP\_GET\_AUDIO\_SHARE\_RECORD\_SOUND\_RENDERER

### Introduction

This function can get current sound renderer of sound output device used in share recording.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
UINT *	pSoundNum	OUT	Pointer to the Sound number

### Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

### Examples

*Example : Get the current audio recording input device in share recording*

```
QCAP_SET_AUDIO_SHARE_RECORD_SOUND_RENDERER( 0, nSounder );

QCAP_GET_AUDIO_SHARE_RECORD_SOUND_RENDERER( 0, &nSounder );
```

### 9.4.3 QCAP\_SET\_AUDIO\_SHARE\_RECORD\_VOLUME

#### Introduction

This function can set current audio volume value in share recording, range from 0 to 100.

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG	nVolume	IN	Specify the share record volume

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set current audio volume output to 50% in share recording*

```
ULONG nVolume = 50;

QCAP_SET_AUDIO_SHARE_RECORD_VOLUME( 0, 50 );
```

### 9.4.4 QCAP\_GET\_AUDIO\_SHARE\_RECORD\_VOLUME

#### Introduction

This function can get current audio volume value in share recording, range from 0 to 100.

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
ULONG*	pVolume	OUT	Pointer to the share record volume

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get current audio volume output in share recording*

```
ULONG nVolume = 0;

QCAP_GET_AUDIO_SHARE_RECORD_VOLUME( 0, &nVolume );
```



## 9.5 Share Record Snapshot Functions

### Introduction

This chapter will help to take a snapshot of your current share recording video. Follow this guide to take a snapshot of your whole video display, or any section of the video you want. The user can save a snapshot to a **BMP/JPG**, or to trigger a snapshot to get the image stream buffer from a callback function without saving it to disk.

### 9.5.1 QCAP\_SNAPSHOT\_SHARE\_RECORD\_BMP

### 9.5.2 QCAP\_SNAPSHOT\_SHARE\_RECORD\_BMP\_EX

#### Introduction

This function takes a snapshot of video and saves to **BMP** 24bit or 32bit file in share recording.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_BMP\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.BMP24" → save to 24bit <b>BMP</b> "Filename.BMP32 or BMP" → save to 32bit <b>BMP</b> "BMP24" → To snapshot stream in callback only (no save to file.) "BMP32"/"BMP" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_BMP_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_BMP_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_BMP_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_BMP_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_BMP_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_BMP_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.



## 9.5.3 QCAP\_SNAPSHOT\_SHARE\_RECORD\_JPG

## 9.5.4 QCAP\_SNAPSHOT\_SHARE\_RECORD\_JPG\_EX

### Introduction

This function takes a snapshot of video and saves to **JPEG** image file format in share recording.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_JPG\_EX()**.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.JPG" → To snapshot to <b>JPEG</b> image file "JPG" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX()</b>
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100
BOOL	bIsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example 1: Take a snapshot in share recording, cropping a rectangle area and scale to 720x480 in **JPEG***

```
QCAP_SNAPSHOT_SHARE_RECORD_JPG( 0,
                                "C:/PICTURE1.JPG",
                                80 );

QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX( 0,
                                    "C:/PICTURE2.JPG",
                                    20, 80,
                                    1900, 1000,
                                    720, 480,
                                    100 );
```

*Example 2: Take a snapshot without saving to **JPEG** file in share recording*

```
QRETURN share_record_snapshot_stream_callback( PVOID pDevice,
                                                CHAR *pszFilePathName,
                                                BYTE *pStreamBuffer,
                                                LONG nStreamBufferLen,
                                                PVOID pUserData )
{
    //Get snapshot stream data here

    return QCAP_RT_OK;
}

QCAP_SNAPSHOT_SHARE_RECORD_JPG( 0, "JPG", 80 );

QCAP_SNAPSHOT_SHARE_RECORD_JPG_EX( 0,
                                    "JPG",
                                    10, 40,
                                    1900, 1000, 720, 480,
                                    80 );
```

## 9.6 Share Record OSD Functions

### Introduction

An on-screen display (OSD) are control functions on a video screen that allows you to draw text fields, overlapped pictures, or put customer image buffer with blending or color key effect.

For more detailed parameters descriptions, please refer to **Chapter 7 OSD Function API**.

### 9.6.1 QCAP\_SET\_OSD\_SHARE\_RECORD\_TEXT

### 9.6.2 QCAP\_SET\_OSD\_SHARE\_RECORD\_TEXT\_EX

### 9.6.3 QCAP\_SET\_OSD\_SHARE\_RECORD\_TEXT\_W

### 9.6.4 QCAP\_SET\_OSD\_SHARE\_RECORD\_TEXT\_EX\_W

### Introduction

The user can use this function to create a text field objects used for on-screen display.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_TEXT()**.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to auto calculate width.
INT	h	IN	Specify the height of OSD output Set -1 to auto calculate height.
CHAR * <a href="#">WSTRING</a>	pszString <a href="#">pwszString</a>	IN	Specify to display the text of OSD output <a href="#">Support wide character string</a>
CHAR * <a href="#">WSTRING</a>	pszFontFamilyName <a href="#">pwszFontFamilyName</a>	IN	Specify the font name used to display the text of OSD output <a href="#">Support wide character string</a>
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output : QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
DWORD	dwFontColor	IN	Specify the font color used to display the text of OSD output
DWORD	dwBackgroundColor	IN	Specify the background color used to display the text of OSD output
DWORD	dwBorderColor	IN	



[illegible]

# 9.6.7 QCAP\_SET\_OSD\_SHARE\_RECORD\_PICTURE

## Introduction

This OSD function displays one or more **BMP/JPG/PNG/GIF/EDL.INI** on top of video stream.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_PICTURE()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output. Set -1 to use original picture width.
INT	h	IN	Specify the height of OSD output. Set -1 to use original picture height.
CHAR *	pszFilePathName	IN	Specify the image file name to display in OSD Supported extensions: " <b>BMP</b> " as 24/32-bit, " <b>JPG</b> " and " <b>PNG</b> " Supported animation by <b>GIF,EDL.INI</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a half-transparent PNG on the top of captured video stream in share recording*

```
QCAP_SET_OSD_SHARE_RECORD_PICTURE( 0, 0, 0, 0,
                                   -1, -1,
                                   "C:/SAMPLE.PNG", 128,
                                   QCAP_SEQUENCE_STYLE_FOREMOST );
```



## 9.6.8 QCAP\_SET\_OSD\_SHARE\_RECORD\_BUFFER

## 9.6.8 QCAP\_SET\_OSD\_SHARE\_RECORD\_BUFFER\_EX

### Introduction

The user can use this function to create a frame buffer objects used for on-screen display.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_BUFFER()**.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to set recording parameters, start from 0 Max RecNum is 0-63
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output
INT	h	IN	Specify the height of OSD output
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameWidth	IN	Specify the width of frame contained in a buffer
ULONG	nFrameHeight	IN	Specify the height of frame contained in a buffer
ULONG	nFramePitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate by width and color space format.
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_SHARE_RECORD_BUFFER_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_SHARE_RECORD_BUFFER_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SET_OSD_SHARE_RECORD_BUFFER_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SET_OSD_SHARE_RECORD_BUFFER_EX()</b>
DWORD	dwBorderColor	IN	Specify the border color <b>Only in QCAP_SET_OSD_SHARE_RECORD_BUFFER_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width <b>Only in QCAP_SET_OSD_SHARE_RECORD_BUFFER_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	<b>default 0xFFFFFFFF</b> Specify the key color of share record OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY)



### 9.6.9 QCAP\_MOVE\_OSD\_SHARE\_RECORD\_OBJECT

## Introduction

The user can use this function to move the OSD object around the video window. It is useful to scroll the text string or picture on the video display window.



For more detailed parameters descriptions, please refer to `QCAP_MOVE_OSD_OBJECT()`.

## Parameters

type	parameter	I/O	descriptions
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To scroll the OSD object from left to right in share recording*

[illegible]

## 9.7 Share Record 3D functions

### Introduction

This section provides functions that can generate 3D video in share recording. The supported 3D video format are:

- Side-by-Side (SBS)
- Top-Bottom (TB)
- Line-by-Line (LBL)

This section contains 3D functions for:

#	3D Video Format Functions
1	<b>QCAP_SET_VIDEO_3D_SHARE_RECORD_UNCOMPRESSION_BUFFER()</b>
2	<b>QCAP_SET_VIDEO_3D_SHARE_RECORD_L_UNCOMPRESSION_BUFFER()</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_L_UNCOMPRESSION_BUFFER_EX()</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_R_UNCOMPRESSION_BUFFER()</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_R_UNCOMPRESSION_BUFFER_EX()</b>
3	<b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER()</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX()</b>

### 9.7.1 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER

#### Introduction

The user can use this function to push 3D uncompressed video buffer into share recorder engine. The user can use *bForceKeyFrame* parameter to push keyframe immediately.

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to 3D start recording, start from 0 Max RecNum is 0-63
ULONG	nStereoDisplayMode	IN	<b>default QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE</b> Specify 3D display mode: QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE QCAP_3D_STEREO_DISPLAY_MODE_LEFT_ONLY QCAP_3D_STEREO_DISPLAY_MODE_RIGHT_ONLY
BOOL	bLeftRightSwap	IN	<b>default FALSE</b> Swap the the share record video left/right outputs
BOOL	bForceKeyFrame	IN	<b>default FALSE</b> Enforce the input source's frame is keyframe, the default FALSE
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To set a 3D uncompressed buffer to share recording*

```
QCAP_SET_VIDEO_3D_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, 0, FALSE, FALSE, 0 );
```

## 9.7.2 Share Record 3D L/R Buffer Functions

### 9.7.2.1 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_L\_UNCOMPRESSION\_BUFFER

### 9.7.2.2 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_L\_UNCOMPRESSION\_BUFFER\_EX

### 9.7.2.3 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_R\_UNCOMPRESSION\_BUFFER

### 9.7.2.4 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_R\_UNCOMPRESSION\_BUFFER\_EX

#### Introduction

The user can use this function to push 3D stereo uncompressed video buffer into share recording engine. To push Left side/and Right side of stereo video frames, in result a 3D Side-by-Side video format in share recording.



This function is for Side-by-Side 3D video format only.

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to 3D start recording, start from 0 Max RecNum is 0-63
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the 3D of left input source buffer's width
ULONG	nHeight	IN	Specify the 3D of left input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the 3D of left input source buffer
ULONG	nFrameBufferLen	IN	Specify the 3D of left input source buffer's size
ULONG	nCropX	IN	Specify the x-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_R_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_R_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropW	IN	Specify the width of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_R_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropH	IN	Specify the height of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_R_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nScaleStyle	IN	<b>default QCAP_SCALE_STYLE_STRETCH</b> specify the scale styles: QCAP_SCALE_STYLE_STRETCH QCAP_SCALE_STYLE_FIT QCAP_SCALE_STYLE_FILL



## 9.7.3 Share Record 3D Stereo Buffer Functions

### 9.7.3.1 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_STEREO\_UNCOMPRESSION\_BUFFER

### 9.7.3.2 QCAP\_SET\_VIDEO\_3D\_SHARE\_RECORD\_STEREO\_UNCOMPRESSION\_BUFFER\_EX

#### Introduction

The user can use this function to push 3D stereo uncompressed video buffer into share recording engine. The buffer will pass to the encoder to generate its bitstream, then sent to the file writer to save to a file.

The cropping parameters can crop a display region and scale to the target video size.

#### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to 3D start recording, start from 0 Max RecNum is 0-63
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the 3D input source buffer's width
ULONG	nHeight	IN	Specify the 3D input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the 3D input source buffer
ULONG	nFrameBufferLen	IN	Specify the 3D input source buffer's size
ULONG	nCropX	IN	Specify the x-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropW	IN	Specify the width of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropH	IN	Specify the height of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nScaleStyle	IN	<b>default QCAP_SCALE_STYLE_STRETCH</b> specify the scale styles: QCAP_SCALE_STYLE_STRETCH QCAP_SCALE_STYLE_FIT QCAP_SCALE_STYLE_FILL <b>Only in</b> <b>QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nStereoBufferType	IN	<b>default QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE</b> Specify 3D stereo mode: QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE QCAP_3D_STEREO_BUFFER_TOP_BOTTOM QCAP_3D_STEREO_BUFFER_LINE_BY_LINE



### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push a 3D stereo video stream to share recording engine, and output 3D Side-by-Side format.*

```

QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER( 0,
    QCAP_COLORSPACE_TYEP_YUY2,
    1280, 720,
    pFrameBuffer,
    nFrameBufferLen,
    QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE );

QCAP_SET_VIDEO_3D_SHARE_RECORD_STEREO_UNCOMPRESSION_BUFFER_EX( 0,
    QCAP_COLORSPACE_TYEP_YUY2,
    1280, 720,
    pFrameBuffer,
    nFrameBufferLen,
    100, 150,
    720, 480,
    QCAP_SCALE_STYLE_STRETCH,
    QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE );

```

## 9.8 Share Record Mixer Functions

## Introduction

In this section provide the audio buffer mixer functions. The user can use these function to mix an audio buffer directly with the recording audio stream.

### 9.8.1 QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_PROPERTY\_EX

## Introduction

The user can use this function to set audio mixer format property in share recording. The *nTracks* parameter can set how many audio tracks to create, the user can create four audio track at same time.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to start recording, start from 0 Max RecNum is 0-63
ULONG	nTracks	IN	Specify audio track number, range: 0-3
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE
ULONG	nEncoderFormat	IN	Specify audio encoder format: QCAP_ENCODER_FORMAT_PCM QCAP_ENCODER_FORMAT_AAC QCAP_ENCODER_FORMAT_AAC_ADTS
ULONG	nChannels	IN	Specify the total audio channels
ULONG	nBitsPerSample	IN	Specify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency
ULONG	nBitRate	IN	Specify audio bit rate. default 128k and max is 496k

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

### Example : Set audio mixer format property in share recording

[illegible]

### 9.8.2 QCAP\_GET\_AUDIO\_MX\_SHARE\_RECORD\_PROPERTY\_EX

## Introduction

The user can use this function to get audio mixer property information, when the users use the audio mixer function in share recording.+

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to start recording, start from 0 Max RecNum is 0-63
ULONG *	pTracks	OUT	Pointer to the audio tracks
ULONG *	pEncoderType	OUT	Pointer to the audio encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the audio encoder format
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency
ULONG *	pBitRate	OUT	Pointer to the audio bit rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get audio mixer property in share recording*

[illegible]

9.8.3

QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_MIXING\_UNCOMPRESSION\_BUFFER

9.8.3

QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_MIXING\_UNCOMPRESSION\_BUFFER\_EX

Introduction

The user can use this function to mix audio uncompressed buffer on audio preview callback in share recording. The user must set **QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_MIXING\_UNCOMPRESSION\_BUFFER()** to mix audio recording first before calls **QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER()**.

Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to start recording, start from 0 Max RecNum is 0-63
UINT	iTrackNum	IN	Specify audio track number
UINT	iMixNum	IN	Specify the audio mixer number
ULONG	nChannels	IN	Specify the total audio channels Only in QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER_EX()
ULONG	nBitsPerSample	IN	Specify the audio bits per sample Only in QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER_EX()
ULONG	nSampleFrequency	IN	Specify the audio sample frequency Only in QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER_EX()
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

Example : To mix audio uncompressed buffer in shared recording

```
QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER( 0, 0, 0,
                                                             pFrameBuffer,
                                                             nFrameBufferLen );

QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER( 0, 1, 0,
                                                             pFrameBuffer,
                                                             nFrameBufferLen );

QCAP_SET_AUDIO_MX_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, 0 );

QCAP_SET_AUDIO_MX_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, 1 );
```

## 9.8.4 QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_UNCOMPRESSION\_BUFFER

### Introduction

The user can use this function to set audio share recording uncompressed buffer on audio preview callback in share recording.

### Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to start recording, start from 0 Max RecNum is 0-63
UINT	iTrackNum	IN	Specify audio track number
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame.

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To mix audio uncompressed buffer*

```
QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER( 0, 0,
                                                             pFrameBuffer,
                                                             nFrameBufferLen );

QCAP_SET_AUDIO_MX_SHARE_RECORD_MIXING_UNCOMPRESSION_BUFFER( 0, 1,
                                                             pFrameBuffer,
                                                             nFrameBufferLen );

QCAP_SET_AUDIO_MX_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, 0 );

QCAP_SET_AUDIO_MX_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, 1 );
```

### 9.8.5 QCAP\_SET\_AUDIO\_MX\_SHARE\_RECORD\_COMPRESSION\_BUFFER

## Introduction

The user can use this function to set audio share record compression buffer on audio share record callback.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot to start recording, start from 0 Max RecNum is 0-63
UINT	iTrackNum	IN	Specify audio track number
BYTE *	pStreamBuffer	IN	Pointer to the input source buffer
ULONG	nStreamBufferLen	IN	Specify the max input source buffer's size
double	dSampleTime	IN	<b>default 0.0</b> Specify the sampling time in seconds

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

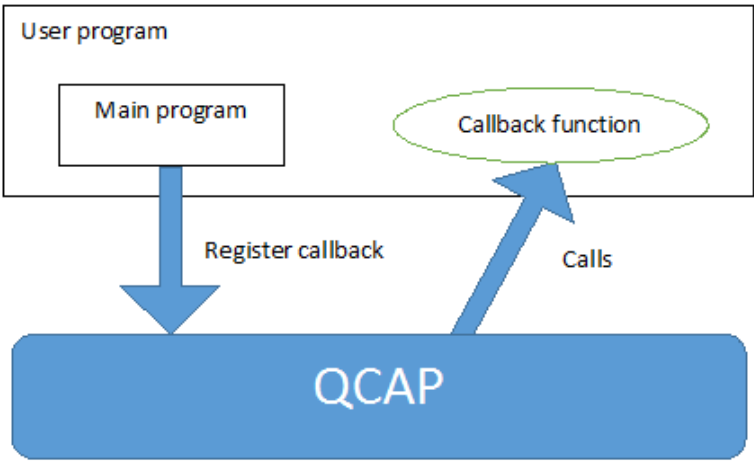
## Examples

*Example : Set audio mixer compression buffer in shared recording*

[illegible]

# 9.9 Share Record Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register share recording callback functions for:

Register functions	Callback functions
QCAP_REGISTER_SHARE_RECORD_DONE_CALLBACK	PF_SHARE_RECORD_DONE_CALLBACK
QCAP_REGISTER_SHARE_RECORD_FAIL_CALLBACK	PF_SHARE_RECORD_FAIL_CALLBACK
<b>Stream Callbacks</b>	
QCAP_REGISTER_VIDEO_SHARE_RECORD_CALLBACK	PF_VIDEO_SHARE_RECORD_CALLBACK
QCAP_REGISTER_AUDIO_SHARE_RECORD_CALLBACK	PF_AUDIO_SHARE_RECORD_CALLBACK
QCAP_REGISTER_MEDIA_SHARE_RECORD_CALLBACK	PF_MEDIA_SHARE_RECORD_CALLBACK
<b>Snapshot Callbacks</b>	
QCAP_REGISTER_SHARE_RECORD_SNAPSHOT_DONE_CALLBACK	PF_SHARE_RECORD_SNAPSHOT_DONE_CALLBACK
QCAP_REGISTER_SHARE_RECORD_SNAPSHOT_STREAM_CALLBACK	PF_SHARE_RECORD_SNAPSHOT_STREAM_CALLBACK
<b>Decoder Callbacks</b>	
QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK	PF_VIDEO_DECODER_SHARE_RECORD_CALLBACK
QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK	PF_AUDIO_DECODER_SHARE_RECORD_CALLBACK
<b>Audio Mixer Callbacks</b>	
QCAP_REGISTER_AUDIO_MX_SHARE_RECORD_CALLBACK	PF_AUDIO_MX_SHARE_RECORD_CALLBACK
QCAP_REGISTER_AUDIO_DECODER_MX_SHARE_RECORD_CALLBACK	PF_AUDIO_DECODER_MX_SHARE_RECORD_CALLBACK
<b>Media Timer Callbacks</b>	
QCAP_REGISTER_VIDEO_SHARE_RECORD_MEDIA_TIMER_CALLBACK	PF_VIDEO_SHARE_RECORD_MEDIA_TIMER_CALLBACK
QCAP_REGISTER_AUDIO_SHARE_RECORD_MEDIA_TIMER_CALLBACK	PF_AUDIO_SHARE_RECORD_MEDIA_TIMER_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

# 9.9.1 QCAP\_REGISTER\_SHARE\_RECORD\_DONE\_CALLBACK

## Introduction

The user can register a **PF\_SHARE\_RECORD\_DONE\_CALLBACK** function to get notification when the share recording process is completed. This callback function needs to be registered before share recording starts.



For more detailed parameters descriptions, please refer to **QCAP\_REGISTER\_RECORD\_DONE\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<b>PF_SHARE_RECORD_DONE_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_SHARE\_RECORD\_DONE\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
CHAR *	pszFilePathName	The video recorded filename
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for share recording to notify completion*

```
QRETURN share_record_done( UINT iRecNum,
                           CHAR * pszFilePathName,
                           PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SHARE_RECORD_DONE_CALLBACK pCB = share_record_done;

    QCAP_REGISTER_SHARE_RECORD_DONE_CALLBACK( 0, pCB, pUserData );
}
```



# 9.9.1 QCAP\_REGISTER\_SHARE\_RECORD\_FAIL\_CALLBACK

## Introduction

The user can register a *PF\_SHARE\_RECORD\_FAIL\_CALLBACK* function to get notification when the share recording process is fail for some reason.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<i>PF_SHARE_RECORD_FAIL_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_SHARE\_RECORD\_FAIL\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
CHAR *	pszFilePathName	The video recorded filename
QRESULT	nErrorStatus	IN
The error status of record fail	PVOID	pUserData

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for share recording to notify fail*

```
QRETURN share_record_fail( UINT iRecNum,
                           CHAR * pszFilePathName,
                           QRESULT nErrorStatus,
                           PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SHARE_RECORD_FAIL_CALLBACK pCB = share_record_fail;

    QCAP_REGISTER_SHARE_RECORD_FAIL_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.2 QCAP\_REGISTER\_VIDEO\_SHARE\_RECORD\_CALLBACK

## Introduction

When in share recording, this callback function will provide video compressed frames data callback no matter software encodes / hardware encoder is used.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<b>PF_VIDEO_SHARE_RECORD_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_VIDEO\_SHARE\_RECORD\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source frame buffer
ULONG	nStreamBufferLen	Specify the length of source frame buffer
BOOL	blsKeyFrame	Specify the input source's frame is keyframe or not
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed. The video compressed data will be sent to file writer
<b>QCAP_RT_FAIL</b>	The frame will be dropped and will not be saved on the file

## Examples

*Example : Register a callback function to get share recording video compressed data*

```
QRETURN video_share_record_callback( UINT iRecNum,
                                     double dSampleTime,
                                     BYTE * pStreamBuffer,
                                     ULONG nStreamBufferLen,
                                     BOOL bIsKeyFrame,
                                     PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_SHARE_RECORD_CALLBACK pCB = video_share_record_callback;

    QCAP_REGISTER_VIDEO_SHARE_RECORD_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.3 QCAP\_REGISTER\_AUDIO\_SHARE\_RECORD\_CALLBACK

## Introduction

When in share recording, this callback function will provide **PCM / AAC** audio data frames callback no matter software encode / hardware encoder is used.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<b>PF_AUDIO_SHARE_RECORD_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_AUDIO\_SHARE\_RECORD\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source frame buffer
ULONG	nStreamBufferLen	Specify the length of source frame buffer
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed. The video compressed data will be sent to file writer
<b>QCAP_RT_FAIL</b>	The frame will be dropped and will not be saved on the file

## Examples

*Example : Register a callback function to get share recording audio compressed data*

```
QRETURN audio_share_record_callback( UINT iRecNum,
                                     double dSampleTime,
                                     BYTE * pStreamBuffer,
                                     ULONG nStreamBufferLen,
                                     PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_AUDIO_SHARE_RECORD_CALLBACK pCB = audio_share_record_callback;

    QCAP_REGISTER_AUDIO_SHARE_RECORD_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.4 QCAP\_REGISTER\_MEDIA\_SHARE\_RECORD\_CALLBACK

## Introduction

The user can register a **PF\_MEDIA\_SHARE\_RECORD\_CALLBACK** function to process the media stream content (both audio/video) for every frame in a recording. Since media record buffer is used for streaming purpose, so it only supports for **TS** and **FLV** video format.



For more detailed parameters descriptions, please refer to **QCAP\_REGISTER\_MEDIA\_RECORD\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<b>PF_MEDIA_SHARE_RECORD_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_MEDIA\_SHARE\_RECORD\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source frame buffer
ULONG	nStreamBufferLen	Specify the length of source frame buffer
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for share recording streaming frame in recording*

```
QRETURN media_share_record_callback( UINT iRecNum,
                                     double dSampleTime,
                                     BYTE * pStreamBuffer,
                                     ULONG nStreamBufferLen,
                                     PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_MEDIA_SHARE_RECORD_CALLBACK pCB = media_share_record_callback;

    QCAP_REGISTER_MEDIA_SHARE_RECORD_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.5 QCAP\_REGISTER\_SHARE\_RECORD\_SNAPSHOT\_DONE\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_SHARE\_RECORD\_BMP/JPG()** is completed. For users who use asynchronous mode, can use this function to know when snapshot file is ready.



For more detailed parameters descriptions, please refer to **QCAP\_REGISTER\_SNAPSHOT\_DONE\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<i>PF_SHARE_RECORD_SNAPSHOT_DONE_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_SHARE\_RECORD\_SNAPSHOT\_DONE\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
CHAR *	pszFilePathName	pointer to snapshot filename
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for share recording snapshot done*

```
QRETURN share_record_snapshot_done_callback( UINT iRecNum,
                                             CHAR *pszFilePathName,
                                             PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SHARE_RECORD_SNAPSHOT_DONE_CALLBACK pCB = share_record_snapshot_done_callback;

    QCAP_REGISTER_SHARE_RECORD_SNAPSHOT_DONE_CALLBACK( pDevice, pCB, pUserData );
}
```

# 9.9.6 QCAP\_REGISTER\_SHARE\_RECORD\_SNAPSHOT\_STREAM\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_SHARE\_RECORD\_BMP/JPG()** image stream is generated, then user can get image stream in buffer directly.



For more detailed parameters descriptions, please refer to **QCAP\_REGISTER\_SNAPSHOT\_STREAM\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<b>PF_SHARE_RECORD_SNAPSHOT_STREAM_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_SHARE\_RECORD\_SNAPSHOT\_STREAM\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
CHAR *	pszFilePathName	pointer to snapshot filename
BYTE *	pStreamBuffer	Pointer to the image frame buffer
ULONG	nStreamBufferLen	Specify the length of image frame buffer
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for snapshot stream completion in share recording*

```
// SNAPSHOT_STREAM_CALLBACK
QRETURN share_record_snap_stream_callback( UINT iRecNum,
                                           CHAR *pszFilePathName,
                                           BYTE *pStreamBuffer,
                                           LONG nStreamBufferLen,
                                           PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SHARE_RECORD_SNAPSHOT_STREAM_CALLBACK pCB = share_record_snap_stream_callback;

    QCAP_REGISTER_SHARE_RECORD_SNAPSHOT_STREAM_CALLBACK( pDevice, pCB, pUserData );
}
```

# 9.9.7 QCAP\_REGISTER\_VIDEO\_DECODER\_SHARE\_RECORD\_CALLBACK

## Introduction

The user can register this callback function to get the decoding uncompressed video data from share recording, before display on the preview window.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<i>PF_VIDEO_DECODER_SHARE_RECORD_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## *PF\_VIDEO\_DECODER\_SHARE\_RECORD\_CALLBACK*

### Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer's size
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	to drop the frame and don't display on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	to skip display on the window

## Examples

*Example : Register a callback function for video decoding data callback in share recording*

```
QRETURN video_decoder_share_record_callback( UINT iRecNum,
                                             double dSampleTime,
                                             BYTE * pFrameBuffer,
                                             ULONG nFrameBufferLen,
                                             PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_DECODER_SHARE_RECORD_CALLBACK pCB = video_decoder_share_record_callback;

    QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK( pDevice, pCB, pUserData );
}
```

# 9.9.8 QCAP\_REGISTER\_AUDIO\_DECODER\_SHARE\_RECORD\_CALLBACK

## Introduction

The user can register this callback function to get the decoding uncompressed audio data from share recording, before playback on the preview window.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify share recorder slot of recorder, start from 0
<i>PF_AUDIO_DECODER_SHARE_RECORD_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## *PF\_AUDIO\_DECODER\_SHARE\_RECORD\_CALLBACK*

### Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer's size
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The frame buffer will be dropped and its sound won't be playback
<b>QCAP_RT_SKIP_DISPLAY</b>	The audio uncompressed data will not be played on the window.

## Examples

*Example : Register a callback function for audio decoding data callback in share recording*

```
QRETURN audio_decoder_share_record_callback( UINT iRecNum,
                                             double dSampleTime,
                                             BYTE * pFrameBuffer,
                                             ULONG nFrameBufferLen,
                                             PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_AUDIO_DECODER_SHARE_RECORD_CALLBACK pCB = audio_decoder_share_record_callback;

    QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK( 0, pCB, pUserData );
}
```



# 9.9.9 QCAP\_REGISTER\_AUDIO\_MX\_SHARE\_RECORD\_CALLBACK

## Introduction

The user can register this callback function to get the audio mixer data from share recording.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<b>PF_AUDIO_MX_SHARE_RECORD_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_MX\_SHARE\_RECORD\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
UINT	iTrackNum	Specify track number of the mixer
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source frame buffer
ULONG	nStreamBufferLen	Specify the length of source frame buffer
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The frame stream buffer will be dropped and will not be saved on the file.

## Examples

*Example : Register a callback function for audio mixer data callback in share recording*

```
QRETURN audio_mx_share_record_callback( UINT iRecNum,
                                         double dSampleTime,
                                         BYTE * pFrameBuffer,
                                         ULONG nFrameBufferLen,
                                         PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_AUDIO_MX_SHARE_RECORD_CALLBACK pCB = audio_mx_share_record_callback;

    QCAP_REGISTER_AUDIO_MX_SHARE_RECORD_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.10 QCAP\_REGISTER\_AUDIO\_DECODER\_MX\_SHARE\_RECORD\_CALLBACK

## Introduction

The user can register this callback function to get the decoding uncompressed audio decoder mixer data from share recording, before playback on the preview window.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<b><i>PF_AUDIO_DECODER_MX_SHARE_RECORD_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## ***PF\_AUDIO\_DECODER\_MX\_SHARE\_RECORD\_CALLBACK***

### *Parameters of Callback*

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
UINT	iTrackNum	Specify track number of the mixer
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer's size
PVOID	pUserData	Pointer to custom user data

### *Return value of Callback*

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The frame buffer will be dropped and its sound will be muted.
<b>QCAP_RT_SKIP_DISPLAY</b>	The audio uncompressed data will not be played on the window.

## Examples

*Example : Register a callback function for audio decoding mixer data callback in share recording*

```
QRETURN audio_decoder_mx_share_record_callback( UINT iRecNum,
                                                UINT iTrackNum,
                                                double dSampleTime,
                                                BYTE * pFrameBuffer,
                                                ULONG nFrameBufferLen,
                                                PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_AUDIO_DECODER_MX_SHARE_RECORD_CALLBACK pCB = audio_decoder_mx_share_record_callback;

    QCAP_REGISTER_AUDIO_DECODER_MX_SHARE_RECORD_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.11 QCAP\_REGISTER\_VIDEO\_SHARE\_RECORD\_MEDIA\_TIMER\_CALLBACK

## Introduction

This callback function will provide a video media timer function to video share recorder.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<i>PF_VIDEO_SHARE_RECORD_MEDIA_TIMER_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## *PF\_VIDEO\_SHARE\_RECORD\_MEDIA\_TIMER\_CALLBACK*

### Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
double	dDelayTime	Specify the video delay time
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for video media timer in share recording*

```
QRETURN video_share_record_media_timer_callback( UINT iRecNum,
                                                double dSampleTime,
                                                double dDelayTime,
                                                PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIDEO_SHARE_RECORD_MEDIA_TIMER_CALLBACK pCB = video_share_record_media_timer_callback;

    QCAP_REGISTER_VIDEO_SHARE_RECORD_MEDIA_TIMER_CALLBACK( 0, pCB, pUserData );
}
```

# 9.9.12 QCAP\_REGISTER\_AUDIO\_SHARE\_RECORD\_MEDIA\_TIMER\_CALLBACK

## Introduction

This callback function will provide an audio media timer function to video share recorder.

## Parameters

type	parameter	I/O	descriptions
UINT	iRecNum	IN	Specify the recorder slot of recorder, start from 0
<b>PF_AUDIO_SHARE_RECORD_MEDIA_TIMER_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_AUDIO\_SHARE\_RECORD\_MEDIA\_TIMER\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
UINT	iRecNum	Specify the recorder slot of recorder, start from 0
double	dSampleTime	The sampling time in seconds
double	dDelayTime	Specify the audio delay time
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for audio media timer in share recording*

```
QRETURN audio_share_record_media_timer_callback( UINT iRecNum,
                                                double dSampleTime,
                                                double dDelayTime,
                                                PVOID pUserData )
{
    return QCAP_RT_OK;
}

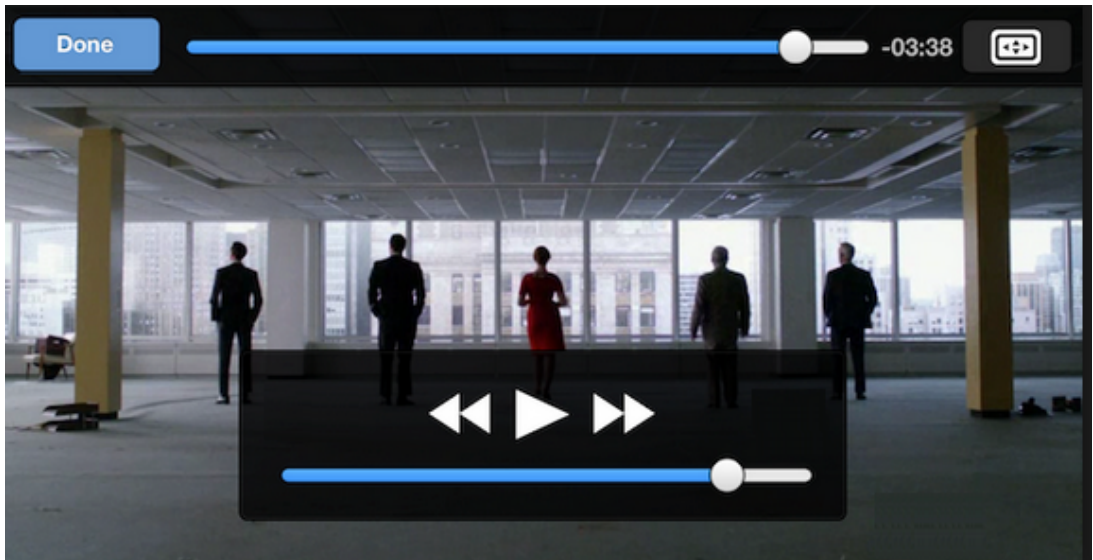
void test_callback()
{
    PF_VIDEO_SHARE_RECORD_MEDIA_TIMER_CALLBACK pCB = audio_share_record_media_timer_callback;

    QCAP_REGISTER_AUDIO_SHARE_RECORD_MEDIA_TIMER_CALLBACK( 0, pCB, pUserData );
}
```

# 10 File Playback/Editing Function API

---

## Introduction



QCAP SDK offers a set of software functions with an extensive set of features for creating robust custom video player applications. The File Playback/Editing APIs enables a user to incorporate video playback functionality into your applications. The API defines methods for loading vFor example, by using the API, you can play, pause, or seek to a specific point in the currently loaded video, or trim the unless part of your recording video, also can control playback programmatically.

For example, by using the API, you can play, pause, or seek to a specific point in the currently loaded video, or trim the unless part of your recording video, also can control playback programmatically.

# 10.1 Playback Major Function

## Introduction

This section provides functions to open a recorded video files, start/stop playing videos, set the current video playing position, and set the playback speed. It provides a professional playback function for users to easily view their video files after recorded.

### 10.1.1 QCAP\_OPEN\_FILE

### 10.1.2 QCAP\_OPEN\_FILE\_EX

## Introduction

The user can use this function to open a recorded video file for on-screen video playback. If a user wants to open the video in editing mode only (no display), please set the *hAttachedWindow* to NULL.

The extended **QCAP\_OPEN\_FILE\_EX()** has identical arguments with its original version, and is a new implementation of a professional video playback function to replace Microsoft® **DirectShow** framework (seek by frames), and must be used alone with these functions:

- **QCAP\_SET\_FILE\_POSITION\_EX()**
- **QCAP\_GET\_FILE\_POSITION\_EX()**
- **QCAP\_REGISTER\_VIDEO\_DECODER\_FILE\_CALLBACK\_EX()**
- **QCAP\_REGISTER\_AUDIO\_DECODER\_FILE\_CALLBACK\_EX()**



For more detailed parameters descriptions, please refer to **QCAP\_CREATE()**.

## Parameters

type	parameter	I/O	descriptions
CHAR *	pszFileName	IN	Specify the video file name to display video output, Supported format is <b>MP4</b> , <b>AVI</b>
PVOID *	ppFile	OUT	Handle of the file object
ULONG	nDecoderType	IN	Specify the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
ULONG *	pVideoEncoderFormat	OUT	Pointer to the video format
ULONG *	pVideoWidth	OUT	Pointer to the video width
ULONG *	pVideoHeight	OUT	Pointer to the video height
double *	pVideoFrameRate	OUT	Pointer to the video frame rate
ULONG *	pAudioEncoderFormat	OUT	Pointer to the audio format
ULONG *	pAudioChannels	OUT	Pointer to the total audio channels
ULONG *	pAudioBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pAudioSampleFrequency	OUT	Pointer to the audio sample frequency

double *	pTotalDurationTimes	OUT	Pointer to the total duration times
ULONG *	pTotalVideoFrames	OUT	Pointer to the total video frames
ULONG *	pTotalAudioFrames	OUT	Pointer to the audio frames
ULONG *	pTotalMetadataFrames	OUT	pointer to total meta-data frames available
HWND	hAttachedWindow	IN	Handle of the window to show the preview video. Set NULL to use in editing mode only.
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : Open a video file to playback on the window*

//original version

```
QCAP_OPEN_FILE( "D:/CH01.AVI",
                &pFile,
                QCAP_DECODER_TYPE_SOFTWARE,
                &nVideoFormat,
                &nVideoWidth,
                &nVideoHeight,
                &nVideoFrameRate,
                &nAudioFormat,
                &nAudioChannels,
                &nAudioBitsPerSample,
                &nAudioSampleFrequency,
                &nTotalDurationTimes,
                &nTotalVideoFrames,
                &nTotalAudioFrames,
                &nTotalMetadataFrames,
                hAttachedWindow );
```

//new implementation professional playback function

```
QCAP_OPEN_FILE_EX( "D:/CH01.MP4",
                  &pFile,
                  QCAP_DECODER_TYPE_SOFTWARE,
                  &nVideoFormat,
                  &nVideoWidth,
                  &nVideoHeight,
                  &nVideoFrameRate,
                  &nAudioFormat,
                  &nAudioChannels,
                  &nAudioBitsPerSample,
                  &nAudioSampleFrequency,
                  &nTotalDurationTimes,
                  &nTotalVideoFrames,
                  &nTotalAudioFrames,
                  &pTotalMetadataFrames,
                  hAttachedWindow );
```



## 10.1.3 QCAP\_OPEN\_TIMESHIFT\_FILE\_EX

### Introduction

In the timeshift recording, a user can continue recording one video while playing back the same video recording file. That means a user can pause the video, instant replay and jump back to specific times to replay your favorite scenes while the video recording. to have timeshift function, a user needs to use time-shifting recording API instead of normal recording, then a user can playback the video in recording on-the-fly, just like time-shifted.

The *ppPhysicalFileWriter* parameter is returned by a start recording API :

- QCAP\_START\_TIMESHIFT\_RECORD()
- QCAP\_START\_TIMESHIFT\_SHARE\_RECORD()
- QCAP\_START\_BROADCAST\_CLIENT\_TIMESHIFT\_RECORD()

is a file handle in which a time-shifting recording is in progress. This file handle could be used by this timeshift function and other playback functions.

The time-shifted function currently supports **MP4** format only.



If you want to access this file in editing mode, you can set the *hAttachedWindow* as NULL.

For timeshift video file, there is a **QCAP\_REFRESH\_TIMESHIFT\_FILE\_INFO()** function can get the video files information in run-time.

Just like the extended **QCAP\_OPEN\_FILE\_EX()**, this function is a new implementation of a professional video playback function to replace Microsoft® **DirectShow** framework (seek by frames), and must be used alone with these functions:

- **QCAP\_SET\_FILE\_POSITION\_EX()**
- **QCAP\_GET\_FILE\_POSITION\_EX()**
- **QCAP\_REGISTER\_VIDEO\_DECODER\_FILE\_CALLBACK\_EX()**
- **QCAP\_REGISTER\_AUDIO\_DECODER\_FILE\_CALLBACK\_EX()**



For more detailed parameters descriptions, please refer to **QCAP\_CREATE()**.

Parameters

type	parameter	I/O	descriptions
PVOID	pPhysicalFileWriter	IN	Handle of Physical File Writer object This object can be obtained from a start recording API
PVOID *	ppFile	OUT	Handle of the file object
ULONG	nDecoderType	IN	Specify the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
ULONG *	pVideoEncoderFormat	OUT	Pointer to the video format
ULONG *	pVideoWidth	OUT	Pointer to the video width
ULONG *	pVideoHeight	OUT	Pointer to the video height
double *	pVideoFrameRate	OUT	Pointer to the video frame rate
ULONG *	pAudioEncoderFormat	OUT	Pointer to the audio format
ULONG *	pAudioChannels	OUT	Pointer to the total audio channels
ULONG *	pAudioBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pAudioSampleFrequency	OUT	Pointer to the audio sample frequency
double *	pTotalDurationTimes	OUT	Pointer to the total duration times
ULONG *	pTotalVideoFrames	OUT	Pointer to the total video frames
ULONG *	pTotalAudioFrames	OUT	Pointer to the audio frames
ULONG *	pTotalMetadataFrames	OUT	pointer to total meta-data frames available
HWND	hAttachedWindow	IN	Handle of the window to show the preview video. Set NULL to use in editing mode only.
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

*Example : start a time-shifted recording for **MP4**, and playback the video (still in recording)*

```

QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_SOFTWARE,
                                QCAP_ENCODER_FORMAT_AAC ); //MP4 use AAC audio

QCAP_SET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
                                QCAP_ENCODER_FORMAT_H264,
                                QCAP_RECORD_MODE_CBR,
                                8000,
                                12*1024*1024, //bit rate=12M
                                30,
                                0,
                                0,
                                QCAP_DOWNSCALE_MODE_OFF ); //downscale off (1920x1080)

PVOID pPhysicalFileWriter = NULL;

PVOID pFile = NULL;

QCAP_START_TIMEShift_RECORD( pDevice, 0,
                             "D:/CH01.MP4",
                             &pPhysicalFileWriter );

QCAP_OPEN_TIMEShift_FILE_EX( pPhysicalFileWriter,
                             &pFile,
                             QCAP_DECODER_TYPE_SOFTWARE,
                             &nVideoFormat,
                             &nVideoWidth,
                             &nVideoHeight,
                             &nVideoFrameRate,
                             &nAudioFormat,
                             &nAudioChannels,
                             &nAudioBitsPerSample,
                             &nAudioSampleFrequency,
                             &nTotalDurationTimes,
                             &nTotalVideoFrames,
                             &nTotalAudioFrames,
                             hAttachedWindow );

QCAP_PLAY_FILE( m_pFile ); //playback the video (still in time-shifted recording)

```

# 10.1.4 QCAP\_OPEN\_SCF\_FILE

## Introduction

The user can use this function to create one video **SCF** file for on-screen playback. If you want to use it in editing mode only (no display), please set *hAttachedWindow* as NULL.

If user **QCAP\_OPEN\_SCF\_FILE()** to open a **SCF** video file, there are more functions in **Playback SCF functions** section.



For more detailed parameters descriptions, please refer to **QCAP\_CREATE()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iChNum	IN	Specify channel number to open <b>SCF</b> file, start from 0
PVOID *	ppFile	OUT	Handle of the file object
ULONG	nDecoderType	IN	Specify the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
HWND	hAttachedWindow	IN	Handle of the window to show the preview video. Set NULL to use in editing mode only.
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio
double	dStartSampleTime	IN	<b>default 0.0</b> Specify the start sample time in seconds
double	dStopSampleTime	IN	<b>default 0.0</b> Specify the stop sample time in seconds

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To open a **SCF** file for video playback*

```
QCAP_OPEN_SCF_FILE( 0,
    &pFile,
    QCAP_DECODER_TYPE_SOFTWARE,
    hWnd,
    TRUE,
    FALSE,
    0, 0 );
```

## 10.1.5 QCAP\_OPEN\_3D\_FILE

### Introduction

The user can use this function to open a 3D/2D video file for on-screen playback. Currently supported video format is: **TS**. If user wants to open the video in editing mode only (no display), please set the *hAttachedWindow* to NULL.

If user **QCAP\_OPEN\_3D\_FILE()** to open a 3D/2D video file, there are more functions in **Playback 3D functions** section.



For more detailed parameters descriptions, please refer to **QCAP\_CREATE()**.

### Parameters

type	parameter	I/O	descriptions
CHAR *	pszFileName	IN	Specify the video file name to display video output, Supported extensions: <b>"TS"</b>
PVOID *	ppFile	OUT	Handle of the file object
BOOL *	pls3D	OUT	If true, input file is 3D file. If false, input file is 2D file.
ULONG	nDecoderType	IN	Specify the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
ULONG *	pVideoEncoderFormat	OUT	Pointer to the video format
ULONG *	pVideoWidth	OUT	Pointer to the video width
ULONG *	pVideoHeight	OUT	Pointer to the video height
double *	pVideoFrameRate	OUT	Pointer to the video frame rate
ULONG *	pAudioEncoderFormat	OUT	Pointer to the audio format
ULONG *	pAudioChannels	OUT	Pointer to the total audio channels
ULONG *	pAudioBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pAudioSampleFrequency	OUT	Pointer to the audio sample frequency
double *	pTotalDurationTimes	OUT	Pointer to the total duration times
ULONG *	pTotalVideoFrames	OUT	Pointer to the total video frames
ULONG *	pTotalAudioFrames	OUT	Pointer to the audio frames
ULONG *	pTotalMetadataFrames	OUT	pointer to total meta-data frames available
HWND	hAttachedWindowL	IN	Handle of the window to show the preview video of left side. Set NULL to use in editing mode only.
BOOL	bThumbDrawL	IN	<b>default FALSE</b> Enable/Disable the left side thumb draw renderer
BOOL	bMaintainAspectRatioL	IN	<b>default FALSE</b> Enable/Disable the left side maintain aspect ratio
HWND	hAttachedWindowR	IN	<b>default NULL</b> Handle of the window to show the preview video of right side. Set NULL to use in editing mode only.
BOOL	bThumbDrawR	IN	<b>default FALSE</b> Enable/Disable the right side thumb draw renderer
BOOL	bMaintainAspectRatioR	IN	<b>default FALSE</b> Enable/Disable the right side maintain aspect ratio

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Open a 3D/2D video file for video playback*

```
QCAP_OPEN_3D_FILE( "D:/DEMO3D.TS", &pFile,  
                  &pIs3D,  
                  QCAP_DECODER_TYPE_SOFTWARE,  
                  &nVideoFormat,  
                  &nVideoWidth,  
                  &nVideoHeight,  
                  &nVideoFrameRate,  
                  &nAudioFormat,  
                  &nAudioChannels,  
                  &nAudioBitsPerSample,  
                  &nAudioSampleFrequency,  
                  &nTotalDurationTimes,  
                  &nTotalVideoFrames,  
                  &nTotalAudioFrames,  
                  HAttachedWindowL,  
                  TRUE,  
                  FALSE,  
                  HAttachedWindowR,  
                  TRUE,  
                  FALSE );
```

## 10.1.6 QCAP\_PLAY\_FILE

### Introduction

The user can use this function to start playing / resume play from pause of a recorded video file.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Start playing a video file on window*

```
QCAP_PLAY_FILE( pFile );
```

---

## 10.1.7 QCAP\_PAUSE\_FILE

### Introduction

The user can use this function to pause a video playing. It is useful when user wants to pause a video from playing,

When user wants to adjust the new video play position, please pause the video first.

To resume video playing, please call **QCAP\_PLAY\_FILE()** again.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To pause a video playing*

```
QCAP_PAUSE_FILE( pFile );  
  
QCAP_PLAY_FILE( pFile );
```

# 10.1.8 QCAP\_STOP\_FILE

## Introduction

The user can use this function to stop a video playing.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To stop a video playing*

```
QCAP_STOP_FILE( pFile );
```

# 10.1.9 QCAP\_DESTROY\_FILE

## Introduction

This function can destroy video playing object and release its system resource.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To destroy a video playing object and release its resources*

```
QCAP_DESTROY_FILE( pFile );
```



## 10.1.10 QCAP\_SET\_FILE\_POSITION

## 10.1.11 QCAP\_SET\_FILE\_POSITION\_EX

### Introduction

The user can use this function to set current video position such as video seeking function.

The **QCAP\_PAUSE\_FILE()** must be call first before setting the new video position. To resume the play please call **QCAP\_PLAY\_FILE()** again.

The **QCAP\_SET\_FILE\_POSITION\_EX()** must be used with:

- **QCAP\_OPEN\_FILE\_EX()**
- **QCAP\_OPEN\_TIMESHIFT\_FILE\_EX()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
double	dSampleTime	IN	Specify the time-stamp of this frame.
ULONG	nTimeUnit	IN	<b>default QCAP_FILE_TIMEUNIT_FRAME</b> Specify Time unit type: QCAP_FILE_TIMEUNIT_TIME QCAP_FILE_TIMEUNIT_FRAME <b>Only in QCAP_SET_FILE_POSITION_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the video position by sample time or frame unit*

```
QCAP_PAUSE_FILE( pFile );

QCAP_SET_FILE_POSITION( pFile, SetSampleTime );

QCAP_PLAY_FILE( pFile );

//for extension version

QCAP_PAUSE_FILE( pFile );

QCAP_SET_FILE_POSITION_EX( pFile, 0, QCAP_FILE_TIMEUNIT_FRAME );

QCAP_PLAY_FILE( pFile );
```

## 10.1.12 QCAP\_GET\_FILE\_POSITION

## 10.1.13 QCAP\_GET\_FILE\_POSITION\_EX

### Introduction

The user can use this function to set current video seeking a position.

The **QCAP\_SET\_FILE\_POSITION\_EX()** must be used with:

- **QCAP\_OPEN\_FILE\_EX()**
- **QCAP\_OPEN\_TIMEShift\_FILE\_EX()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
double *	pSampleTime	OUT	Pointer to the sample time
ULONG	nTimeUnit	IN	<b>default QCAP_FILE_TIMEUNIT_FRAME</b> Specify Time unit type: QCAP_FILE_TIMEUNIT_TIME QCAP_FILE_TIMEUNIT_FRAME <b>Only in QCAP_GET_FILE_POSITION_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get the video position by sample time*

```
QCAP_GET_FILE_POSITION( pFile, &nSampleTime );

QCAP_GET_FILE_POSITION_EX( pFile, &nSampleTime, QCAP_FILE_TIMEUNIT_TIME );
```

# 10.1.14 QCAP\_SET\_FILE\_PLAYBACK\_SPEED

## Introduction

The user can use this function to control file playback speed.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
double	dSpeed	IN	Specify the playback speed, range 0.0 to 4.0

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To set the video playback speed*

```
QCAP_SET_FILE_PLAYBACK_SPEED( pFile, nSpeed );
```

# 10.1.15 QCAP\_GET\_FILE\_PLAYBACK\_SPEED

## Introduction

The user can use this function to get the current video playback speed.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
double *	pSpeed	OUT	Pointer to the speed

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get the video playback speed*

```
QCAP_GET_FILE_PLAYBACK_SPEED( pFile, &nSpeed );
```

### 10.1.16 QCAP\_REFRESH\_TIMESHIFT\_FILE\_INFO

## Introduction

The time-shift function playing a video file while the video is still recording. that's mean the total audio/video frames is increasing while the time-shift video playing. A user can use this function to get current time-shifted video file information during recording.

The `QCAP_OPEN_TIMEShift_FILE_EX()` must be called to use this function.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the file object
double *	pTotalDurationTimes	OUT	Pointer to the total duration times
ULONG *	pTotalVideoFrames	OUT	Pointer to the total video frames
ULONG *	pTotalAudioFrames	OUT	Pointer to the total audio frames
ULONG *	pTotalMetadataFrames	OUT	pointer to total meta-data frames available

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get the run-time information of currently playing time-shift video*

[illegible]

0

# 10.2 Playback SCF Functions

## Introduction

The SCF video file format is originally designed by QCAP SDK framework. It is designed for Scale / Crop video file format used for security applications.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use these functions.

## 10.2.1 QCAP\_SET\_SCF\_FILE\_TIMER

### Introduction

The user can use this function to enable internal timer in a **SCF** channel.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

### Parameters

type	parameter	I/O	descriptions
UINT	iChNum	IN	Specify <b>SCF</b> channel number, start from 0
BOOL	bEnableGlobalTimer	IN	Enable/Disable the global time

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To enable a **SCF** internal timer*

```
QCAP_SET_SCF_FILE_TIMER( 0, FALSE );
```

## 10.2.2 QCAP\_SET\_GLOBAL\_SCF\_FILE\_POSITION

### Introduction

The user can use this function to set global **SCF** file playing position by the sample time.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

### Parameters

type	parameter	I/O	descriptions
double	dSampleTime	IN	Specify the time-stamp of this frame. Set 0 to auto generated by system clock time.

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set global **SCF** file playing position*

```
QCAP_SET_GLOBAL_SCF_FILE_POSITION( nSampleTime );
```

## 10.2.3 QCAP\_GET\_GLOBAL\_SCF\_FILE\_POSITION

### Introduction

The user can use this function to get global **SCF** file playing a position in sample time.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

### Parameters

type	parameter	I/O	descriptions
double *	pSampleTime	OUT	Pointer to the sample time

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get global **SCF** file playing position*

```
QCAP_GET_GLOBAL_SCF_FILE_POSITION( &nSampleTime );
```

## 10.2.4 QCAP\_PLAY\_GLOBAL\_SCF\_FILE

### Introduction

The user can use this function to play a **SCF** videos.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

### Parameters

**VOID**

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To play a **SCF** video*

```
QCAP_PLAY_GLOBAL_SCF_FILE();
```

C

---

## 10.2.5 QCAP\_PAUSE\_GLOBAL\_SCF\_FILE

### Introduction

The user can use this function to pause a **SCF** video from playing.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

### Parameters

**VOID**

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To pause a **SCF** video from playing*

```
QCAP_PAUSE_GLOBAL_SCF_FILE();
```

C

## 10.2.6 QCAP\_STOP\_GLOBAL\_SCF\_FILE

### Introduction

The user can use this function to stop a **SCF** video from playing.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

### Parameters

**VOID**

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To stop a **SCF** video from playing*

```
QCAP_STOP_GLOBAL_SCF_FILE();
```

C



# 10.2.7 QCAP\_SCF\_FILE\_ENUMERATION

## Introduction

The user can use this function to get detailed information from a **SCF** channel, such as video EncoderFormat, FrameRate and audio sample frequency ...etc.

The **QCAP\_OPEN\_SCF\_FILE()** must be called to use this function.

## Parameters

type	parameter	I/O	descriptions
UINT	iChNum	IN	Specify channel number to get <b>SCF</b> file parameters, start from 0
double	dStartSearchTime	IN	Specify the start search time of <b>SCF</b> file
double	dStopSearchTime	IN	Specify the stop search time of <b>SCF</b> file
ULONG *	pFileSizeHigh	OUT	Pointer to the <i>high-order</i> of the file size
ULONG *	pFileSizeLow	OUT	Pointer to the <i>low-order</i> of the file size
double *	pFileStartTime	OUT	Pointer to the file start time
double *	pFileStopTime	OUT	Pointer to the file stop time
double *	pVideoStartTime	OUT	Pointer to the video start time
double *	pVideoStopTime	OUT	Pointer to the video stop time
double *	pAudioStartTime	OUT	Pointer to the audio start time
double *	pAudioStopTime	OUT	Pointer to the audio stop time
ULONG *	pVideoEncoderFormat	OUT	Pointer to the video encoder format
ULONG *	pVideoWidth	OUT	Pointer to the video width
ULONG *	pVideoHeight	OUT	Pointer to the video height
double *	pVideoFrameRate	OUT	Pointer to the video frame rate
ULONG *	pAudioEncoderFormat	OUT	Pointer to the audio encoder format
ULONG *	pAudioChannels	OUT	Pointer to the total audio channels
ULONG *	pAudioBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pAudioSampleFrequency	OUT	Pointer to the audio sample frequency
double *	pTotalDurationTimes	OUT	Pointer to the total duration times
ULONG *	pTotalVideoFrames	OUT	Pointer to the total video frames
ULONG *	pTotalAudioFrames	OUT	Pointer to the total audio frames
ULONG *	pTotalMetadataFrames	OUT	pointer to total meta-data frames available
BOOL	bNext	IN	<b>default FALSE</b> Specifies whether to use number n of the <b>SCF</b> file on the folder. Set FALSE to use the first <b>SCF</b> file on the folder.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get the detail information of a **SCF** video file*

```
QCAP_SCF_FILE_ENUMERATION( 0,
    0, 0,
    NULL, NULL,
    NULL, NULL,
    NULL, NULL,
    NULL, NULL,
    &nVideoFormat,
    &nVideoWidth,
    &nVideoHeight,
    &dVideoFrameRate,
    &nAudioFormat,
    &nAudioChannels,
    &nAudioBitsPerSample,
    &nAudioSampleFrequency,
    &dFileTotalDuationTimes,
    &nFileTotalVideoFrames,
    &nFileTotalAudioFrames,
    TRUE );
```

# 10.3 Playback 3D functions

## Introduction

This section provides functions that can generate 3D video in video playback. The supported 3D video format are:

- Side-by-Side (SBS)
- Top-Bottom (TB)
- Line-by-Line (LBL)

The **QCAP\_OPEN\_3D\_FILE()** must be called to use these functions.

## 10.3.1 QCAP\_SET\_VIDEO\_3D\_FILE\_DISPLAY\_MODE

### Introduction

The user can use this function to set current 3D display format of the video file.

The **QCAP\_OPEN\_3D\_FILE()** must be called to use this function.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nStereoDisplayMode	IN	<b>default QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE</b> Specify 3D display mode: QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE QCAP_3D_STEREO_DISPLAY_MODE_LEFT_ONLY QCAP_3D_STEREO_DISPLAY_MODE_RIGHT_ONLY
BOOL	bLeftRightSwap	IN	<b>default FALSE</b> Swap the the playback video left/right outputs

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the current 3D video file display mode*

```
QCAP_SET_VIDEO_3D_FILE_DISPLAY_MODE( pFile, 0, FALSE );
```

### 10.3.2 QCAP\_GET\_VIDEO\_3D\_FILE\_DISPLAY\_MODE

## Introduction

The user can use this function to get current 3D display format of the video file.

The **QCAP\_OPEN\_3D\_FILE()** must be called to use this function.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pStereoDisplayMode	OUT	Pointer to the current Stereo Display Mode
BOOL *	pLeftRightSwap	OUT	Pointer to the current Left Right Swap

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

### Example : Get the 3D video file display mode

[illegible]

C

# 10.4 Playback Data Functions

## Introduction

The video file is formed by a sequence of frames in time sequences. Each frame consists of audio data, video data, along with user-defined meta-data. When a player opens a video file to playback the video, it's the same idea to travel all the video frames and display it on the windows. This section provides functions to retrieve each frame data inside a video file.

These function descriptions are list as:

1	<b>QCAP_GET_VIDEO_FILE_STREAM_BUFFER()</b>	Extract the video stream data
2	<b>QCAP_GET_AUDIO_FILE_STREAM_BUFFER()</b>	Extract the audio stream data
3	<b>QCAP_GET_METADATA_FILE_DATA_BUFFER()</b>	Extract the user defined meta-data data (optional)

### 10.4.1 QCAP\_GET\_VIDEO\_FILE\_STREAM\_BUFFER

### 10.4.2 QCAP\_GET\_AUDIO\_FILE\_STREAM\_BUFFER

### 10.4.3 QCAP\_GET\_METADATA\_FILE\_DATA\_BUFFER

## Introduction

By given a serial frame number, a user can use this function to random read from any audio / video frames (and meta-data) in a video file. All frames in a video file can be extracted via these functions.

While **channel recording** / **share recording** each frame can send a user-defined information(or meta-data), that could output pre-frames to the video file. Then the meta-data can be retrieved by calling meta-data function for different applications.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iFrameNum	IN	Specify the frame number to get stream
BYTE *	pStreamBuffer	OUT	Pointer to the stream source buffer
ULONG *	pStreamBufferLen	IN/OUT	Specify the length of stream source buffer
BOOL *	pIsKeyFrame	OUT	Pointer to the frame is keyframe or not <b>Only in QCAP_GET_VIDEO_FILE_STREAM_BUFFER()</b>
double *	pSampleTime	OUT	Pointer to the exact sample time of the frame

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To the audio/video framebuffer, meta-data at the 100th video frame*

```
//Retrieve the audio/video stream buffer in a video file

QCAP_GET_VIDEO_FILE_STREAM_BUFFER( pFile,
    100,
    &nStreamBuffer_video,
    &nStreamBufferLen_video,
    &nIsKeyFrame,
    &nSampleTime );

QCAP_GET_AUDIO_FILE_STREAM_BUFFER( pFile,
    100,
    &nStreamBuffer_audio,
    &nStreamBufferLen_audio,
    &nSampleTime );

//Retrieve the first user-defined meta-data in a video file

QCAP_GET_METADATA_FILE_DATA_BUFFER( pFile,
    100,
    nStreamBuffer_meta,
    &nStreamBufferLen_meta,
    &nSampleTime );
```

# 10.4.4 QCAP\_GET\_METADATA\_FILE\_HEADER

## Introduction

This function can get the media information in file header.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
CHAR **	ppszTitle	OUT	The title information in file header
CHAR **	ppszArtist	OUT	The artist information in file header
CHAR **	ppszComments	OUT	The comment information in file header
CHAR **	ppszGenre	OUT	The genre information in file header
CHAR **	ppszComposer	OUT	The composer information in file header

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get the meta-data information from file header.*

```
QCAP_GET_METADATA_FILE_HEADER( pFile,
                                &title, &artist, &comment, &genre, &composer );
```

# 10.5 Playback Video Property Functions

## Introduction

This section provides functions to adjust the color space of video playback buffer, set the clipping region, and flipping the display the video.

### 10.5.1 QCAP\_SET\_VIDEO\_FILE\_REGION\_DISPLAY

#### Introduction

This function can set the clipping area of playback video, the crop region will be scaled to capture device display output.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nCropX	IN	Specify the x-coordinate of the crop region display
ULONG	nCropY	IN	Specify the y-coordinate of the crop region display
ULONG	nCropW	IN	Specify the width of crop region display
ULONG	nCropH	IN	Specify the height of crop region display

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### 10.5.2 QCAP\_GET\_VIDEO\_FILE\_REGION\_DISPLAY

#### Introduction

This function can get the current region clipping area parameters.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pCropX	OUT	Pointer to the x-coordinate of the crop region display
ULONG *	pCropY	OUT	Pointer to the y-coordinate of the crop region display
ULONG *	pCropW	OUT	Pointer to the horizontal width of crop region display
ULONG *	pCropH	OUT	Pointer to the vertical height of crop region display

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set/get crop region x,y(0,0) to w,h(1280x720) to be displayed*

```
QCAP_SET_VIDEO_FILE_REGION_DISPLAY( pFile, 0, 0, 1280, 720 );

QCAP_GET_VIDEO_FILE_REGION_DISPLAY( pFile, &bCropX, &bCropY, &bCropW, &bCropH );
```



# 10.5.3 QCAP\_SET\_VIDEO\_FILE\_MIRROR

## Introduction

This function can set the video flipping mode in both vertical and horizontal way.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
BOOL	bHorizontalMirror	IN	Enable/Disable horizontal of mirror
BOOL	bVerticalMirror	IN	Enable/Disable vertical of mirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Flipping video up-down & left-rig*

```
QCAP_SET_VIDEO_FILE_MIRROR( pFile, TRUE, TRUE );
```

# 10.5.4 QCAP\_GET\_VIDEO\_FILE\_MIRROR

## Introduction

This function can get the flipping mode of a video display.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
BOOL *	pHorizontalMirror	OUT	Pointer to the current horizontal mirror
BOOL *	pVerticalMirror	OUT	Pointer to the current vertical mirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video flipping mode*

```
BOOL bHorizontalMirror, bVerticalMirror;

QCAP_GET_VIDEO_FILE_MIRROR( pFile, &bHorizontalMirror, &bVerticalMirror );
```

# 10.5.5 QCAP\_SET\_VIDEO\_FILE\_BRIGHTNESS

## Introduction

This function can set video brightness value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nValue	IN	Specify the value for brightness, from 0-255

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the video brightness value plus by 20*

```
QCAP_GET_VIDEO_FILE_BRIGHTNESS( pFile, &nValue );

QCAP_SET_VIDEO_FILE_BRIGHTNESS( pFile, nValue + 20 );
```

# 10.5.6 QCAP\_GET\_VIDEO\_FILE\_BRIGHTNESS

## Introduction

This function can set video contrast value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pValue	OUT	Return the current brightness

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video brightness value*

```
QCAP_GET_VIDEO_FILE_BRIGHTNESS( pFile, &nValue );

QCAP_SET_VIDEO_FILE_BRIGHTNESS( pFile, nValue + 20 );
```

# 10.5.7 QCAP\_SET\_VIDEO\_FILE\_CONTRAST

## Introduction

The user can use this function to set video file contrast.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nValue	IN	Specify the value for contrast, from 0-255

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the video contrast value*

```
QCAP_GET_VIDEO_FILE_CONTRAST( pFile, &nValue );

QCAP_SET_VIDEO_FILE_CONTRAST( pFile, nValue + 20 );
```

# 10.5.8 QCAP\_GET\_VIDEO\_FILE\_CONTRAST

## Introduction

This function can get video contrast value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pValue	OUT	Return the current contrast

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video contrast value*

```
QCAP_GET_VIDEO_FILE_CONTRAST( pFile, &nValue );

QCAP_SET_VIDEO_FILE_CONTRAST( pFile, nValue + 20 );
```

# 10.5.9 QCAP\_SET\_VIDEO\_FILE\_HUE

## Introduction

This function can set video hue value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nValue	IN	Specify the value for hue, from 0-255

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the video hue value*

```
QCAP_GET_VIDEO_FILE_HUE( pFile, &nValue );

QCAP_SET_VIDEO_FILE_HUE( pFile, nValue + 20 );
```

# 10.5.10 QCAP\_GET\_VIDEO\_FILE\_HUE

## Introduction

This function can set video hue value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pValue	OUT	Return the current hue

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video hue value*

```
QCAP_GET_VIDEO_FILE_HUE( pFile, &nValue );

QCAP_SET_VIDEO_FILE_HUE( pFile, nValue + 20 );
```

# 10.5.11 QCAP\_SET\_VIDEO\_FILE\_SATURATION

## Introduction

This function can set video saturation value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nValue	IN	Specify the value for saturation, from 0-255

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the video saturation value*

```
QCAP_GET_VIDEO_FILE_SATURATION( pFile, &nValue );

QCAP_SET_VIDEO_FILE_SATURATION( pFile, nValue + 20 );
```

# 10.5.12 QCAP\_GET\_VIDEO\_FILE\_SATURATION

## Introduction

This function can get video saturation value range from 0 to 255. The neutral value is 128, the real value depends on the input video source setting. It is recommended to get the default value before setting a new value.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pValue	OUT	Return the current saturation

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video saturation value*

```
QCAP_GET_VIDEO_FILE_SATURATION( pFile, &nValue );

QCAP_SET_VIDEO_FILE_SATURATION( pFile, nValue + 20 );
```

## 10.6 Playback Audio Functions

### Introduction

This section contains the audio property to set in the device, such as sound renderer can decide which output device to play the sound, and its volume setting in video playback.

### 10.6.1 QCAP\_SET\_AUDIO\_FILE\_SOUND\_RENDERER

#### Introduction

This function can set current sound renderer from available sound output device in video playback.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iSoundNum	IN	Sound Renderer, default Renderer is 0

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : To set the sound renderer for video playback*

```
QCAP_SET_AUDIO_FILE_SOUND_RENDERER( pFile, 1 );
```

### 10.6.2 QCAP\_GET\_AUDIO\_FILE\_SOUND\_RENDERER

#### Introduction

This function can get current sound renderer of sound output device used in video playback.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT *	pSoundNum	OUT	Pointer to the Sound number

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : To get the current sound renderer for video playback*

```
QCAP_GET_AUDIO_FILE_SOUND_RENDERER( pFile, &nSounder );
```

# 10.6.3 QCAP\_SET\_AUDIO\_FILE\_VOLUME

## Introduction

This function can set current audio volume value in video playback, range from 0 to 100.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nVolume	IN	Specify the audio volume value is from 0-100

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To set the sound renderer volume for video playback*

```
QCAP_SET_AUDIO_FILE_VOLUME( pFile, 50 );
```

# 10.6.4 QCAP\_GET\_AUDIO\_FILE\_VOLUME

## Introduction

This function can get current audio volume value in video playback.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG *	pVolume	OUT	Pointer to the audio volume

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get the sound renderer volume for video playback*

```
QCAP_GET_AUDIO_FILE_VOLUME( pFile, nVolume );
```

## 10.7 Playback Snapshot Functions

### Introduction

This chapter will help to take a snapshot of your current playing video. Follow this guide to take a snapshot of your whole video display, or any section of the video you want. The user can save the snapshot to a **BMP/JPG**, or to trigger a snapshot to get the image stream buffer from a callback function without saving it to disk.

### 10.7.1 QCAP\_SNAPSHOT\_FILE\_BMP

### 10.7.2 QCAP\_SNAPSHOT\_FILE\_BMP\_EX

#### Introduction

This function takes a snapshot of video and saves to **BMP** 24bit or 32bit file in video playback.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_BMP\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.BMP24" → save to 24bit <b>BMP</b> "Filename.BMP32 or BMP" → save to 32bit <b>BMP</b> "BMP24" → To snapshot stream in callback only (no save to file.) "BMP32"/"BMP" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_FILE_BMP_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_FILE_BMP_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_FILE_BMP_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_FILE_BMP_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_FILE_BMP_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_FILE_BMP_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.



### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Take a snapshot and save to file in video playback*

```
QCAP_SNAPSHOT_FILE_BMP( pFile,  
                        "C:/PICTURE1.BMP24" );  
  
QCAP_SNAPSHOT_FILE_BMP_EX( pFile,  
                           "C:/PICTURE1.BMP",  
                           10, 40,  
                           1900, 1000, 720, 480 );
```

### 10.7.3 QCAP\_SNAPSHOT\_FILE\_JPG

### 10.7.4 QCAP\_SNAPSHOT\_FILE\_JPG\_EX

#### Introduction

This function takes a snapshot of video and saves to **JPEG** image file format in video playback.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_JPG\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.JPG" → To snapshot to <b>JPEG</b> image file "JPG" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_FILE_JPG_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_FILE_JPG_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_FILE_JPG_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_FILE_JPG_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_FILE_JPG_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_FILE_JPG_EX()</b>
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Take a snapshot in video playback, cropping a rectangle area and scale to 720x480 in **JPEG***

```
QCAP_SNAPSHOT_FILE_JPG( pFile,  
    "C:/PICTURE1.JPG",  
    100 );  
  
QCAP_SNAPSHOT_FILE_JPG_EX( pFile,  
    "C:/PICTURE1.JPG",  
    10, 40,  
    1900, 1000,  
    720, 480,  
    100 );
```

## 10.8 Playback OSD Functions

### Introduction

An on-screen display (OSD) are control functions on a video screen that allows you to draw text fields, overlapped pictures, or put customer image buffer with blending or color key effect.

For more detailed parameters descriptions, please refer to **Chapter 7 OSD Function API**.

### 10.8.1 QCAP\_SET\_OSD\_FILE\_TEXT

### 10.8.2 QCAP\_SET\_OSD\_FILE\_TEXT\_EX

### 10.8.3 QCAP\_SET\_OSD\_FILE\_TEXT\_W

### 10.8.4 QCAP\_SET\_OSD\_FILE\_TEXT\_EX\_W

### Introduction

The user can use this function to create a text field objects used for on-screen display.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_TEXT()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to auto calculate width.
INT	h	IN	Specify the height of OSD output Set -1 to auto calculate height.
CHAR * WSTRING	pszString pwszString	IN	Specify to display the text of OSD output Support wide character string
CHAR * WSTRING	pszFontFamilyName pwszFontFamilyName	IN	Specify the font name used to display the text of OSD output Support wide character string
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output Available values are QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output

DWORD	dwFontColor	IN	Specify the font color used to display the text of OSD output
DWORD	dwBackgroundColor	IN	Specify the background color used to display the text of OSD output
DWORD	dwBorderColor	IN	Specify the border color of the text <b>Only in QCAP_SET_OSD_FILE_TEXT_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width in pixel, set 0 to disable border. <b>Only in QCAP_SET_OSD_FILE_TEXT_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
INT	nTextStartPosX	IN	<b>default 0</b> Specify the text scrolling start position X of the upper-left corner of OSD text
INT	nTextStartPosY	IN	<b>default 0</b> Specify the text scrolling start position Y of the upper-left corner of OSD text
ULONG	nStringAlignmentStyle	IN	<b>default QCAP_STRING_ALIGNMENT_STYLE_LEFT</b> The alignment styles are: QCAP_STRING_ALIGNMENT_STYLE_LEFT QCAP_STRING_ALIGNMENT_STYLE_NEAR QCAP_STRING_ALIGNMENT_STYLE_CENTER QCAP_STRING_ALIGNMENT_STYLE_RIGHT QCAP_STRING_ALIGNMENT_STYLE_FAR <b>Only in QCAP_SET_OSD_FILE_TEXT_EX()</b>
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Put a string "CH01" on the top of video playback*

```

QCAP_SET_OSD_FILE_TEXT( pFile, 0,
    0, 0,
    -1, -1,
    "CH01", "Arial",
    QCAP_FONT_STYLE_BOLD, 12,
    0xFF000000,
    0xFFFFFFFF,
    128, 0, 0,
    QCAP_SEQUENCE_STYLE_FOREMOST );

QCAP_SET_OSD_FILE_TEXT_EX( pFile, 0,
    0, 0,
    -1, -1,
    "CH01", "Arial",
    QCAP_FONT_STYLE_BOLD, 12,
    0xFF000000,
    0xFFFFFFFF,
    0,0, //border color & width
    128, 0, 0,
    QCAP_STRING_ALIGNMENT_STYLE_LEFT,
    QCAP_SEQUENCE_STYLE_FOREMOST );

```

# 10.8.5 QCAP\_GET\_OSD\_FILE\_TEXT\_BOUNDARY

# 10.8.6 QCAP\_GET\_OSD\_FILE\_TEXT\_BOUNDARY\_W

## Introduction

The user can use this function to get size of OSD string on a playback video.



For more detailed parameters descriptions, please refer to **QCAP\_GET\_OSD\_TEXT\_BOUNDARY()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
CHAR * <a href="#">WSTRING</a>	pszString <a href="#">pwszString</a>	IN	Specify to display the text of OSD output <a href="#">Support wide character string</a>
CHAR * <a href="#">WSTRING</a>	pszFontFamilyName <a href="#">pwszFontFamilyName</a>	IN	Specify the font name used to display the text of OSD output <a href="#">Support wide character string</a>
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output Available values are QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
ULONG *	pBoundaryWidth	OUT	Pointer to the boundary width
ULONG *	pBoundaryHeight	OUT	Pointer to the boundary height

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the boundary width/height of the OSD text in video playback*

```
QCAP_GET_OSD_FILE_TEXT_BOUNDARY( pFile,
                                0,
                                "CH01",
                                "Arial",
                                QCAP_FONT_STYLE_BOLD,
                                12,
                                &BoundaryWidth,
                                &BoundaryHeight );
```

# 10.8.7 QCAP\_SET\_OSD\_FILE\_PICTURE

## Introduction

This OSD function displays one or more **BMP/JPG/PNG/GIF/EDL.INI** on top of playback video.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_PICTURE()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output. Set -1 to use original picture width.
INT	h	IN	Specify the height of OSD output. Set -1 to use original picture height.
CHAR *	pszFilePathName	IN	Specify the image file name to display in OSD Supported extensions: " <b>BMP</b> " as 24/32-bit, " <b>JPG</b> " and " <b>PNG</b> " Supported animation by <b>GIF,EDL.INI</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a half-transparent PNG on the top of captured video stream in video playback*

```
QCAP_SET_OSD_FILE_PICTURE( pFile,
    0,
    0, 0,
    -1, -1,
    "C:/SAMPLE.PNG",
    128,
    QCAP_SEQUENCE_STYLE_FOREMOST );
```



## 10.8.8 QCAP\_SET\_OSD\_FILE\_BUFFER

## 10.8.8 QCAP\_SET\_OSD\_FILE\_BUFFER\_EX

### Introduction

The user can use this function to create a framebuffer object used for on-screen display.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_BUFFER()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output
INT	h	IN	Specify the height of OSD output
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameWidth	IN	Specify the width of frame contained in a buffer
ULONG	nFrameHeight	IN	Specify the height of frame contained in a buffer
ULONG	nFramePitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate by width and color space format.
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_FILE_BUFFER_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_FILE_BUFFER_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SET_OSD_FILE_BUFFER_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SET_OSD_FILE_BUFFER_EX()</b>
DWORD	dwBorderColor	IN	Specify the border color <b>Only in QCAP_SET_OSD_FILE_BUFFER_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width <b>Only in QCAP_SET_OSD_FILE_BUFFER_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	

			<b>default 0xFFFFFFFF</b> Specify the key color of broadcast server OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY) 2. 0x00FF0000 (MASK BLUE) 3. 0x0000FF00 (MASK GREEN)
ULONG	nKeyColorThreshold	IN	<b>default 25</b> Specify the threshold of key color, the range from 0 to 128
ULONG	nKeyColorBlurLevel	IN	<b>default 2</b> Specify the blur level, range 0-2
BOOL	bKeyColorSpillSuppress	IN	<b>default TRUE</b> Specify the color spill suppress value
ULONG	nKeyColorSpillSuppressThreshold	IN	<b>default 22</b> Specify the threshold value of color spill suppress
BYTE *	pMaskBuffer	IN	<b>default NULL</b> Specify a mask OSD background buffer, default null
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Place a picture directly from a framebuffer on the video display in video playback*

```
QCAP_SET_OSD_FILE_BUFFER( pFile,  
    0,  
    0, 0, 1920, 1080,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    pFrameBuffer,  
    800, 600,  
    0,  
    128,  
    0xFFFFFFFF,  
    25, 2,  
    NULL,  
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

```
QCAP_SET_OSD_FILE_BUFFER_EX( pFile,  
    0,  
    0, 0, 1920, 1080,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    pFrameBuffer,  
    800, 600,  
    0,  
    0,0,0,0,  
    128,  
    0xFFFFFFFF,  
    25, 2, FALSE,  
    NULL,  
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

### 10.8.9 QCAP\_MOVE\_OSD\_FILE\_OBJECT

## Introduction

The user can use this function to move the OSD object around the video window. It is useful to scroll the text string or picture on the playback video window.



For more detailed parameters descriptions, please refer to **QCAP\_MOVE\_OSD\_OBJECT()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To scroll the OSD object from left to right in playback video*

[illegible]

## 10.9 Playback Merge Functions

### Introduction

This section provides functions lets you concatenate two or more video files into one longer video file. This is needed when many small split videos need to be merged. The exporting ensures that the trimmed clip is saved as a separate new video rather than overwriting the existing video. If there are tons of video files needs to be the merge, the user describes it by **EDL** INI script and call **QCAP\_MERGE\_FILES\_BY\_EDL()** to merge them all at once.

### 10.9.1 QCAP\_MERGE\_FILES

### 10.9.2 QCAP\_MERGE\_FILES\_EX

### 10.9.3 QCAP\_MERGE\_FILES\_EX\_C

### Introduction

These three functions can let user merges two or more video clips into on big video file. The resolution of video files must be the same, for example, a user cannot combine video resolutions: 1920x1080 and 1280x720 together. Currently only support for **MP4** video format.

The description of different functions are list as:

#	API	descriptions
1	<b>QCAP_MERGE_FILES()</b>	To merge two video file, save to one video file
2	<b>QCAP_MERGE_FILES_EX()</b>	To merge more video file in variable arguments, save to one video file
3	<b>QCAP_MERGE_FILES_EX_C()</b>	To merge more video file in an array, save to one video file

### Parameters

**QCAP\_MERGE\_FILES()** parameter

type	parameter	I/O	descriptions
CHAR *	pszFrontEndFileName	IN	Specify the front-end file name to merge
CHAR *	pszBackEndFileName	IN	Specify the back-end file name to merge
CHAR *	pszMergedFileName	IN	Specify the merged result file name Supported extensions: " <b>MP4</b> "

**QCAP\_MERGE\_FILES\_EX()** parameter

type	parameter	I/O	descriptions
CHAR *	pszMergedFileName	IN	Specify the merged result file name Supported extensions: " <b>MP4</b> "
ULONG	nFileArgs	IN	Specify the file number to merge
CHAR *	pszSourceFileName1	IN	Specify the 1st file name to merge Supported extensions: " <b>MP4</b> "
CHAR *	pszSourceFileName2	IN	Specify the 2nd file name to merge Supported extensions: " <b>MP4</b> "
...	...	...	

			<b>Variable arguments</b> (to support for 3rd, 4th, 5th, 6th... files etc.)
--	--	--	--

QCAP\_MERGE\_FILES\_EX\_C() parameter

type	parameter	I/O	descriptions
CHAR *	pszMergedFileName	IN	Specify the merged result file name Supported extensions: <b>"MP4"</b>
ULONG	nFileArgs	IN	Specify the files number to merge
CHAR **	pszSourceFileNames	IN	Specify a list of file names to merge Supported extensions: <b>"MP4"</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To merge multiple small video files into one video file*

```
//To merge two files

QCAP_MERGE_FILES( "C:/1.MP4", "C:/2.MP4", "C:/MERGE_OUTPUT.MP4" );


//To merge file in arguments

QCAP_MERGE_FILES_EX( "MERGE.MP4", 4,
                    "1.MP4", "2.MP4", "3.MP4", "4.MP4" );


//To merge file in a array

char *filenames[4]= {"1.MP4", "2.MP4", "3.MP4", "4.MP4"};

QCAP_MERGE_FILES_EX_C( "MERGE.MP4", 4, filenames );
```

# 10.9.4 QCAP\_MERGE\_FILES\_BY\_EDL

## Introduction

The is function provide a merge files function that using an **EDL** INI script to describe the source video files. The user also can specify the start/stop time for each video file before merging. The result will save to an output video file. The resolution of video files must be the same, for example, a user cannot combine video resolutions: 1920x1080 and 1280x720 together.

*An example of MERGE.EDL.EXAMPLE.INI*

```
[OUTLINE]
SEGMENT.COUNT = 4
OUTPUT.FILE.NAME = D:\TEST\OUT.MP4

[SEGMENT.0]
SOURCE.FILE.NAME = D:\TEST\IN_0.MP4
START.TIME = 10
STOP.TIME = 30

[SEGMENT.1]
SOURCE.FILE.NAME = D:\TEST\IN_1.MP4
START.TIME = 20
STOP.TIME = 50

[SEGMENT.2]
SOURCE.FILE.NAME = D:\TEST\IN_0.MP4
START.TIME = 50
STOP.TIME = 60

[SEGMENT.3]
SOURCE.FILE.NAME = D:\TEST\IN_1.MP4
START.TIME = 30
STOP.TIME = 80
```

## Parameters

type	parameter	I/O	descriptions
CHAR *	pszEDLFileName	IN	Specify a <b>EDL</b> INI file that describes: * <b>SOURCE.FILE.NAME</b> : The source video file names * <b>SEGMENT.COUNT</b> : number of file to merge * <b>START.TIME</b> : the start time of video clip * <b>STOP.TIME</b> : the stop time of video clip * <b>OUTPUT.FILE.NAME</b> : the output video file name

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To merge files described by an EDL script in QCAP\_MERGE\_FILES\_BY\_EDL()*

```
char *ini = "MERGE.EDL.EXAMPLE.INI";

QCAP_MERGE_FILES_BY_EDL( ini );
```

# 10.10 Playback Export Functions

## Introduction

This section provides functions lets you shorten the total length of the video clip by trimming it down to a shorter length. This is perfect for eliminating needless parts of a video. The exporting ensures that the trimmed clip is saved as a separate new video rather than overwriting the existing video.

### 10.10.1 QCAP\_EXPORT\_FILE

### 10.10.2 QCAP\_EXPORT\_FILE\_EX

### 10.10.3 QCAP\_EXPORT\_FILE\_EX\_C

## Introduction

This section provides functions to export your video clip to a *single* video file by different beginning and end of each clip. If the user wants to export to multiple files please refer to **QCAP\_EXPORT\_FILES\_EX()**. Currently supported vide formats are: **MP4**, **TS**, **FLV**. The description of different functions are list as:

#	API	descriptions
1	<b>QCAP_EXPORT_FILE()</b>	To trim a video clip by a start/stop position and save to another file
2	<b>QCAP_EXPORT_FILE_EX()</b>	To trim a video clip by multiple start/stop positions and save to another file
3	<b>QCAP_EXPORT_FILE_EX_C()</b>	To trim a video clip by multiple start/stop positions in an array and save to another file

## Parameters

### QCAP\_EXPORT\_FILE() Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
double	dStartSampleTime	IN	Specify the start sample time in seconds
double	dStopSampleTime	IN	Specify the stop sample time in seconds
CHAR *	pszExportedFileName	IN	Specify the file name for the exported file Supported extensions: <b>MP4</b> , <b>TS</b> , <b>FLV</b>

### QCAP\_EXPORT\_FILE\_EX() Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
CHAR *	pszExportedFileName	IN	Specify the file name for the exported file Supported extensions: <b>MP4</b> , <b>TS</b> , <b>FLV</b>
ULONG	nFileArgs	IN	Specify the file number to merge
double	dStartSampleTime1	IN	Specify the 1st start sample time in seconds
double	dStopSampleTime1	IN	Specify the 1st stop sample time in seconds
double	dStartSampleTime2	IN	Specify the 2nd start sample time in seconds



double	dStopSampleTime2	IN	Specify the 2nd stop sample time in seconds
...	...	...	<b>Variable arguments</b> (to support for 3rd, 4th, 5th, 6th... sample times etc.)

QCAP\_EXPORT\_FILE\_EX\_C() Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
CHAR *	pszExportedFileName	IN	Specify the file name for the exported file Supported extensions: <b>MP4, TS, FLV</b>
ULONG	nFileArgs	IN	Specify the file number to merge
double *	dStartSampleTimes	IN	Specify a list of start sample times in seconds
double *	dStopSampleTimes	IN	Specify a list of stop sample times in seconds

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

Example 1: Export a video to another file by different begin/end positions

```
QCAP_EXPORT_FILE( pFile,
    5.000,
    10.000,
    "C:/EXPORT1.MP4" );

QCAP_EXPORT_FILE_EX( pFile,
    "C:/EXPORT2.MP4",
    2,
    5.000, 10.000,
    15.000, 20.000, );
```

Example 2: Export a video to another file by 4 different begin/end positions

```
double start_sample_times[4] = { 5.0, 15.0, 25.0, 35.0 };

double stop_sample_times[4] = {10.0, 20.0, 30.0, 40.0 };

QCAP_EXPORT_FILE( pFile,
    "C:/EXPORT3.MP4",
    4,
    start_sample_times,
    stop_sample_times );
```

# 10.10.4 QCAP\_EXPORT\_FILES\_EX

# 10.10.5 QCAP\_EXPORT\_FILES\_EX\_C

## Introduction

This section provides functions to export your video clip to *multiple* video files by different beginning and end of each clip. Currently supported extensions are: **MP4, TS, FLV**. The description of different functions are list as:

#	API	descriptions
1	QCAP_EXPORT_FILES_EX()	To trim a video clip by multiple start/stop positions, save each to different files
2	QCAP_EXPORT_FILES_EX_C()	To trim a video clip by multiple start/stop positions in an array, save each to different files

## Parameters

### QCAP\_EXPORT\_FILES\_EX() Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nFileArgs	IN	Specify the file number to merge
double	dStartSampleTime1	IN	Specify the 1st start sample time in seconds
double	dStopSampleTime1	IN	Specify the 1st stop sample time in seconds
CHAR *	pszExportedFileName1	IN	Specify the 1st file name for the exported file Supported extensions: <b>MP4, TS, FLV</b>
double	dStartSampleTime2	IN	Specify the start 2nd sample time in seconds
double	dStopSampleTime2	IN	Specify the stop 2nd sample time in seconds
CHAR *	pszExportedFileName2	IN	Specify the 2nd file name for the exported file Supported extensions: <b>MP4, TS, FLV</b>
...	...	...	<b>Variable arguments</b> (to support for 3rd, 4th, 5th, 6th... sample times etc.)

### QCAP\_EXPORT\_FILES\_EX\_C() Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the filer object
ULONG	nFileArgs	IN	Specify the file number to merge
double *	dStartSampleTimes	IN	Specify a list of start sample times in seconds
double *	dStopSampleTimes	IN	Specify a list of stop sample times in seconds
CHAR **	pszExportedFileNames	IN	Specify a list of file names for the exported files Supported extensions: <b>MP4, TS, FLV</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example 1: Export a video clip to 2 videos files by different begin/end positions*

```
QCAP_EXPORT_FILES_EX( pFile,  
    2,  
    5.000, 10.000, "C:/EXPORT_1.MP4",  
    15.000, 20.000, "C:/EXPORT_2.MP4" );
```

*Example 2: Export a video clip to 4 videos files by different begin/end positions*

```
double start_sample_times[4] = { 5.0, 15.0, 25.0, 35.0 };  
  
double stop_sample_times[4] = {10.0, 20.0, 30.0, 40.0 };  
  
char filenames[4]= {"1.MP4", "2.MP4", "3.MP4", "4.MP4"};  
  
QCAP_EXPORT_FILES_EX_C( pFile,  
    4,  
    start_sample_times,  
    stop_sample_times,  
    filenames );
```

# 10.11 Playback Other Functions

## Introduction

This section provides functions lets you diagnostic/repair video files that recorded by QCAP SDK and build Video-on-Demand file from a recorded file.

### 10.11.1 QCAP\_BUILD\_VIDEO\_ON\_DEMAND\_FILE

#### Introduction

The user can use this function to convert the video recording **MP4** to one **VOD** (Video on Demand) file.

#### Parameters

type	parameter	I/O	descriptions
CHAR *	pszOriginalFileName	IN	Specify the original video file name
CHAR *	pszVideoOnDemandFileName	IN	Specify the on demand video file name

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Convert video file **MP4** to a video on demand file*

```
QCAP_BUILD_VIDEO_ONDEMAND_FILE( "File.MP4", "VOD.MP4" );
```

# 10.11.2 QCAP\_DIAGNOSE\_FILE

## Introduction

This function can help a user to check the integrity of a video file of a video file recorded by QCAP SDK.

## Parameters

type	parameter	I/O	descriptions
CHAR *	pszFileName	IN	Specify the video file name to display video output
BOOL *	plsHealthy	OUT	Pointer to the video healthy condition

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Check the video file structure to see if any problems*

```
QCAP_DIAGNOSE_FILE( "File.MP4", &Ishealthy );
```

# 10.11.3 QCAP\_REPAIR\_FILE

## Introduction

If a recorded video file has some error when playback, this function can help a user to repair a video file recorded by QCAP SDK.



Please refer to **QCAP\_SET\_SYSTEM\_CONFIGURATION()**, the repair function required the *bEnableFileRepairFunction* parameter set to true.

## Parameters

type	parameter	I/O	descriptions
CHAR *	pszBadFileName	IN	Specify the file name Supported extensions: <b>"MP4"</b>
CHAR *	pszRepairedFileName	IN	Specify the first repaired file name Supported extensions: <b>"MP4"</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

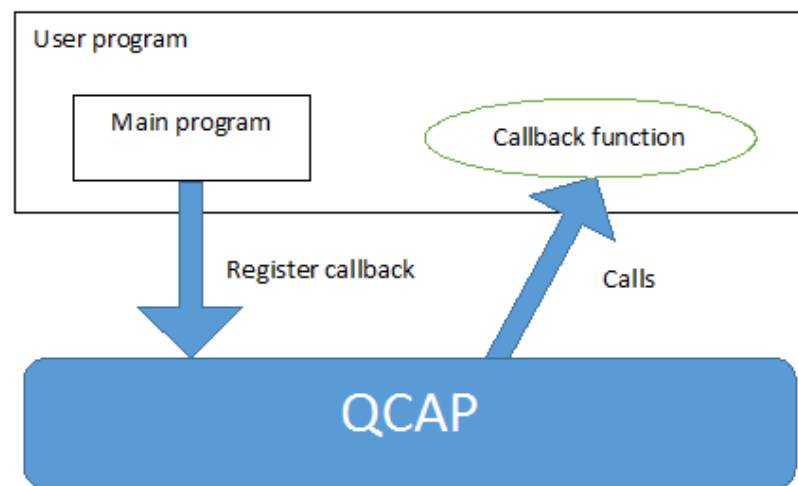
## Examples

*Example : Repair a bad video file recorded by QCAP SDK*

```
QCAP_REPAIR_FILE( "BadFile.MP4", "Repaired.MP4" );
```

# 10.12 Playback Callback Functions

## Introduction



This callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_FILE_SNAPSHOT_DONE_CALLBACK	PF_FILE_SNAPSHOT_DONE_CALLBACK
QCAP_REGISTER_FILE_SNAPSHOT_STREAM_CALLBACK	PF_FILE_SNAPSHOT_STREAM_CALLBACK
QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK	PF_VIDEO_DECODER_FILE_CALLBACK
QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK_EX	PF_VIDEO_DECODER_FILE_CALLBACK_EX
QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK	PF_AUDIO_DECODER_FILE_CALLBACK
QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK_EX	PF_AUDIO_DECODER_FILE_CALLBACK_EX
QCAP_REGISTER_VIDEO_DECODER_3D_FILE_CALLBACK	PF_VIDEO_DECODER_3D_FILE_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

### 10.12.1 QCAP\_REGISTER\_FILE\_SNAPSHOT\_DONE\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_FILE\_BMP/JPG()** is completed. The user can get the path filename of the snapshot in the callback.

When uses asynchronous snapshot (*blsAsync* = TRUE), it is useful to know when snapshot file is ready.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the file object
<b><i>PF_FILE_SNAPSHOT_DONE_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**PF FILE SNAPSHOT DONE CALLBACK**

### Parameters of Callback

<i><b>type</b></i>	<i><b>parameter</b></i>	<i><b>callback descriptions</b></i>
PVOID	pFile	Handle of the file object
CHAR *	pszFilePathName	pointer to the snapshot filename
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP RT OK** or **QCAP RT FAIL** after callback processed.

## Examples

*Example : Register a callback function to get snapshot completion*

[illegible]

### 10.12.2 QCAP\_REGISTER\_FILE\_SNAPSHOT\_STREAM\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_FILE\_BMP/JPG()** image stream is generated, then user can get image stream in buffer directly.

Fo. The users who want a snapshot without saving to a file, call **QCAP\_SNAPSHOT\_FILE\_BMP/JPG()** by passing *pszFilePathName* file extension only(e.g. "BMP"), then a user can use this callback to retrieve the snapshot image in the buffer.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the file object
<b><i>PF_FILE_SNAPSHOT_STREAM_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF FILE SNAPSHOT STREAM CALLBACK

### Parameters of Callback

<b>type</b>	<b>parameter</b>	<b>callback descriptions</b>
PVOID	pFile	Handle of the file object
CHAR *	pszFilePathName	pointer to the snapshot filename
BYTE *	pStreamBuffer	Pointer to the image framebuffer
ULONG	nStreamBufferLen	Specify the length of image framebuffer
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

Returns **QCAP RT OK** or **QCAP RT FAIL** after callback processed.

## Examples

*Example : Register a callback function to get the snapshot stream*

[illegible]



### 10.12.3 QCAP\_REGISTER\_VIDEO\_DECODER\_FILE\_CALLBACK

### 10.12.4 QCAP\_REGISTER\_VIDEO\_DECODER\_FILE\_CALLBACK\_EX

#### Introduction

This callback function will provide video uncompressed data **after** video decoding for the developer to use.

The **QCAP\_REGISTER\_VIDEO\_DECODER\_FILE\_CALLBACK\_EX()** will provide current frame number and video uncompressed data **after** video decoding. and must be used with:

- **QCAP\_OPEN\_FILE\_EX()**
- **QCAP\_OPEN\_TIMEShift\_FILE\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the file object
<b>PF_VIDEO_DECODER_FILE_CALLBACK+</b>	pCB	IN	Callback function <b>Only in</b> <b>QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK()</b>
<b>PF_VIDEO_DECODER_FILE_CALLBACK_EX</b>	pCB	IN	Callback function <b>Only in</b> <b>QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK_EX()</b>
PVOID	pUserData	IN	Pointer to custom user data

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_VIDEO\_DECODER\_FILE\_CALLBACK

### PF\_VIDEO\_DECODER\_FILE\_CALLBACK\_EX

#### Parameters of Callback

type	parameter	callback descriptions
PVOID	pFile	Handle of the file object
UINT	iFrameNum	Specify the current frame number
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer's size
PVOID	pUserData	Pointer to custom user data

#### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and will not be displayed on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	The video uncompressed data will not be played on the window

#### Examples

*Example : Register a callback function to get video uncompressed data after decoding.*

[illegible]

# 10.12.5 QCAP\_REGISTER\_AUDIO\_DECODER\_FILE\_CALLBACK

# 10.12.6 QCAP\_REGISTER\_AUDIO\_DECODER\_FILE\_CALLBACK\_EX

## Introduction

This callback function will provide audio uncompressed data *after* audio decoding for the developer to use.

The `QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK_EX()` will provide current frame number and video uncompressed data *after* video decoding. and must be used with:

- `QCAP_OPEN_FILE_EX()`
- `QCAP_OPEN_TIMEShift_FILE_EX()`.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the file object
<i>PF_AUDIO_DECODER_FILE_CALLBACK</i>	pCB	IN	Callback function Only in <code>QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK()</code>
<i>PF_AUDIO_DECODER_FILE_CALLBACK_EX</i>	pCB	IN	Callback function Only in <code>QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK_EX()</code>
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

## PF\_AUDIO\_DECODER\_FILE\_CALLBACK

## PF\_AUDIO\_DECODER\_FILE\_CALLBACK\_EX

### Parameters of Callback

type	parameter	callback descriptions
PVOID	pFile	Handle of the file object
UINT	iFrameNum	Specify the current frame number
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer's size
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<code>QCAP_RT_OK</code>	callback already processed
<code>QCAP_RT_FAIL</code>	the framebuffer will be dropped and its sound will be muted
<code>QCAP_RT_SKIP_DISPLAY</code>	the audio uncompressed data will not be played on the window

## Examples

Example : Register a callback function to get audio uncompressed data after decoding.

[illegible]

# 10.12.7 QCAP\_REGISTER\_VIDEO\_DECODER\_3D\_FILE\_CALLBACK

## Introduction

This callback function will provide video recording’s 3D uncompressed data ***after*** video decoding in file playback.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFile	IN	Handle of the file object
<i><b>PF_VIDEO_DECODER_3D_FILE_CALLBACK</b></i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# ***PF\_VIDEO\_DECODER\_3D\_FILE\_CALLBACK***

## *Parameters of Callback*

<i><b>type</b></i>	<i><b>parameter</b></i>	<i><b>callback descriptions</b></i>
PVOID	pFile	Handle of the file object
UINT	iChNum	The channel number to get <b>SCF</b> file parameters, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer's size
PVOID	pUserData	Pointer to custom user data

## *Return value of Callback*

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and will not be displayed on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	The video uncompressed data will not be played on the window

## Examples

*Example : Register a callback function to get 3D video uncompressed data after decoding.*

[illegible]

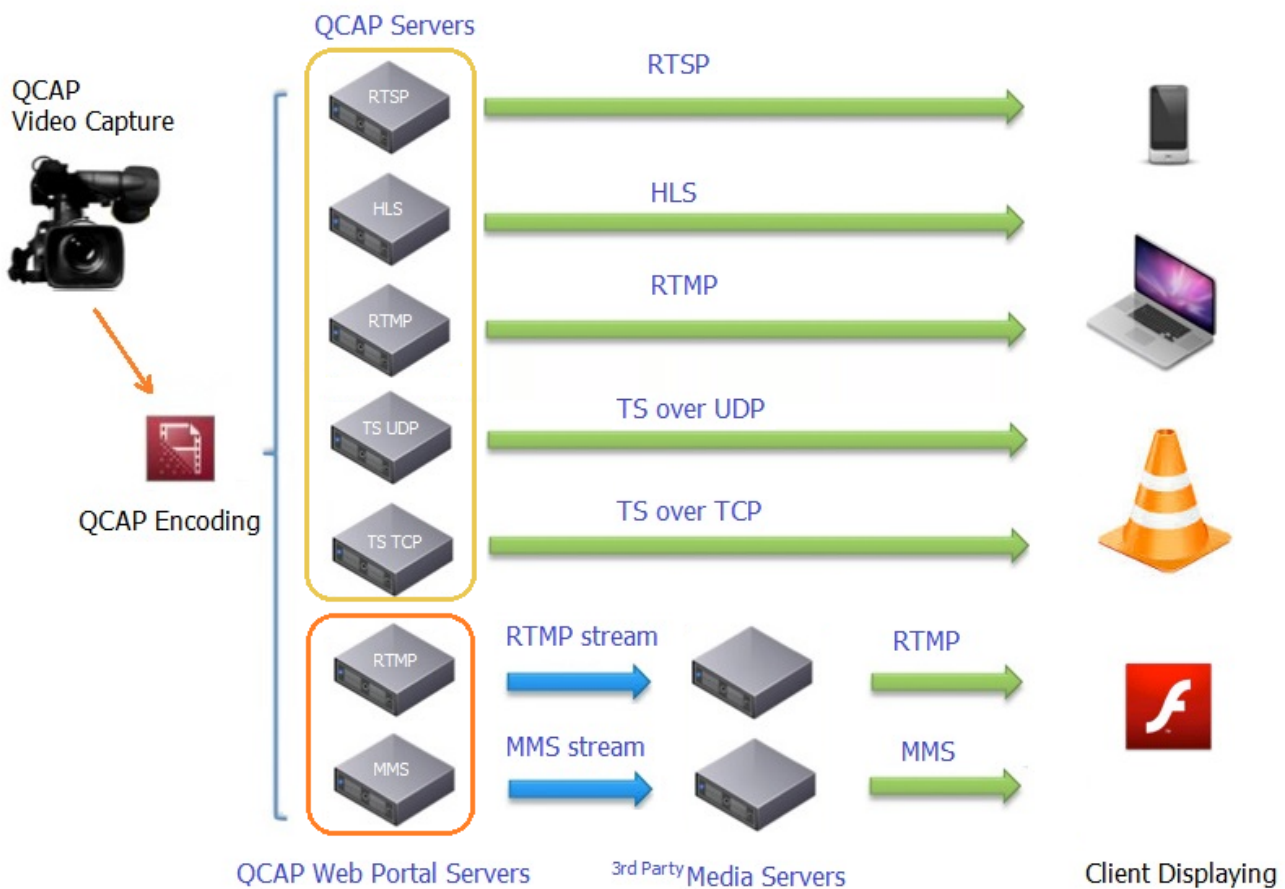
# 11 Broadcasting Server & Client Function API

## Introduction

In this chapter consists two parts of a chapter: Server Broadcast functions and Client Broadcast functions. You will understand **The Delay Live Broadcasting Feature** supported by QCAP SDK, how to design a streaming server for **RTSP, RTMP, HLS, TS over TCP/UDP, MMS**. By using QCAP broadcasting APIs. These protocols can be embedded into your software right away and enabled the broadcasting capability to the world.

## 11.1 Broadcasting Overview

### 11.1.1 Broadcasting Introduction



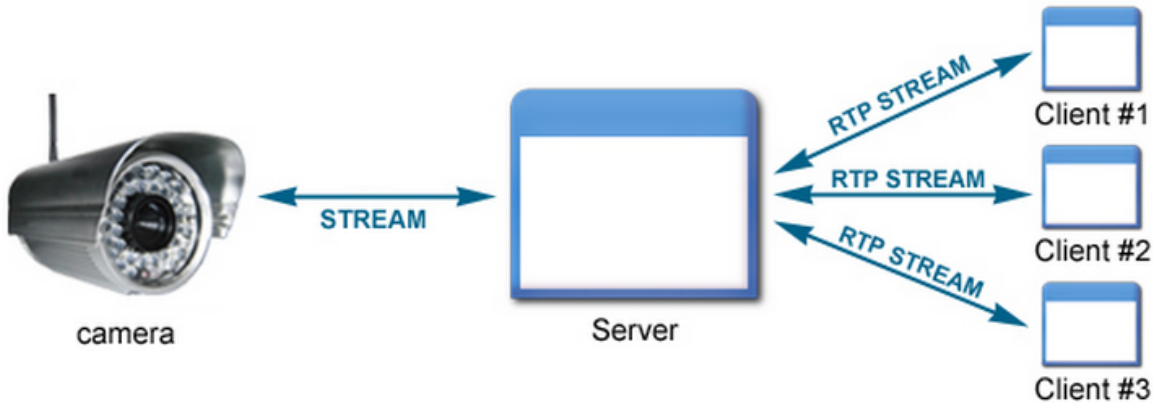
This chapter provides functions can broadcast your audio/video streams from a server, or receiving / displaying the live stream in a client. Those APIs can be used to provide broadcasting experience for a capture card by using QCAP API broadcasting for both server/client applications.

It support most popular streaming technologies: **RTSP, RTMP, MMS, HLS, TS over TCP/UDP**. Those protocols are designed for use in streaming media systems which allows a client to remotely control a streaming media server, issuing VCR-like commands such as "play" and "pause", and allowing time-based access to files on a server. The typical customers of this solution are media content providers that need networking capability to broadcast on their content.

The server/client protocol supporting list are:

protocol	Server-side	Client-side
RTSP	Supported	Supported
HLS	Supported	Not-Supported
RTMP	Supported	Supported <i>(built with QCAP SDK)</i>
MMS	Supported	Not-Supported
TS over TCP	Supported	Supported
TS over UDP	Supported	Supported

### 11.1.2 Broadcasting Server API



#### The Delay Live Broadcasting Feature

The is a **Delay Live Feature** that provides *Network Delay Live Streaming* during the broadcasting for Content-based regulation. In QCAP delay live buffer, the live broadcasting audio/video will be delayed/buffered a certain duration and stored in the caches of buffer before the live stream content is output to the client. This way the user can verify the content of the audio/video and decides what content must be cut off. It is useful when a live video needs to be preprocessed and follow authority policies.



For the **Delay Live Broadcasting Feature**, please refer to [Server Delay Live Streaming Functions](#) section.

#### Broadcast Server Flow

Steps	Operations	API calls
1	Create Broadcast Server Object	QCAP_CREATE_BROADCAST_RTSP_SERVER( ..., &pServer )
2	Register Callback Function	QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK( pServer, ... ) QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK( pServer, ... )
3	Set Video / Audio Property	QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer, ... ) QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY( pServer, ... )
4	StartVideo / Audio Broadcast Server	QCAP_START_BROADCAST_SERVER( pServer )
5	Push Uncompressed/Compressed Buffer	QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, ... ) QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, ... ) QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer, ... ) QCAP_SET_AUDIO_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer, ... )
6	Stop Video / Audio Broadcast Server	QCAP_STOP_BROADCAST_SERVER( pServer )
7	Delete BroadcastServer Object	QCAP_DESTROY_BROADCAST_SERVER( pServer )



To utility different streaming protocols, all you have to do is change their creation function, the rest of programming are the same among different protocols in QCAP SDK. For example, if you had already used **QCAP\_CREATE\_BROADCAST\_RTSP\_SERVER()** to create a **RTSP** server, you can change it to **QCAP\_CREATE\_BROADCAST\_HLS\_SERVER()** to enable **HLS** server by modifying just one function. The rest of programming steps are identical as **RTSP** streaming program and no change is required.

### 11.1.3 Broadcasting Client API

#### Broadcast Client Flow

Steps	Operations	API calls
1	Create Broadcast Client Object	QCAP_CREATE_BROADCAST_CLIENT ( ..., &pClient, ... )
2	Register Callback Function	QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK( pClient, ... ) QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK( pClient, ... )
3	Start Video / Audio Broadcast Client	QCAP_START_BROADCAST_CLIENT( pClient )
4	Stop Video / Audio Broadcast Client	QCAP_STOP_BROADCAST_CLIENT( pClient )
5	Delete Broadcast Client Object	QCAP_DESTROY_BROADCAST_CLIENT( pClient )

The Client functions provide viewing and control of live streams as a standard streaming client. For example, QCAP provides the function to support the **RTSP** client to allow you to receiving the live audio/video streams from the Internet, its role acts like a **RTSP** client such as **VLC Player** or **QuickTime Player**.

For **RTMP** protocol, QCAP supports **RTMP Web Portal**, instead of streaming to the client, it allows you to send audio/video streams directly to a remote **RTMP server**, such as Adobe® Flash Media Server. Then the **RTMP** clients can then connect to the public **RTMP** server and access those stream you prepared for them. QCAP APIs also can create a private **RTMP** server (with a **RTMP Web Portal** built-in), so that user can ONLY use a QCAP **RTMP** client to access it.

For the **TS over TCP/UDP** streaming is also well supported, with both Unicast / Multicast broadcasting technology. The user can either use QCAP **TS** client to playback the stream, or access the lives streams from a VLC Player.

### 11.1.4 Broadcasting Server Example

This section provides a Server streaming application sample as well as a Client receiving the sample to show the programming flow of QCAP API.

Example : A broadcasting server example

```
ULONG ServerType = SELECT_SERVER_TYPE;

if( ServerType == 1 ) //RTSP SERVER
{
    QCAP_CREATE_BROADCAST_RTSP_SERVER( 0, 1 , &pServer ); //1 sessions
}
else if( ServerType == 2 ) //HLS SERVER
{
    QCAP_CREATE_BROADCAST_HLS_SERVER( 0, 1,  &pServer, "C:/AppServer/www/", "hls/" ); //1 sessions
}
```

```

else if( ServerType == 3 ) //RTMP SERVER
{
    QCAP_CREATE_BROADCAST_RTMP_SERVER_EX( 0, 1, &pServer );    //1 sessions
}

else if( ServerType == 4 ) //RTMP Web Portal SERVER
{
    QCAP_CREATE_BROADCAST_RTMP_WEB_PORTAL_SERVER_EX( 0, "rtmp://xxx.xxx.xxx.xxx/live/session0.mpg", &pServer );
}
else if( ServerType == 5 ) //MMS Web Portal SERVER
{
    QCAP_CREATE_BROADCAST_MMS_WEB_PORTAL_SERVER( 0, "mms://xxx.xxx.xxx:yyy/MMS", &pServer );
}
else if( ServerType == 6 ) //TS over UDP SERVER
{
    QCAP_CREATE_BROADCAST_TS_OVER_UDP_SERVER( 0, "udp://xxx.xxx.xxx.xxx:yyy", &pServer );
}
else if( ServerType == 7 ) //TS over TCP SERVER
{
    QCAP_CREATE_BROADCAST_TS_OVER_TCP_SERVER( 0, "tcp://xxx.xxx.xxx.xxx:yyy", &pServer );
}
else
{
    //please select a correct server type;
    return;
}

//-----

QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer,
    0, /*CH01*/
    QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
    QCAP_ENCODER_FORMAT_H264,
    QCAP_COLORSPACE_TYEP_YUY2,
    480,
    270,
    30,
    QCAP_RECORD_MODE_CBR,
    8000,
    1000000,
    30,
    0,
    0,
    NULL,
    FALSE,
    FALSE,
    QCAP_BROADCAST_FLAG_NETWORK | QCAP_BROADCAST_FLAG_ENCODE );
QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer,
    1 /*CH02*/,
    QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,
    QCAP_ENCODER_FORMAT_H264,
    QCAP_COLORSPACE_TYEP_YUY2,
    480,
    270,
    30,

```

```

        QCAP_RECORD_MODE_CBR,
        8000,
        1000000,
        30,
        0,
        0,
        NULL,
        FALSE,
        FALSE,
        QCAP_BROADCAST_FLAG_NETWORK | QCAP_BROADCAST_FLAG_ENCODE );

QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY( pServer,
        0, /*CH01*/
        QCAP_ENCODER_TYPE_SOFTWARE,
        QCAP_ENCODER_FORMAT_AAC,
        2,
        16,
        48000 );

QCAP_START_BROADCAST_SERVER( pServer );

//Server is RUNNING!

//Keep push new audio/video buffer into the server for broadcasting while a server is running.

while ( server_runing )
{

    QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer,
        0 /*CH01*/,
        QCAP_COLORSPACE_TYEP_YUY2,
        1920,
        1080,
        pFrameBuffer,
        nFrameBufferLen );

    QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer,
        0 /*CH01*/,
        pFrameBuffer,
        nFrameBufferLen );

}

//Server will be stopped.

QCAP_STOP_BROADCAST_SERVER( pServer );

QCAP_DRSTROY_BROADCAST_SERVER( pServer );

```

## 11.1.5 Broadcasting Client Example

This section provides a Server streaming application sample as well as a Client receiving sample to show the programming flow of QCAP API.

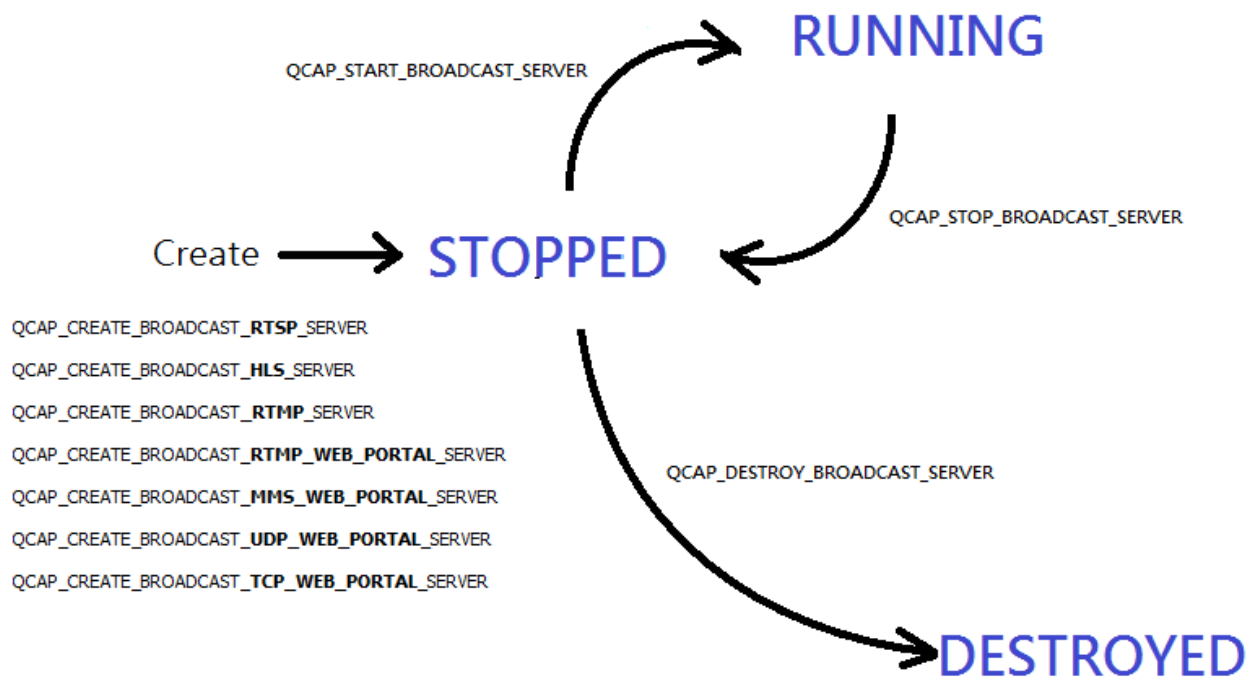
*Example : A broadcasting client example*

```
QCAP_CREATE_BROADCAST_CLIENT( 0,  
    "rtsp://root:1234@127.0.0.1:554/session0.mpg",  
    &pClient,  
    QCAP_DECODER_TYPE_SOFTWARE,  
    hWnd,  
    TRUE );  
  
QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK( pClient, on_process_video_stream, this );  
  
QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK( pClient, on_process_audio_stream, this );  
  
QCAP_START_BROADCAST_CLIENT( pClient );  
  
QCAP_STOP_BROADCAST_CLIENT( pClient );  
  
QCAP_DESTROY_BROADCAST_CLIENT( pClient );
```

## 11.2 Server Major Functions

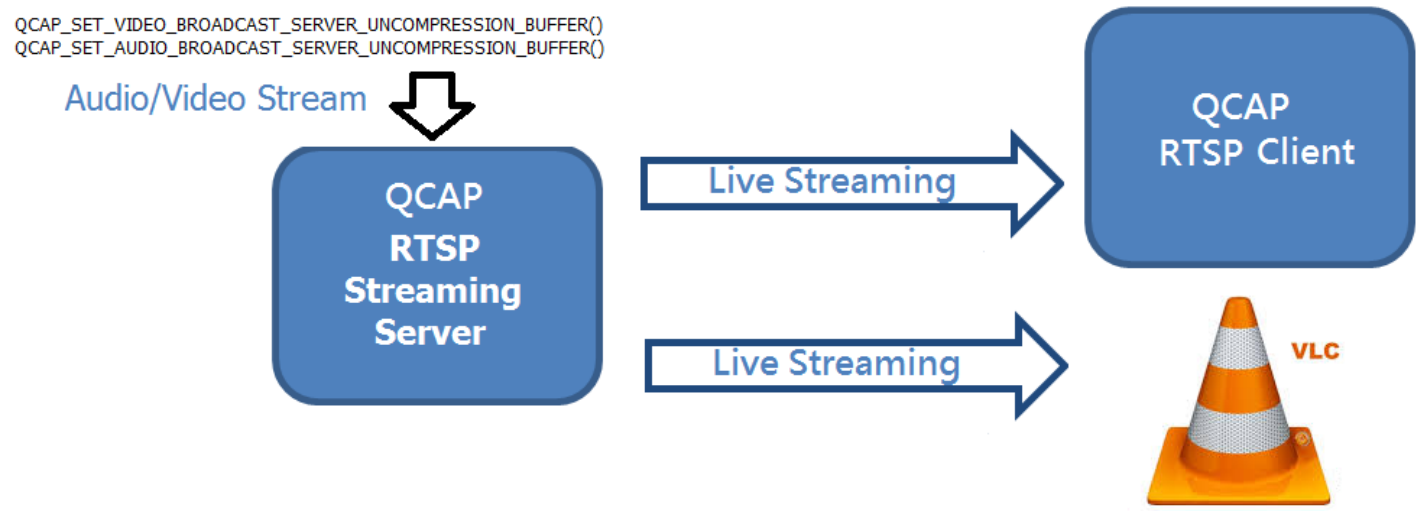
### Introduction

The is chapter provides functions to create a broadcasting server that support different streaming protocols. The user can depend on their applications to choice a technology by creating different server-client architecture. These functions made the server easy to establish and also provide client functions to connect to them.



# 11.2.1 QCAP\_CREATE\_BROADCAST\_RTSP\_SERVER

## Introduction



The Real Time Streaming Protocol (**RTSP**) is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. Most RTSP servers use the Real-time Transport Protocol (**RTP**) for media stream delivery. The default port number is 554.

This function can create a broadcast server object by using **RTSP** streaming protocol. The user can set the *iSvrNum* to select which server slot to create server objects. In each server, a slot could have the different port number.

The **Session** represents a pair of audio/video signal source of the capture device. The user can specify the total channel number in *nTotalSessions* to each server.

In server-side authentication, a log in account/password can set for the valid client application when they log in.

To establish a **RTSP** session where the video data (**RTP**) and commands (**RTSP**) are transported as HTTP traffic as known as "*HTTP Tunneling*", The *nNetworkPort\_RTSPOverHTTP* can set to enable it and allows the **RTSP** data to easily go through the firewalls,

Multi-casting ability helps you conserve bandwidth by reducing the number of live streams in use. It requires a specially configured network. The *bEnableMulticasting* can enable the Multi-casting function in server-side.

Like a VLC player, The **RTSP** client can use the URL, "*rtsp://account:password@ip:port/session.mpg*"# to access the server:

- account : the account name to log in the server
- password : the account password to log in the server
- ip: the IP address of server
- port : the of RTSP server port
- #: the index of session, start from 0

For example a URL: "*rtsp://user:1234@127.0.0.1:454/session0.mpg*"

## Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
ULONG	nTotalSessions	IN	Specify the total number of sessions, max is 8
PVOID *	ppServer	OUT	Handle of the broadcast server object
CHAR *	pszAccount	IN	<b>default NULL</b> Specify the server-side account information to log in
CHAR *	pszPassword	IN	

			<b>default NULL</b> Specify the server-side password information to log in
ULONG	nNetworkPort_RTSP	IN	<b>default 554</b> Specify the port number that the server listens on for <b>RTSP</b>
ULONG	nNetworkPort_RTSPOverHTTP	IN	<b>default 0</b> Specify the port number for server to listen on <b>RTSP</b> over <b>HTTP</b> requests Usually is 8080, Set 0 to turn off this function.
BOOL	bEnableMulticasting	IN	<b>default FALSE</b> Enable/Disable the multi-casting

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

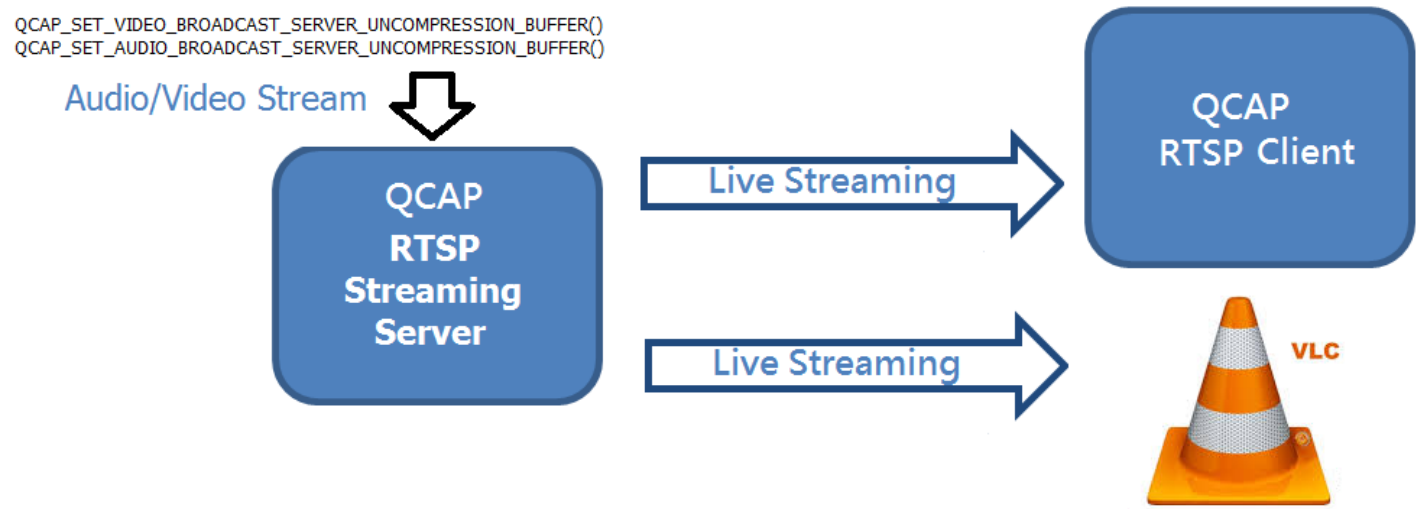
**Examples**

*Example : Create a **RTSP** Server with 4 sessions of audio/video input source*

```
QCAP_CREATE_BROADCAST_RTSP_SERVER( 0, 4,
                                     &pServer,
                                     "root",
                                     "1234",
                                     554,
                                     0,
                                     FALSE );
```

# 11.2.1 QCAP\_CREATE\_BROADCAST\_RTSP\_RAW\_UDP\_SERVER

## Introduction



This function can create a raw UDP broadcast server object by using **RTSP** streaming protocol.

## Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
ULONG	nTotalSessions	IN	Specify the total number of sessions, max is 8
PVOID *	ppServer	OUT	Handle of the broadcast server object
CHAR *	pszAccount	IN	<b>default NULL</b> Specify the server-side account information to log in
CHAR *	pszPassword	IN	<b>default NULL</b> Specify the server-side password information to log in
ULONG	nNetworkPort_RTSP	IN	<b>default 554</b> Specify the port number that the server listens on for <b>RTSP</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Create a **RTSP** UDP Server with 4 sessions of audio/video input source*

```
QCAP_CREATE_BROADCAST_RTSP_RAW_UDP_SERVER( 0, 4,  
                                             &pServer,  
                                             "root",  
                                             "1234",  
                                             554 );
```

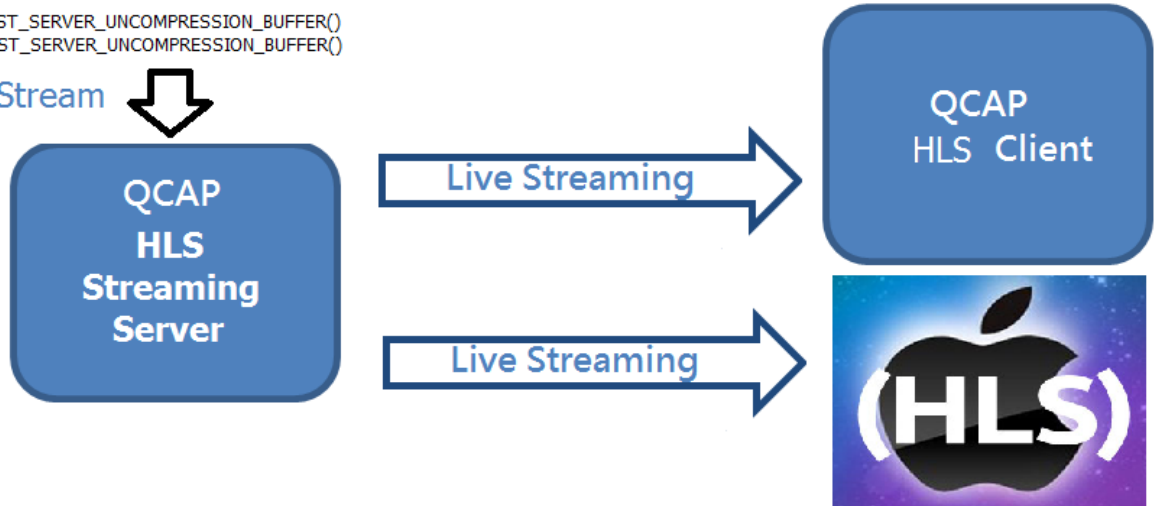


## 11.2.2 QCAP\_CREATE\_BROADCAST\_HLS\_SERVER

### Introduction

QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER()  
QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER()

Audio/Video Stream



**HTTP Live Streaming (HLS)** is an adaptive streaming communications protocol created by Apple® to communicate with iOS / Mac. The default port number is 80 for HTTP.

This function can create a broadcast server object by using **HLS** streaming protocol. The user can set the *iSvrNum* to select which server slot to create server objects. In each server, a slot could have the different port number.

The **Session** represents a pair of audio/video signal source of the capture device. The user can specify the total channel number in *nTotalSessions* to each server.

The encoded source stream will be divided into a series of small media files of equal duration by *nSegmentDuration*. These encoded streams place at HTTP web server root folder, along with a **M3U8** file that directs the player to each segment. The video files are made from a continuous stream which can be reconstructed seamlessly.

The **HLS** client can use the URL, "[#](http://xxx.xxx.xxx.xxx/hls/session.m3u8)" to access the server:

- xxx : the IP address of server
- has: the subfolder name can be changed by *SubFolderPath*
- #: the index of session, start from 0

For example a URL: <http://xxx.xxx.xxx.xxx/hls/session0.m3u8>



Due to **HLS** protocol needs accurate time-stamp for each frame, user must set *dSampleTime* parameter when calling:

- QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER(),
- QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER(),
- QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER(),
- QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER()

### Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
ULONG	nTotalSessions	IN	Specify the total number of sessions, max is 8
PVOID *	ppServer	OUT	Handle of the broadcast server object
CHAR *	pszWebServerRootFolderPath	IN	Specify the host <b>HTTP</b> web server's root folder path
CHAR *	pszSubFolderPath	IN	Specify the subfolder path to save <b>HLS</b> files.

ULONG	nSegmentDuration	IN	<b>default 1000</b> Set the segment duration in ms.
BOOL	bResumeSegmentNum	IN	<b>default FALSE</b> Set true to resume the segmentation number.
ULONG	nSegmentPlaylistCount	IN	<b>default 3</b> Set the HLS segmented streaming playlist count
CHAR *	pszWebServerIP	IN	<b>default NULL</b> Specify the web server IP address

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

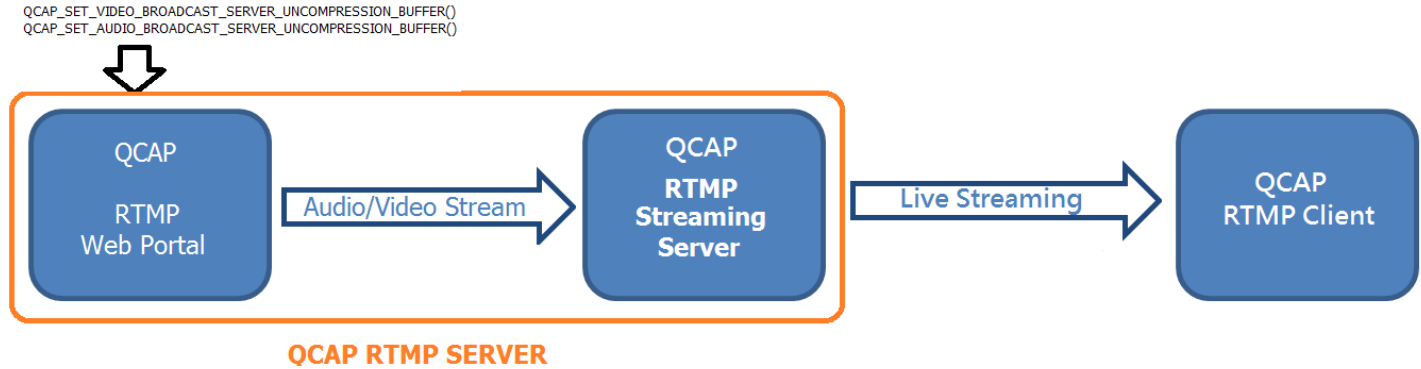
*Example : Create a **HLS** broadcasting server*

```
QCAP_CREATE_BROADCAST_HLS_SERVER( 0, 4,
                                   &pServer,
                                   "C:/AppServ/www/",
                                   "hls/",
                                   1000,
                                   FALSE, 3,
                                   NULL );
```

### 11.2.3 QCAP\_CREATE\_BROADCAST\_RTMP\_SERVER

### 11.2.4 QCAP\_CREATE\_BROADCAST\_RTMP\_SERVER\_EX

#### Introduction



Real Time Messaging Protocol (**RTMP**) Is a proprietary protocol developed by Adobe® Systems. That is primarily used a Flash Media Server to stream audio and video over the Internet to the Adobe® Flash Player client. The default connection port is 1935.

This function can create a broadcast server object by using **RTMP** streaming protocol. The user can set the *iSvrNum* to select which server slot to create server objects. In each server, a slot could have the different port number.

The **Session** represents a pair of audio/video signal source of the capture device. The user can specify the total channel number in *nTotalSessions* to each server. The audio encoding format has to be **QCAP\_ENCODER\_FORMAT\_AAC\_ADTS** in a **RTMP** server.

The *nNetworkPort\_RTMPOverHTTP* will enable "HTTP Tunneling" to allows the **RTMP** data to easily go through the firewalls,

In QCAP SDK internal design, these **RTMP** extended functions have the different implementation in library software.

- **QCAP\_CREATE\_BROADCAST\_RTMP\_SERVER()**
- **QCAP\_CREATE\_BROADCAST\_RTMP\_SERVER\_EX()**

The result for those 2 functions is the same. This function actually creates 2 objects:

1. One **RTMP** Web Portal object - To push the audio/video encoded stream to internal **RTMP** server
2. One **RTMP** Server object - The internal **RTMP** server for QCAP **RTMP** client to do content streaming.

Due to the protected streaming, the QCAP **RTMP** streaming server could only be connected by a **RTMP** client built with QCAP SDK. The client can uses the URL: "rtmp://account:password@ip:port/server\_name/session.mpg"# to access the server:

- account : the account name to log in the server
- password : the account password to log in the server
- port : the of **RTMP** server port
- server\_name: the custom server name *pszServerName*
- #: the index of session, start from 0

For example a URL: "rtmp://user:1234@127.0.0.1/LiveRTMPServer/session0.mpg"



Please refer to [QCAP SDK RTMP Server Installation Guide ENG](#).

## Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
ULONG	nTotalSessions	IN	Specify the total number of sessions, max is 8
PVOID *	ppServer	OUT	Handle of the broadcast server object
CHAR *	pszAccount	IN	<b>default NULL</b> Specify the account information for client to log in
CHAR *	pszPassword	IN	<b>default NULL</b> Specify the password for client to log in
ULONG	nNetworkPort_RTMP	IN	<b>default 1935</b> Specify the port number that the server listens on for <b>RTMP</b>
ULONG	nNetworkPort_RTMPOverHTTP	IN	<b>default 0</b> Specify the port number that the server listens on for <b>RTMP</b> over HTTP requests Usual is 8080. Set 0 to disable this function.
CHAR *	pszServerName	IN	<b>default NULL</b> Specify the server name, Set NULL to use the default name "flvplayback"
CHAR *	pszMediaFolderPath=NULL	IN	Specify the media folder path

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To create a broadcast **RTMP** server with custom **RTMP** Server Name*

```
//URL for Client to connect: "rtmp://root:1234@127.0.0.1/flvplayback/session0.mpg"
QCAP_CREATE_BROADCAST_RTMP_SERVER( 0, 1,
    &pServer,
    "root",
    "1234",
    1935,
    0,
    NULL );

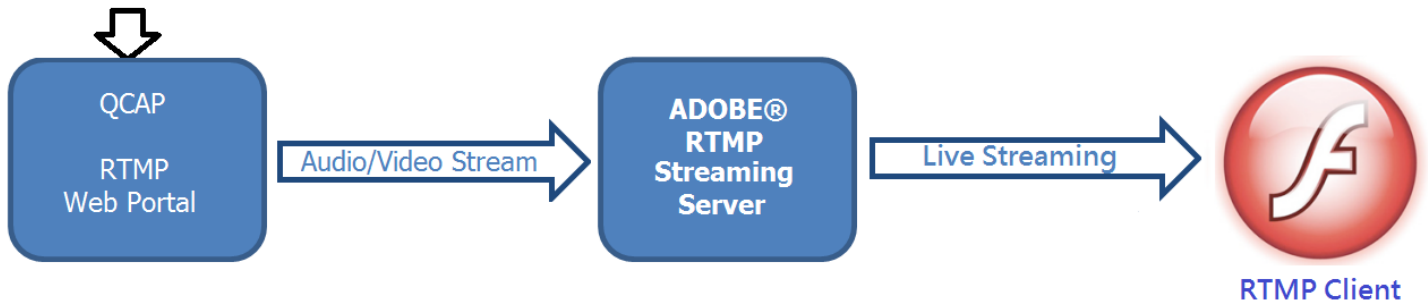
//URL for Client to connect: "rtmp://root:1234@127.0.0.1/LiveRTMPServer/session0.mpg"
QCAP_CREATE_BROADCAST_RTMP_SERVER_EX( 0, 1,
    &pServer,
    "root",
    "1234",
    1935,
    0,
    "LiveRTMPServer" );
```

## 11.2.5 QCAP\_CREATE\_BROADCAST\_RTMP\_WEB\_PORTAL\_SERVER

## 11.2.6 QCAP\_CREATE\_BROADCAST\_RTMP\_WEB\_PORTAL\_SERVER\_EX

### Introduction

QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER()  
QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER()



Real Time Messaging Protocol (**RTMP**) is a proprietary protocol developed by Adobe® Systems. This function can create a Web Portal server object by using **RTMP** streaming protocol. The user can set the *iSvrNum* to select which server slot to create server objects. In each server, a slot could have the different port number. The audio encoding format has to be **QCAP\_ENCODER\_FORMAT\_AAC\_ADTS** for a **RTMP** server.

The Web Portal server is *NOT* for clients to directly connected with, but for a user to push the audio/video encoded stream to a remote **RTMP** media server. The Web Portal object means to forward encoded media stream to a public Adobe® RTMP media server. Then any **RTMP** client like *Flash Player* could connect to the server to request the media contents.

In QCAP SDK internal design, these **RTMP** extended functions have the different implementation in library software.

- **QCAP\_CREATE\_BROADCAST\_RTMP\_WEB\_PORTAL\_SERVER()**
- **QCAP\_CREATE\_BROADCAST\_RTMP\_WEB\_PORTAL\_SERVER\_EX()**

The result for those 2 functions is the same.

The client uses the same URL in *pszURL* to access live streams from **RTMP** server, For example: *"rtmp://root:1234@10.10.0.80/live/session0.mpg"*.



Please refer to [QCAP SDK RTMP Web Portal with Adobe® FMS Security Account Installation Guide](#).

### Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
CHAR *	pszURL	IN	The URL string of the remote <b>RTMP</b> media server to forward content to e.g. <i>"rtmp://10.10.0.80/live/session0.mpg"</i>
PVOID *	ppServer	OUT	Handle of the broadcast server object
CHAR *	pszAccount	IN	<b>default NULL</b> Specify the account to log in remote server
CHAR *	pszPassword	IN	<b>default NULL</b> Specify the password to log in remote server

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

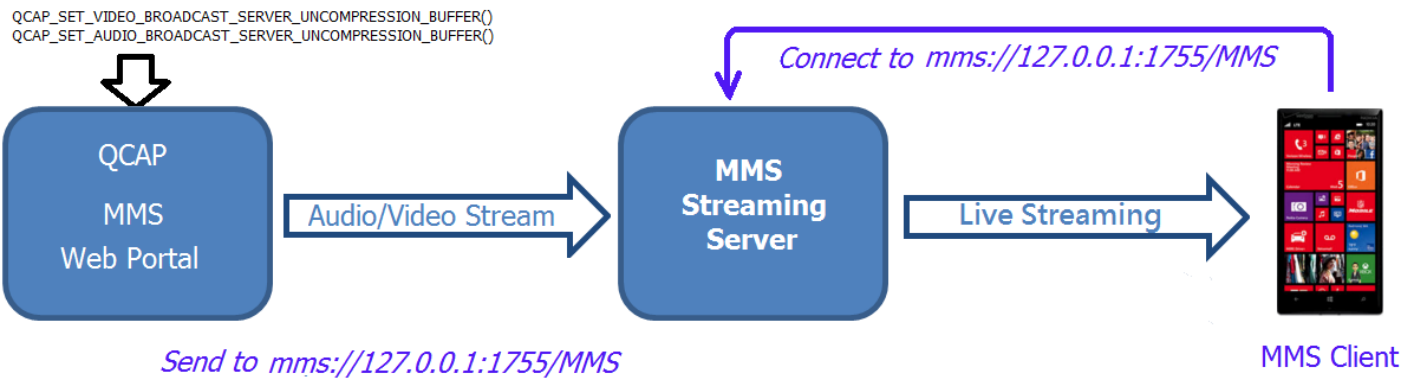
### Examples

*Example : Create a **RTSP** Web Portal Server to forward the encoded streams*

```
QCAP_CREATE_BROADCAST_RTMP_WEB_PORTAL_SERVER( 0,  
        "rtmp://10.10.0.80/live/session0.mpg",  
        &pServer,  
        "root",  
        "1234" );  
  
QCAP_CREATE_BROADCAST_RTMP_WEB_PORTAL_SERVER_EX( 0,  
        "rtmp://10.10.0.80/live/session0.mpg",  
        &pServer,  
        "root",  
        "1234" );
```

# 11.2.7 QCAP\_CREATE\_BROADCAST\_MMS\_WEB\_PORTAL\_SERVER

## Introduction



Microsoft® Media Server (**MMS**), a Microsoft® proprietary network streaming protocol, serves to transfer unicast data in Windows® Media Services (previously called *NetShow Services*). The **MMS** default port number is 1755 (**TCP/UDP**).

This function can create a Web Portal server object by using **MMS** streaming protocol. The user can set the *iSvrNum* to select which server slot to create server objects. In each server, a slot could have a different port number.

The Web Portal server is *NOT* for clients to directly connected with, but for a user to push the audio/video encoded stream to a remote **MMS** media server. The Web Portal means to forward encoded media stream to a public Microsoft® Media Server. Then any **MMS** clients could connect to the server to request the media contents.

The client uses the same URL in *pszURL* to access live streams from **RTMP** server, For example: *"mms://root:1234@127.0.0.1:1755/MMS"*

## Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
CHAR *	pszURL	IN	The URL string of the remote <b>RTMP</b> media server to forward content to e.g. <i>"mms://127.0.0.1:1755/MMS"</i>
PVOID *	ppServer	OUT	Handle of the broadcast server object
CHAR *	pszAccount	IN	<b>default NULL</b> Specify the server-side account information to log in
CHAR *	pszPassword	IN	<b>default NULL</b> Specify the server-side password information to log in

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Create a **MMS** broadcasting server*

```
QCAP_CREATE_BROADCAST_MMS_WEB_PORTAL_SERVER( 1,
    "mms://127.0.0.1:1755/MMS",
    &pServer,
    "root",
    "1234" );
```

## 11.2.7 QCAP\_CREATE\_BROADCAST\_WEBRTC\_SERVER

### Introduction

This function can create a Web RTC server object by using WebRTC streaming.

WebRTC (Web Real-Time Communication) is an API definition drafted by the World Wide Web Consortium (W3C) that supports browser-to-browser applications for voice calling, video chat, and P2P file sharing without the need of either internal or external plugins.

### Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
CHAR *	pszLoginIP	IN	Specify the Login IP address
CHAR *	pszLoginPort	IN	Specify the Login port number
ULONG	nLoginID	IN	Specify the Login account

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Create a WebRTC broadcasting server*

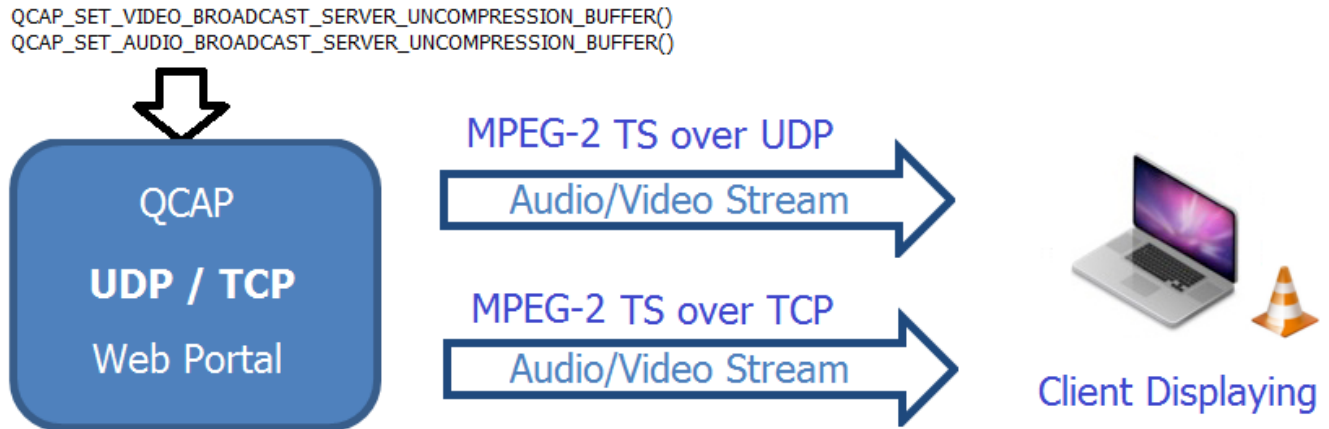
```
QCAP_CREATE_BROADCAST_WEBRTC_SERVER( 1,  
                                     "127.0.0.1",  
                                     1234,  
                                     "user" );
```



## 11.2.8 QCAP\_CREATE\_BROADCAST\_TS\_OVER\_UDP\_SERVER

## 11.2.9 QCAP\_CREATE\_BROADCAST\_TS\_OVER\_TCP\_SERVER

### Introduction



This function can create a server object by using **TS\_over\_UDP** streaming technology for sending **MPEG2-TS** encoded stream.

The server is for **MPEG2-TS** clients to directly connected with, it transfers **MPEG2-TS** encoded media stream to clients that requests the media contents.

The user can set the *iSvrNum* to select which server slot to create server objects. In each server, a slot could have the different port number. The internal session corresponds to an pair of audio/video signal source of the capture device.

The user can specify the custom Program Management PID, Video PID, Audio PID to identify the program currently in broadcasting. The *nTransferbit rate* parameter can control the maximum bandwidth used, set 0 to auto control bit rate by QCAP.

The TCP/UDP client can use the URL, "ppp://xxx.xxx.xxx.xxx:yyy" to access the server:

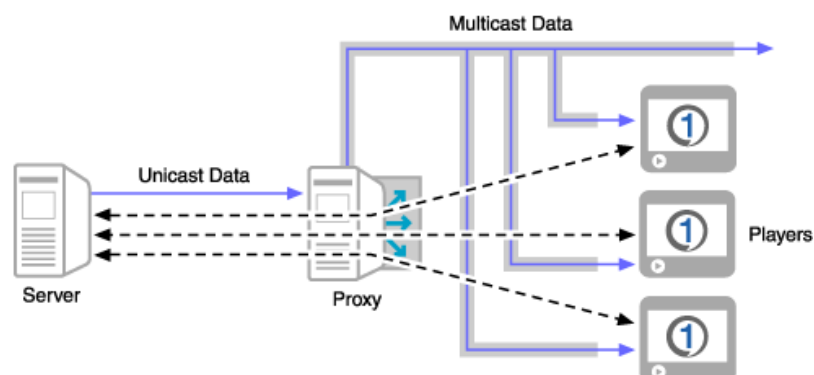
ppp : choice UDP or TCP protocol

xxx : the IP address of server

yyy: the port number TCP/UDP server

The client uses the same URL in *pszURL* to access live streams from a **MPEG2-TS** server. If your network infrastructure supports multi-casting, you can different URL to select between them. For example, a VLC player could connect stream to:

- Unicast: "udp://10.10.80.XX:888"
- Multicast: "udp://10.0.0.1:888"



Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
CHAR *	pszURL	IN	The URL string of the remote <b>MPEG-TS</b> media server to forward content to The IP address can select Unicast / Multicast transfer protocol. e.g.: * <b>TCP</b> Unicast: <u>"tcp://10.10.80.XX:888"</u> * <b>TCP</b> Multicast: <u>"tcp://234.0.0.1:888"</u> * <b>UDP</b> Unicast: <u>"udp://10.10.80.XX:888"</u> * <b>UDP</b> Multicast: <u>"udp://234.0.0.1:888"</u>
PVOID *	ppServer	OUT	Handle of the broadcast server object
ULONG	nServiceID	IN	<b>default 1</b> Specify the custom Service ID
CHAR *	pszServiceName	IN	<b>default NULL</b> Specify the custom stream name information
CHAR *	pszServiceProviderName	IN	<b>default NULL</b> Specify the custom Publisher name information to log in
ULONG	nTransferBitRate	IN	<b>default 0 = FREE</b> Specify the MUX of max transfer bit rate, 0 for auto set
ULONG	nPMT_PID	IN	<b>default 4096</b> Specify the custom Program Management PID
ULONG	nVideo_PID	IN	<b>default 256</b> Specify the custom Video PID
ULONG	nAudio_PID	IN	<b>default 257</b> Specify the custom Audio PID
ULONG	nVideo_CodecID	IN	<b>default 0 = AUTO</b> Specify the Video codec ID
ULONG	nAudio_CodecID	IN	<b>default 0 = AUTO</b> Specify the Audio codec ID
CHAR *	pszNetworkAdapterIP	IN	The IP address for the client network adapter to do transmission <b>Only in QCAP_CREATE_BROADCAST_TS_OVER_UDP_SERVER()</b>

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

Example : Create a **TS\_over\_UDP / TS\_over\_UDP** broadcasting server

```
QCAP_CREATE_BROADCAST_TS_OVER_UDP_SERVER( 0,
    "udp://10.10.80.XX:888",
    &pServer,
    1,
    "UDP LIVE SHOW",
    "FOX TV",
    0,
    4096,
    256,
    257 );

QCAP_CREATE_BROADCAST_TS_OVER_TCP_SERVER( 1,
    "tcp://10.10.80.XX:888",
    &pServer,
    1,
    "TCP LIVE SHOW",
    "FOX TV",
    0,
    4096,
    256,
    257 );
```

# 11.2.10 QCAP\_START\_BROADCAST\_SERVER

## Introduction

The user can use this function to start a broadcast server, and change its status to running.



The user must set `QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY()` and `QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY()` before calls to this function.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object

## Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

## Examples

*Example : To start a broadcasting server for clients to request streaming*

```
QCAP_START_BROADCAST_SERVER( pServer );
```

# 11.2.11 QCAP\_STOP\_BROADCAST\_SERVER

## Introduction

The user can use this function to stop a broadcast server and stop the live content streaming.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object

## Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

## Examples

*Example : To stop a broadcasting server from live streaming*

```
QCAP_STOP_BROADCAST_SERVER( pServer );
```

# 11.2.12 QCAP\_DESTROY\_BROADCAST\_SERVER

## Introduction

This function can destroy the broadcast server object and release its system resource.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To destroy a broadcast object and free resources*

```
QCAP_DESTROY_BROADCAST_SERVER( pServer );
```

# 11.2.13 QCAP\_GET\_BROADCAST\_SERVER\_STATUS

## Introduction

The user can use this function to monitor the broadcast server's resources. For example, it can help to see if a server slot index is available to start a new broadcast server.

## Parameters

type	parameter	I/O	descriptions
UINT	iSvrNum	IN	Specify the index of broadcast servers to create object, range from 0-63
BOOL	pIsValid	OUT	Pointer to the availability of this server slot

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : See if server slot 0 is in-used or not.*

```
QCAP_GET_BROADCAST_SERVER_STATUS( 0, &pIsValid );
```

# 11.3 Server Property Functions

## Introduction

This chapter provides functions to access broadcasting encoder properties and are for **Software Encoder Only**. Before you want to broadcast those live streams to the client-side, you may want to fine-tune properties/setting is server-side so that the broadcasting servers will output the effect you want. When a user pushes uncompressed buffers to the broadcasting engine, those functions can adjust the detailed parameters of encoder behavior.

To access the audio/video broadcasting server properties functions:

mode	functions
For <b>Software Encoder Only</b>	
Set broadcasting properties	QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY()
	QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()
	QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY()
	QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY_EX()
Get broadcasting properties	QCAP_SET_VIDEO_BROADCAST_SERVER_DYNAMIC_PROPERTY_EX()
	QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY()
	QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()
	QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY()
	QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY_EX()
	QCAP_GET_VIDEO_BROADCAST_SERVER_DYNAMIC_PROPERTY_EX()



For hardware encoder property please refer to **QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY()**

## 11.3.1 QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_PROPERTY

## 11.3.2 QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_PROPERTY\_EX

### Introduction

The user can use this function to set video properties for the i-th video stream session in a broadcasting server.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_VIDEO\_SHARE\_RECORD\_PROPERTY\_EX()**.



In broadcasting server, this function is For **Software encoder** only!  
For hardware encoder property please refer to **QCAP\_SET\_VIDEO\_HARDWARE\_ENCODER\_PROPERTY()**

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE QCAP_ENCODER_TYPE_HARDWARE QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK QCAP_ENCODER_TYPE_AMD_STREAM QCAP_ENCODER_TYPE_NVIDIA_CUDA QCAP_ENCODER_TYPE_NVIDIA_NVENC <b>Note: For hard ware encoder property please refer to</b> <b>QCAP_SET_VIDEO_HARDWARE_ENCODER_PROPERTY()</b>
ULONG	nEncoderFormat	IN	Specify video encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D

			QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the width
ULONG	nHeight	IN	Specify the height
double	dFrameRate	IN	Specify the frame rate
ULONG	nRecordProfile	IN	<b>default QCAP_RECORD_PROFILE_BASELINE</b> Specify recording profile: QCAP_RECORD_PROFILE_BASELINE QCAP_RECORD_PROFILE_MAIN QCAP_RECORD_PROFILE_HIGH QCAP_RECORD_PROFILE_CONSTRAINED_BASELINE QCAP_RECORD_PROFILE_CONSTRAINED_HIGH <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nRecordLevel	IN	<b>default 41</b> Specify recording levels: QCAP_RECORD_LEVEL_1 QCAP_RECORD_LEVEL_1B QCAP_RECORD_LEVEL_11 QCAP_RECORD_LEVEL_12 QCAP_RECORD_LEVEL_13 QCAP_RECORD_LEVEL_2 QCAP_RECORD_LEVEL_21 QCAP_RECORD_LEVEL_22 QCAP_RECORD_LEVEL_3 QCAP_RECORD_LEVEL_31 QCAP_RECORD_LEVEL_32 QCAP_RECORD_LEVEL_4 QCAP_RECORD_LEVEL_41 QCAP_RECORD_LEVEL_42 QCAP_RECORD_LEVEL_50 QCAP_RECORD_LEVEL_51 QCAP_RECORD_LEVEL_52 QCAP_RECORD_LEVEL_60 QCAP_RECORD_LEVEL_61 QCAP_RECORD_LEVEL_62 <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nRecordEntropy	IN	<b>default QCAP_RECORD_ENTROPY_CAVLC</b> Specify recording entropy: QCAP_RECORD_ENTROPY_CAVLC QCAP_RECORD_ENTROPY_CABAC <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nRecordComplexity	IN	<b>default 0</b> Specify recording complexity: QCAP_RECORD_COMPLEXITY_0 (Best Speed) QCAP_RECORD_COMPLEXITY_1 QCAP_RECORD_COMPLEXITY_2 QCAP_RECORD_COMPLEXITY_3

			QCAP_RECORD_COMPLEXITY_4 QCAP_RECORD_COMPLEXITY_5 QCAP_RECORD_COMPLEXITY_6 (Best Quality) <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify video quality. It is used by <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify video bit rate. It is used by <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording GOP size, from 0-255
ULONG	nBFrames	IN	<b>default 0</b> Specify recordingBFrames <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL	bIsInterleaved	IN	<b>default FALSE</b> Specify broadcast server Interleaved <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nSlices	IN	<b>default 0</b> Specify broadcast server slices <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nLayers	IN	<b>default 0</b> Specify broadcast server layers <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nSceneCut	IN	<b>default 0</b> Specify recording Scene Cut, recommended value is 40, 0 is function turned off <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL	bMultiThread	IN	<b>default TRUE</b> Enable/Disable the multi-threaded CPU loading balance support <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL	bMBBRC	IN	<b>default FALSE</b> Enable/Disable the mbbrc <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL	bExtBRC	IN	<b>default FALSE</b> Enable/Disable the extbrc <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nMinQP	IN	<b>default 0</b> Specify the value of x264 Minimum quantizer settings <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nMaxQP	IN	<b>default 0</b> Specify the value of x264 Maximum quantizer settings <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nVBVMaxRate	IN	<b>default 0</b> Specify the value that x264 fills the buffer at (up to) the max rate <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nVBVBufSize	IN	<b>default 0</b> Specify the size that x264 fills the buffer <b>Only in QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG	nAspectRatioX	IN	Specify the aspect ratio X axis, 0 to turned off
ULONG	nAspectRatioY	IN	Specify the aspect ratio Y axis, 0 to turned off
HWND	hAttachedWindow	IN	<b>default NULL</b> Specify the handle of the window to show the i-th video stream session
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio
DWORD	dwFlags	IN	<b>default QCAP_BROADCAST_FLAG_FULL</b> Specify broadcasting mode: QCAP_BROADCAST_FLAG_FULL



			QCAP_BROADCAST_FLAG_NETWORK
			QCAP_BROADCAST_FLAG_ENCODE
			QCAP_BROADCAST_FLAG_DISPLAY
			QCAP_BROADCAST_FLAG_DECODE
			QCAP_BROADCAST_FLAG_VIDEO_ONLY
			QCAP_BROADCAST_FLAG_AUDIO_ONLY
			QCAP_BROADCAST_FLAG_VIDEO_USE_IDEAL_TIMESTAMP
			QCAP_BROADCAST_FLAG_AUDIO_USE_IDEAL_TIMESTAMP
			QCAP_BROADCAST_FLAG_VIDEO_USE_MEDIA_TIMER.
			QCAP_BROADCAST_FLAG_AUDIO_USE_MEDIA_TIMER.

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : Set the video broadcasting properties to the server*

```
QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer, 0,
    QCAP_ENCODER_TYPE_SOFTWARE,
    QCAP_ENCODER_FORMAT_H264,
    QCAP_COLORSPACE_TYEP_YUY2,
    320, 240,
    60,
    QCAP_RECORD_MODE_CBR,
    8000, 12 * 1024 * 1024,
    30,
    4, 3,
    hAttachedWindow,
    TRUE,
    FALSE,
    QCAP_BROADCAST_FLAG_FULL );

QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX( pServer, 0,
    QCAP_ENCODER_TYPE_SOFTWARE,
    QCAP_ENCODER_FORMAT_H264,
    QCAP_COLORSPACE_TYEP_YUY2,
    320, 240,
    60,
    QCAP_RECORD_PROFILE_BASELINE,
    41,
    QCAP_RECORD_ENTROPY_CAVLC,
    0,
    QCAP_RECORD_MODE_CBR,
    8000, 12 * 1024 * 1024,
    30,
    0, FALSE, 0, 0, 0,
    TRUE, FALSE, FALSE,
    0, 0, 0, 0,
    4, 3,
    hAttachedWindow,
    TRUE,
    FALSE,
    QCAP_BROADCAST_FLAG_FULL );
```

## 11.3.3 QCAP\_GET\_VIDEO\_BROADCAST\_SERVER\_PROPERTY

## 11.3.4 QCAP\_GET\_VIDEO\_BROADCAST\_SERVER\_PROPERTY\_EX

### Introduction

The user can use this function to get video properties for the i-th video stream session in a broadcasting server.

For more detailed parameters descriptions, please refer to `QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()`.



In broadcasting server, this function is For **Software encoder** only!

For hardware encoder property please refer to `QCAP_GET_VIDEO_HARDWARE_ENCODER_PROPERTY()`

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG *	pEncoderType	OUT	Pointer to the encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the encoder format
ULONG *	pColorSpaceType	OUT	Pointer to the color space type
ULONG *	pWidth	OUT	Pointer to the horizontal width
ULONG *	pHeight	OUT	Pointer to the vertical height
double *	pFrameRate	OUT	Pointer to the frame rate
ULONG *	pRecordProfile	OUT	Pointer to the record profile <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pRecordLevel	OUT	Pointer to the record level <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pRecordEntropy	OUT	Pointer to the record entropy <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pRecordComplexity	OUT	Pointer to the record complexity <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pRecordMode	OUT	Pointer to the record mode
ULONG *	pQuality	OUT	Pointer to the quality
ULONG *	pBitRate	OUT	Pointer to the bit rate
ULONG *	pGOP	OUT	Pointer to the GOP size
ULONG *	pBFrames	OUT	Pointer to the B-Frames <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL *	pIsInterleaved	OUT	Pointer to the current interleaved <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pSlices	OUT	Pointer to the current slices <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pLayers	OUT	Pointer to the current layers <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pSceneCut	OUT	Pointer to the screen cut <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL *	pMultiThread	OUT	Pointer to the current multi-threaded CPU loading balance status <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL *	pMBBRC	OUT	Pointer to the current mbbrc status <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
BOOL *	pExtBRC	OUT	Pointer to the current extbrc status <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pMinQP	OUT	Pointer to the value of x264 Minimum quantizer settings <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pMaxQP	OUT	Pointer to the value of x264 Maximum quantizer settings <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pVBVMaxRate	OUT	Pointer to the value that x264 fills the buffer at (up to) the max rate <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pVBVBufSize	OUT	Pointer to the size that x264 fills the buffer <b>Only in QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX()</b>
ULONG *	pAspectRatioX	OUT	Pointer to the aspect ratio X axis
ULONG *	pAspectRatioY	OUT	Pointer to the aspect ratio Y axis
HWND *	pAttachedWindow	OUT	Pointer to the window to show the i-th video stream session





### 11.3.7 QCAP\_GET\_AUDIO\_BROADCAST\_SERVER\_PROPERTY

### 11.3.8 QCAP\_GET\_AUDIO\_BROADCAST\_SERVER\_PROPERTY\_EX

## Introduction

The user can use this function to get audio properties for the i-th audio stream session in broadcasting server.

For more detailed parameters descriptions, please refer to **QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_PROPERTY\_EX()**.



All audio data use **software encoder** in broadcasting.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG *	pEncoderType	OUT	Pointer to the encoder type for the i-th audio stream session
ULONG *	pEncoderFormat	OUT	Pointer to the encoder format
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency
ULONG *	pBitRate	OUT	Pointer to the audio renderer's bit rate <b>Only in QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY_EX()</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the audio broadcasting properties from server*

[illegible]

( )

## 11.3.9 QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_DYNAMIC\_PROPERTY\_EX

### Introduction

This function is only for software encoders to set current share recording properties *dynamically*, such as *bit rate* & *GOP reconfiguration*... etc.



In broadcasting server, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nRecordMode	IN	Specify recording mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify video quality. It is used by <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify video bit rate. It is used by <b>CBR</b> and <b>ABR</b> . e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify recording GOP size, from 0-255

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## 11.3.10 QCAP\_GET\_VIDEO\_BROADCAST\_SERVER\_DYNAMIC\_PROPERTY\_EX

### Introduction

This function is only for software encoders to retrieve current share recording properties *dynamically*, such as *bit rate* & *GOP reconfiguration*... etc.



In broadcasting server, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG *	pRecordMode	OUT	Pointer to the record mode
ULONG *	pQuality	OUT	Pointer to the quality
ULONG *	pBitRate	OUT	Pointer to the bit rate
ULONG *	pGOP	OUT	Pointer to the GOP size

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Change and retrieve current bit rate setting of broadcasting server in run-time*

```
QCAP_SET_VIDEO_BROADCAST_SERVER_DYNAMIC_PROPERTY_EX( pServer, 0,  
                                                     QCAP_RECORD_MODE_CBR, 8000, 12 * 1024 * 1024, 30 );  
  
QCAP_GET_VIDEO_BROADCAST_SERVER_DYNAMIC_PROPERTY_EX( pServer, 0,  
                                                     &nRecordMode, &nQuality, &nBitRate, &nGOP );
```

## 11.3.11 QCAP\_SET\_AUDIO\_MX\_BROADCAST\_SERVER\_PROPERTY\_EX

### Introduction

This function is only for software encoders to set current audio mixer properties , such as *bit rate & sample frequency... etc.* in server side.



In broadcasting server, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nTracks	IN	Specify the audio track number, range 0-3
ULONG	nEncoderType	IN	Specify the encoder type: QCAP_ENCODER_TYPE_SOFTWARE
ULONG	nEncoderFormat	IN	Specify audio encoder format: QCAP_ENCODER_FORMAT_PCM QCAP_ENCODER_FORMAT_AAC QCAP_ENCODER_FORMAT_AAC_ADTS
ULONG	nChannels	IN	Specify the total audio channels
ULONG	nBitsPerSample	IN	Specify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency
ULONG	nBitRate	IN	Specify audio bit rate. default 128k and max is 496k

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Change current audio mixer setting of broadcasting server in run-time*

```
QCAP_SET_AUDIO_MX_BROADCAST_SERVER_PROPERTY_EX( pServer, 0, 0,  
                                                QCAP_ENCODER_TYPE_SOFTWARE,  
                                                QCAP_ENCODER_FORMAT_AAC,  
                                                2,  
                                                16,  
                                                48000,  
                                                128*512 );
```

## 11.3.12 QCAP\_GET\_AUDIO\_MX\_BROADCAST\_SERVER\_PROPERTY\_EX

### Introduction

This function is only for software encoders to retrieve current audio mixer properties in server side.



In broadcasting server, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG *	pTracks	OUT	Indicate the audio track number, range 0-3
ULONG *	pEncoderType	OUT	Pointer to the encoder type for the i-th audio stream session
ULONG *	pEncoderFormat	OUT	Pointer to the encoder format
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency
ULONG *	pBitRate	OUT	Pointer to the audio renderer's bit rate

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : Retrieve current audio mixer setting of broadcasting server in run-time*

```
QCAP_GET_AUDIO_MX_BROADCAST_SERVER_PROPERTY_EX( pServer, 0, 0,  
                                                  &nEncoderType,  
                                                  &nEncoderFormat,  
                                                  &nChannels,  
                                                  &nBitsPerSample,  
                                                  &nSampleFrequency,  
                                                  &nBitRate );
```

C



# 11.4 Server Audio Functions

## Introduction

This is section contains the audio property to set in the device, such as sound renderer can decide which output device to play the sound, and it's volume setting in server broadcasting.

## 11.4.1 QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_SOUND\_RENDERER

### Introduction

This function can select current sound renderer from available sound output device for the i-th session in broadcasting.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_AUDIO\_SOUND\_RENDERER()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iSoundNum	IN	Sound Renderer, default Renderer is 0

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set the current audio recording input device to Sound Device #1 in broadcasting*

```
UINT nSounder = 1;

QCAP_SET_AUDIO_BROADCAST_SERVER_SOUND_RENDERER( pServer, 0, nSounder );
```

## 11.4.2 QCAP\_GET\_AUDIO\_BROADCAST\_SERVER\_SOUND\_RENDERER

### Introduction

This function can get current sound renderer of sound output device used for the i-th session in broadcasting.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT *	pSoundNum	OUT	Pointer to the Sound number

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get the current audio recording input device in broadcasting*

```
UINT nSounder = 0;

QCAP_GET_AUDIO_BROADCAST_SERVER_SOUND_RENDERER( pServer, 0, &nSounder );
```

## 11.4.3 QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_VOLUME

### Introduction

This function can set current audio volume value for the i-th session in broadcasting, range from 0 to 100.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nVolume	IN	Specify the audio volume, from 0-100, 50 is original volume

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set current audio volume output to 50% in broadcasting*

```
QCAP_SET_AUDIO_BROADCAST_SERVER_VOLUME( pServer, 0, 50 );
```

## 11.4.4 QCAP\_GET\_AUDIO\_BROADCAST\_SERVER\_VOLUME

### Introduction

This function can get current audio volume value in broadcasting, range from 0 to 100.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG *	pVolume	OUT	Pointer to the audio volume

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get current audio volume output in broadcasting*

```
ULONG nVolume = 0;

QCAP_GET_AUDIO_BROADCAST_SERVER_VOLUME( pServer, 0, &nVolume );
```

## 11.5 Server Data Functions

### Introduction

When a broadcasting video setup to start working, the next step is to push the real video data into the broadcasting engine.

For a user who just wants a simple setting of a broadcasting server, without the need to understand how the buffer pushes to the broadcasting. QCAP also provide a **QCAP\_SET\_SESSION\_BROADCAST\_SERVER\_SOURCE()** function, with simple attached a capture object and let the broadcasting engine done the rest for you!

On the other hand to push the stream buffer manually, a user can either push the uncompressed data from the capture device or push the compressed data comes from hardware encoder directly. For every data frames pushed into the broadcasting engine, will be sent to the networking for streaming output.



The performance of **RGB** type is faster than **BGR** in our alpha blending algorithm.

### 11.5.1 QCAP\_SET\_SESSION\_BROADCAST\_SERVER\_SOURCE

### 11.5.2 QCAP\_SET\_SESSION\_BROADCAST\_SERVER\_SOURCE\_EX

#### Introduction

In broadcasting, a user needs to push channel device uncompressed video buffers manually into the broadcasting engine. This function will make the push buffer job done automatically. That means user just provide an capture card object as audio/video input source. By calling this function to attach the device, the rest of audio/video buffer pushing processes will be done automatically by broadcasting engine.

So if a user just wants a simple server setting, don't want to worry about what buffer need to be pushed, just given a capture device object, and then broadcasting audio/video will be working instantly.

This function for a user who want a basic setting of a broadcasting server.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SET_SESSION_BROADCAST_SERVER_SOURCE_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SET_SESSION_BROADCAST_SERVER_SOURCE_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SET_SESSION_BROADCAST_SERVER_SOURCE_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SET_SESSION_BROADCAST_SERVER_SOURCE_EX()</b>
ULONG	nScaleStyle	IN	default <b>QCAP_SCALE_STYLE_STRETCH</b> specify the scale styles: QCAP_SCALE_STYLE_STRETCH QCAP_SCALE_STYLE_FIT QCAP_SCALE_STYLE_FILL <b>Only in QCAP_SET_SESSION_BROADCAST_SERVER_SOURCE_EX()</b>
ULONG	nSequenceStyle	IN	default <b>QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Given a capture object and the rest of buffer jobs will be done automatically*

[illegible]

## 11.5.3 QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER

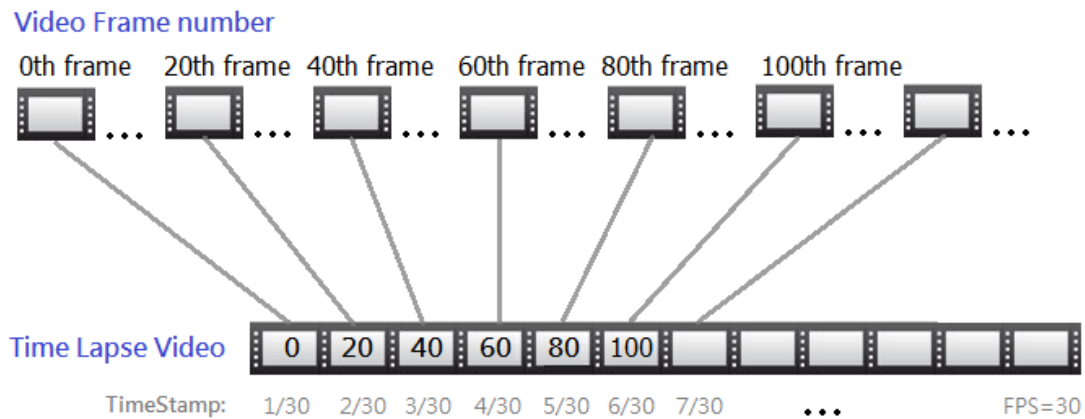
## 11.5.4 QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER\_EX

### Introduction

The user can use this function to push uncompressed video buffer into broadcasting. The server will send the buffer into the encoder and generate its bitstream, the encoded bitstream can be sent out to The internet.

If the crop region width and height of source frame are different from the target frame size, it will be scaled to target display size before encoding.

The users can use *SampleTime* to control time-stamp of video, and will result a Time-Lapse video, as shown below:



In broadcasting, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nColorSpaceType	IN	Specify the input source buffer's color space type, : QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the input source buffer's width
ULONG	nHeight	IN	Specify the input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
ULONG	nCropX	IN	Specify the input source buffer's width <b>Only in</b> <b>QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropY	IN	Specify the input source buffer's height <b>Only in</b> <b>QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropW	IN	

			Specify the width of crop <b>Only in</b> <b>QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in</b> <b>QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nScaleStyle	IN	<b>default QCAP_SCALE_STYLE_STRETCH</b> specify the scale styles: QCAP_SCALE_STYLE_STRETCH QCAP_SCALE_STYLE_FIT QCAP_SCALE_STYLE_FILL <b>Only in</b> <b>QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
BOOL	bForceKeyFrame	IN	<b>default FALSE</b> Enforce the input source's frame is keyframe, the default FALSE <b>Only in</b> <b>QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
double	dSampleTime	IN	<b>default 0.0+</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock. <b>Note:</b> <b>HLS</b> server need to set accuracy time-stamps

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

Example : Push the an uncompressed video buffer to broadcasting engine

```
QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0,
    QCAP_COLORSPACE_TYEP_YUY2,
    1280, 720,
    pFrameBuffer,
    1280 * 720 * 2,
    0 );

QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX( pServer, 0,
    QCAP_COLORSPACE_TYEP_YUY2,
    1280, 720,
    pFrameBuffer,
    1280 * 720 * 2,
    0, 0, 720, 480,
    QCAP_SCALE_STYLE_STRETCH,
    FALSE,
    0 );
```

## 11.5.5 QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER

## 11.5.6 QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER\_EX

### Introduction

The user can call this function to push an uncompressed audio buffer into broadcasting engine. The extended version can also help to re-sample audio format. For example, audio format from (2ch 16Bit 48KHz) transform to (1ch 8Bit 48KHz).



In broadcasting, this function is For **Software encoder** only!

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nChannels	IN	Specify the total audio channels <b>Only in</b> <b>QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nBitsPerSample	IN	Specify the audio bits per sample <b>Only in</b> <b>QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nSampleFrequency	IN	Specify the audio sample frequency <b>Only in</b> <b>QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX()</b>
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock. <b>Note: HLS server need to set accuracy time-stamps</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Push an uncompressed audio buffer to broadcasting engine*

```
QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer,
    0,
    pFrameBuffer,
    4096,
    0 );

QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER_EX( pServer,
    0,
    2, 16, 48000,
    pFrameBuffer,
    nFrameBufferLen );
```

# 11.5.7 QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER

## Introduction

The user can use this function to push compression video stream buffer directly into broadcasting. Due to the stream buffer is already encoded, this function will not enable a software encoder.



In broadcasting, this function is For **Hardware encoder** only!

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
BYTE *	pStreamBuffer	IN	Specify the input source buffer
ULONG	nStreamBufferLen	IN	Specify the input source buffer's size
ULONG	blsKeyFrame	IN	Specify the input source's frame is keyframe or not
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.time. <b>Note: HLS server need to set accuracy time-stamps</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push the a hardware compressed video buffer to broadcasting engine*

```
QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer,
0,
pStreamBuffer,
nStreamBufferLen,
TRUE,
0 );
```



# 11.5.8 QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER

## Introduction

The user can use this function to push compression audio stream buffer directly into broadcasting. Due to the stream buffer is already encoded, this function will not enable a software encoder.



In broadcasting, this function is For **Hardware encoder** only!

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
BYTE *	pStreamBuffer	IN	Specify the input source buffer
ULONG	nStreamBufferLen	IN	Specify the input source buffer's size
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.time. <a href="#">Note: HLS server need to set accuracy time-stamps</a>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push the a hardware compressed video buffer to broadcasting engine*

```
QCAP_SET_AUDIO_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer,
0,
pStreamBuffer,
nStreamBufferLen,
0 );
```

11.5.9

QCAP\_SET\_AUDIO\_MX\_BROADCAST\_SERVER\_MIXING\_UNCOMPRESSION\_BUFFER

11.5.9

QCAP\_SET\_AUDIO\_MX\_BROADCAST\_SERVER\_MIXING\_UNCOMPRESSION\_BUFFER\_EX

Introduction

The user can use this function to mix audio uncompressed buffer on audio preview callback in broadcasting. The user must set `QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER()` to mix audio recording first before calls `QCAP_SET_AUDIO_MX_BROADCAST_SERVER_UNCOMPRESSION_BUFFER()`.

Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iTrackNum	IN	Specify the audio track number, range 0-3
UINT	iMixNum	IN	Specify the audio mixer number
ULONG	nChannels	IN	Specify the total audio channels <b>Only in</b> <b>QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nBitsPerSample	IN	Specify the audio bits per sample <b>Only in</b> <b>QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nSampleFrequency	IN	Specify the audio sample frequency <b>Only in</b> <b>QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER_EX()</b>
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer size

Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

Examples

Example : To mix audio uncompressed buffer in broadcasting

```
QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER( pServer, 0, 0, 0,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER( pServer, 1, 0, 0,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

QCAP_SET_AUDIO_MX_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0, 0 );

QCAP_SET_AUDIO_MX_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0, 1 );
```

# 11.5.10 QCAP\_SET\_AUDIO\_MX\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER

## Introduction

The user can use this function to set audio broadcasting uncompressed buffer on audio preview callback in broadcasting.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iTrackNum	IN	Specify the audio track number, range 0-3
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock. <b>Note: HLS server need to set accuracy time-stamps</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To mix audio uncompressed buffer*

```
QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER( pServer, 0, 0,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

QCAP_SET_AUDIO_MX_BROADCAST_SERVER_MIXING_UNCOMPRESSION_BUFFER( pServer, 1, 0,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

QCAP_SET_AUDIO_MX_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0, 0 );

QCAP_SET_AUDIO_MX_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0, 1 );
```

### 11.5.90 QCAP\_SET\_AUDIO\_MX\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER

## Introduction

The user can call this function to push a broadcasting compressed audio buffer to audio mixer engine.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iTrackNum	IN	Specify the audio track number, range 0-3
BYTE *	pStreamBuffer	IN	Pointer to the source framebuffer
ULONG	nStreamBufferLen	IN	Specify the length of source framebuffer
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock. <b>Note: HLS server need to set accuracy time-stamps</b>

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To mix an audio compressed buffer*

```
QCAP_SET_AUDIO_MX_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer, 0, 0,
pStreamBuffer,
nStreamBufferLen,
0.0f );
```

# 11.5.90 QCAP\_GET\_VIDEO\_BROADCAST\_SERVER\_NETWORK\_STATUS

# 11.5.90 QCAP\_GET\_AUDIO\_BROADCAST\_SERVER\_NETWORK\_STATUS

### Introduction

The user can call this function to detect if audio/video network is busy or not. This function is called before the **QCAP\_SET\_VIDEO\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER/QCAP\_SET\_AUDIO\_BROADCAST\_SERVER\_COMPRESSION\_BUFFER** to ensure the buffer will be sent immediately.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
BOOL *	plsTransferBusy	OUT	The pointer to the network busy flag

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

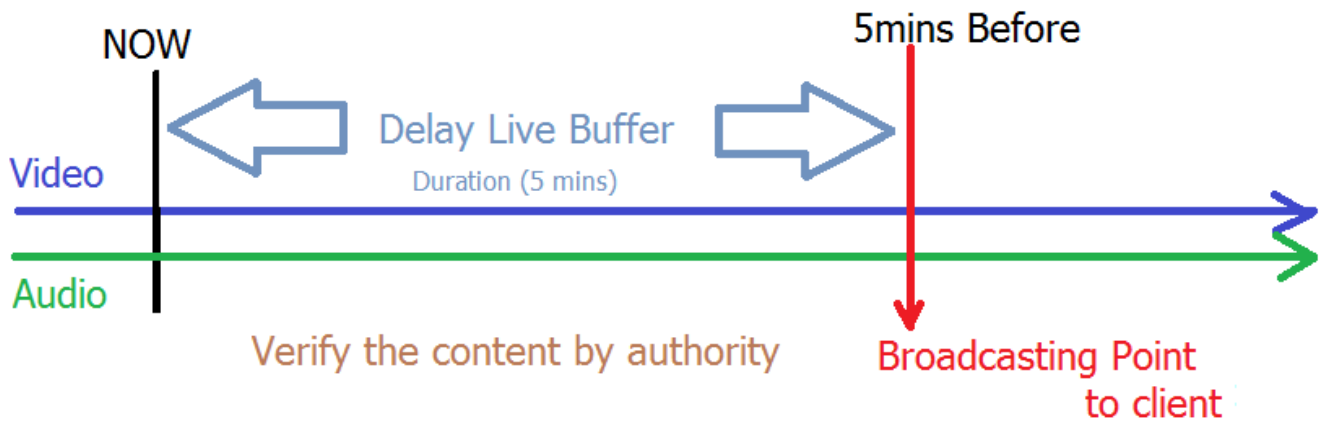
*Example : To get the audio/video network busy status*

```
QCAP_GET_VIDEO_BROADCAST_SERVER_NETWORK_STATUS( pServer, 0, &busy_video );

QCAP_GET_AUDIO_BROADCAST_SERVER_NETWORK_STATUS( pServer, 0, &busy_audio );
```

## 11.6 Server Delay Live Streaming Functions

### Introduction



This section provides the function that can use a feature called **Network Delay Live Streaming** during the broadcasting. In delay live buffer, the live broadcasting audio/video will be delayed/buffered a certain duration and stored in the caches of buffer before the live stream content is output. This way user can verify the content of the audio/video and decide what content must be cut off. It is useful when a live video needs to be preprocessed and follow authority policies.

## 11.6.1 QCAP\_CLEAR\_VIDEO\_BROADCAST\_SERVER\_DELAY\_LIVE\_BUFFER

## 11.6.2 QCAP\_CLEAR\_AUDIO\_BROADCAST\_SERVER\_DELAY\_LIVE\_BUFFER

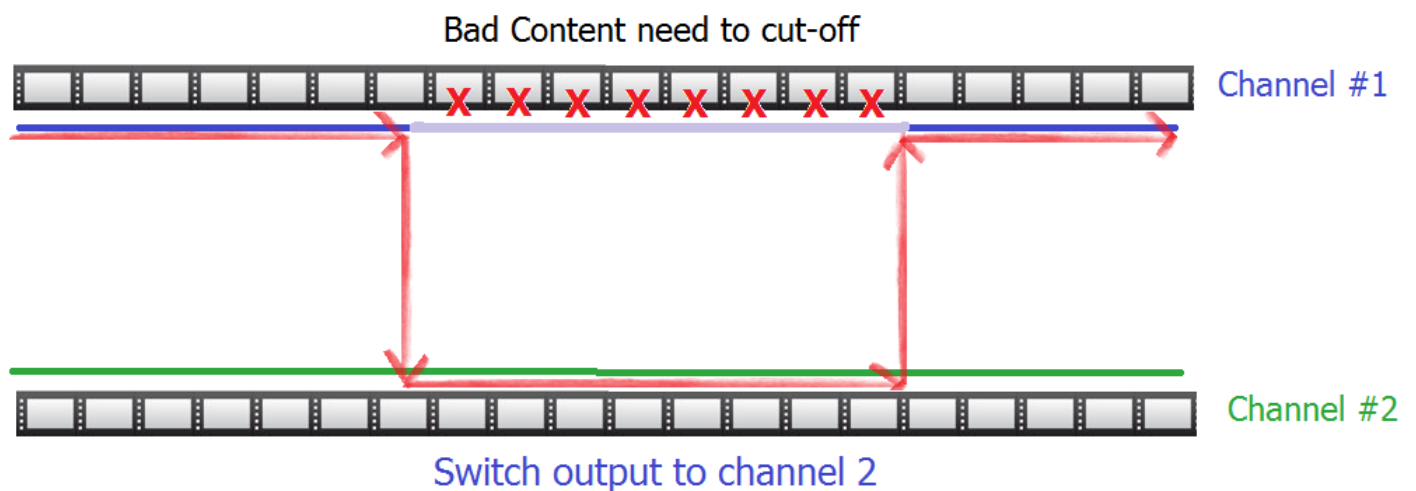
## 11.6.2 QCAP\_CLEAR\_AUDIO\_MX\_BROADCAST\_SERVER\_DELAY\_LIVE\_BUFFER

### Introduction

The **Delay Live Buffer feature** will cache/buffer a duration of audio/video stream for a content verification process. Which means the server-side can **PREVIEW** few seconds of content before sending it to the client-side. If the content is decided not to transmit to the audiences by the authority, this function can immediately clear the video streams cached in delay live buffer in broadcasting. Then client-side won't able to see the content in the delayed live buffer.

After the delay live buffer is cleared, if user wants to continue the broadcasting, user **MUST** to fill new audio/video frame data into it. The delay live buffer is hungry for user to feed more data, so the return value tells if the delay live buffer have enough frames:

- if returns **QCAP\_RS\_ERROR\_NEED\_MORE\_DATA** - The delay live buffer is not FULL of frames, user needs to feed more audio/video frames
- if returns **QCAP\_RS\_SUCCESSFUL** - The delay live buffer is FULL now, the broadcasting with **Delay-Live Buffer Feature** can then continue.



In practical application, there are usually 2 channels of input sources at the same time:

1. If the live broadcasting source (channel #1) is considered not properly to broadcasting, then
2. Call **QCAP\_CLEAR\_AUDIO/VIDEO\_BROADCAST\_SERVER\_DELAY\_LIVE\_BUFFER()** to clear audio/video in delay cached buffer to prevent it from sending out
3. Switch broadcasting stream to another input source (channel #2)
4. Wait for the channel #1 content became acceptable
5. Call **QCAP\_CLEAR\_AUDIO/VIDEO\_BROADCAST\_SERVER\_DELAY\_LIVE\_BUFFER()** again to fill channel #1 data frames into the Delay Live Buffer, until it returns **QCAP\_RS\_SUCCESSFUL**
6. Switch broadcasting stream back to the first input source (channel #1)

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iTrackNum	IN	Specify the audio track number, range 0-3 <b>Only in</b> <b>QCAP_CLEAR_AUDIO_MX_BROADCAST_SERVER_DELAY_LIVE_BUFFER()</b>
BOOL	bEnableClear	IN	The flag to enable buffer clear
BYTE *	pStreamBuffer	IN	Pointer to the source framebuffer
ULONG	nStreamBufferLen	IN	Specify the length of source framebuffer
ULONG	blsKeyFrame	IN	





[illegible]

## 11.6.3 QCAP\_SET\_SESSION\_BROADCAST\_SERVER\_PROPERTY

### Introduction

The user can use this function to set **Video Delay Live Duration** property of the i-th session in broadcasting.

The *ppSessionName* parameter could change the session name in for the client URL:

<rtmp://127.0.0.1/live/NewSessionName>

For **HLS** server user, the session name also needs ".m3u8" extension as part of URL:

<http://hls/NewSessionName.m3u8>

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
CHAR *	pszSessionName	IN	<b>default "session%d.mpg"</b> Specify the server session name
ULONG	nVideoDelayLiveDuration	IN	Specify video delay live duration time (ms)
ULONG	nAudioDelayLiveDuration	IN	Specify audio delay live duration time (ms)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## 11.6.4 QCAP\_GET\_SESSION\_BROADCAST\_SERVER\_PROPERTY

### Introduction

The user can use this function to get **Video Delay Live Duration** property of the i-th session in broadcasting.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
CHAR **	ppszSessionName	OUT	Pointer to the server session name
ULONG *	pVideoDelayLiveDuration	OUT	Pointer to the current video delay live duration time (ms)
ULONG *	pAudioDelayLiveDuration	OUT	Pointer to the current audio delay live duration time (ms)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set/get the Delay live duration property in audio/video broadcasting*

```
QCAP_SET_SESSION_BROADCAST_SERVER_PROPERTY( pServer, 0, nSessionName, 5000, 5000 );

QCAP_GET_SESSION_BROADCAST_SERVER_PROPERTY( pServer, 0, &nSessionName, &nVideoDelayLiveDuration, &nAudioDelayLiveDuration );
```

# 11.7 Server Snapshot Functions

## Introduction

This chapter will help to take a snapshot of your current broadcast server video stream. Follow this guide to take a snapshot of your whole video display, or any section of the video you want. The user can save the snapshot to a **BMP/JPG**, or to trigger a snapshot to get the image stream buffer from a callback function without saving it to disk.

### 11.7.1 QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_BMP

### 11.7.2 QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_BMP\_EX

## Introduction

This function takes a snapshot of video and saves to **BMP** 24bit or 32bit file in the broadcasting video.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_BMP\_EX()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.BMP24" → save to 24bit <b>BMP</b> "Filename.BMP32 or BMP" → save to 32bit <b>BMP</b> "BMP24" → To snapshot stream in callback only (no save to file.) "BMP32"/"BMP" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Take a snapshot and save to file in broadcasting video*

```
QCAP_SNAPSHOT_BROADCAST_SERVER_BMP( pServer, 0,  
                                     "C:/PICTURE1.BMP24" );  
  
QCAP_SNAPSHOT_BROADCAST_SERVER_BMP_EX( pServer, 0,  
                                       "C:/PICTURE1.BMP",  
                                       10, 40,  
                                       1900, 1000,  
                                       720, 480 );
```

### 11.7.3 QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_JPG

### 11.7.4 QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_JPG\_EX

#### Introduction

This function takes a snapshot of video and saves to **JPEG** image file format in the broadcasting video.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_JPG\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.JPG" → To snapshot to <b>JPEG</b> image file "JPG" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX()</b>
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Take a snapshot in broadcasting video, cropping a rectangle area and scale to 720x480 in **JPEG***

```
QCAP_SNAPSHOT_BROADCAST_SERVER_JPG( pServer,0,
                                     "C:/PICTURE1.JPG",
                                     100 );

QCAP_SNAPSHOT_BROADCAST_SERVER_JPG_EX( pServer, 0,
                                     "C:/PICTURE1.JPG",
                                     10, 40,
                                     1900, 1000,
                                     720, 480,
                                     100 );
```

## 11.8 Server OSD Functions

### Introduction

An on-screen display (OSD) are control functions on a video screen that allows you to draw text fields, overlapped pictures, or put customer image buffer with blending or color key effect.

For more detailed parameters descriptions, please refer to **Chapter 7 OSD Function API**.

### 11.8.1 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_TEXT

### 11.8.2 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_TEXT\_EX

### 11.8.3 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_TEXT\_W

### 11.8.4 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_TEXT\_EX\_W

### Introduction

The user can use this function to create a text field objects used for on-screen display in any video stream session.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_TEXT()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to auto calculate width.
INT	h	IN	Specify the height of OSD output Set -1 to auto calculate height.
CHAR * WSTRING	pszString pwszString	IN	Specify to display the text of OSD output Support wide character string
CHAR * WSTRING	pszFontFamilyName pwszFontFamilyName	IN	Specify the font name used to display the text of OSD output Support wide character string
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output : QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
DWORD	dwFontColor	IN	Specify the font color used to display the text of OSD output
DWORD	dwBackgroundColor	IN	Specify the background color used to display the text of OSD output
DWORD	dwBorderColor	IN	Specify the border color of the text <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_TEXT_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width in pixel, set 0 to disable border. <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_TEXT_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255



### 11.8.6 QCAP\_GET\_OSD\_BROADCAST\_SERVER\_TEXT\_BOUNDARY\_W

The user can use this function to get a size of OSD string in any video stream session.

For more detailed parameters descriptions, please refer to **QCAP\_GET\_OSD\_TEXT\_BOUNDARY()**.

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
CHAR * WSTRING	pszString pwszString	IN	Specify to display the text of OSD output Support wide character string
CHAR * WSTRING	pszFontFamilyName pwszFontFamilyName	IN	Specify the font name used to display the text of OSD output Support wide character string
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output : QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
ULONG *	pBoundaryWidth	OUT	Pointer to the boundary width
ULONG *	pBoundaryHeight	OUT	Pointer to the boundary height

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

*Example : Get the boundary width/height of the OSD text in broadcasting video*

[illegible]



# 11.8.7 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_PICTURE

## Introduction

This OSD function displays one or more **BMP/JPG/PNG/GIF/EDL.INI** on top of any video stream session.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_PICTURE()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to use original picture width.
INT	h	IN	Specify the height of OSD output Set -1 to use original picture height.
CHAR *	pszFilePathName	IN	Specify the image file name to display in OSD Supported extensions: " <b>BMP</b> " as 24/32-bit, " <b>JPG</b> " and " <b>PNG</b> " Supported animation by <b>GIF,EDL.INI</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a half-transparent PNG on the top of captured video stream in broadcasting video*

```
QCAP_SET_OSD_BROADCAST_SERVER_PICTURE( pServer, 0,
                                         0,
                                         0,
                                         0,
                                         -1, -1,
                                         "C:/SAMPLE.PNG",
                                         128,
                                         QCAP_SEQUENCE_STYLE_FOREMOST );
```

## 11.8.8 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_BUFFER

## 11.8.8 QCAP\_SET\_OSD\_BROADCAST\_SERVER\_BUFFER\_EX

### Introduction

The user can use this function to create a framebuffer object used for on-screen display in any session.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_BUFFER()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output
INT	h	IN	Specify the height of OSD output
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameWidth	IN	Specify the width of frame contained in a buffer
ULONG	nFrameHeight	IN	Specify the height of frame contained in a buffer
ULONG	nFramePitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate by width and color space format.
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX()</b>
DWORD	dwBorderColor	IN	Specify the border color <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width <b>Only in QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	<b>default 0xFFFFFFFF</b> Specify the key color of broadcast server OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY) 2. 0x00FF0000 (MASK BLUE) 3. 0x0000FF00 (MASK GREEN)
ULONG	nKeyColorThreshold	IN	<b>default 25</b> Specify the threshold of key color, the range from 0 to 128
ULONG	nKeyColorBlurLevel	IN	<b>default 2</b> Specify the blur level, range 0-2
BOOL	bKeyColorSpillSuppress	IN	<b>default TRUE</b> Specify the color spill suppress value

ULONG	nKeyColorSpillSuppressThreshold	IN	<b>default 22</b> Specify the threshold value of color spill suppress
BYTE *	pMaskBuffer	IN	<b>default NULL</b> Specify a mask OSD background buffer, default null
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a picture directly from a framebuffer on the video display in broadcasting*

```

QCAP_SET_OSD_BROADCAST_SERVER_BUFFER( pServer, 0,
0,
0, 0,
1920, 1080,
QCAP_COLORSPACE_TYEP_YUY2,
pFrameBuffer,
800, 600,
0,
128,
0xFFFFFFFF,
25, 2,
NULL,
QCAP_SEQUENCE_STYLE_FOREMOST );

QCAP_SET_OSD_BROADCAST_SERVER_BUFFER_EX( pServer, 0,
0,
0, 0,
1920, 1080,
QCAP_COLORSPACE_TYEP_YUY2,
pFrameBuffer,
800, 600,
0,
0,0,0,0, 0, 0,
128,
0xFFFFFFFF,
25, 2, FALSE,
NULL,
QCAP_SEQUENCE_STYLE_FOREMOST );

```

### 11.8.9 QCAP\_MOVE\_OSD\_BROADCAST\_SERVER\_OBJECT

## Introduction

The user can use this function to move the OSD object around the video window. It is useful to scroll the text string or picture on the video display window.



For more detailed parameters descriptions, please refer to **QCAP\_MOVE\_OSD\_OBJECT()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To scroll the OSD object from left to right in broadcasting video*

[illegible]

2

# 11.9 Server 3D functions

## Introduction

This section provides functions that can generate 3D video in broadcasting. The supported 3D video format are:

- Side-by-Side (SBS)
- Top-Bottom (TB)
- Line-by-Line (LBL)

This section contains 3D functions for:

#	3D Video Format Functions
1	QCAP_SET_VIDEO_3D_BROADCAST_SERVER_UNCOMPRESSION_BUFFER()
2	QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER() QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX() QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER() QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX()
3	QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER() QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX()

# 11.9.1 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_UNCOMPRESSION\_BUFFER

## Introduction

The user can use this function to push 3D uncompressed video buffer into a broadcasting engine. The user can use *bForceKeyFrame* parameter to push keyframe immediately.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nStereoDisplayMode	IN	<b>default QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE</b> Specify 3D display mode: QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE QCAP_3D_STEREO_DISPLAY_MODE_LEFT_ONLY QCAP_3D_STEREO_DISPLAY_MODE_RIGHT_ONLY
BOOL	bLeftRightSwap	IN	<b>default FALSE</b> Swap the the broadcast video left/right outputs
BOOL	bForceKeyFrame	IN	<b>default FALSE</b> Enforce the input source's frame is keyframe, the default FALSE
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock. <b>Note:</b> <b>HLS</b> server need to set accuracy time-stamps

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To set a 3D uncompressed buffer to broadcasting*

```
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0,
    0,
    FALSE,
    FALSE,
    0 );
```

## 11.9.2 Server 3D L/R Buffer Functions

### 11.9.2.1 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_L\_UNCOMPRESSION\_BUFFER

### 11.9.2.2 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_L\_UNCOMPRESSION\_BUFFER\_EX

### 11.9.2.3 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_R\_UNCOMPRESSION\_BUFFER

### 11.9.2.4 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_R\_UNCOMPRESSION\_BUFFER\_EX

#### Introduction

The user can use this function to push 3D stereo uncompressed video buffer into a broadcasting engine. To push Left side/and Right side of stereo video frames, in result a 3D Side-by-Side video format in a broadcasting video.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the 3D of left input source buffer's width
ULONG	nHeight	IN	Specify the 3D of left input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the 3D of left input source buffer
ULONG	nFrameBufferLen	IN	Specify the 3D of left input source buffer's size
ULONG	nCropX	IN	Specify the x-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX</b>
ULONG	nCropW	IN	Specify the width of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX</b>
ULONG	nCropH	IN	Specify the height of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX</b>
ULONG	nScaleStyle	IN	default QCAP_SCALE_STYLE_STRETCH specify the scale styles: QCAP_SCALE_STYLE_STRETCH QCAP_SCALE_STYLE_FIT QCAP_SCALE_STYLE_FILL <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX(),</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX</b>

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.



## Examples

*Example 1: To push left/right of 3D stereo video frame in broadcasting*

```
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER( pServer,  
    0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720,  
    pFrameBuffer,  
    nFrameBufferLen);  
  
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER( pServer,  
    0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280,  
    720,  
    pFrameBuffer,  
    nFrameBufferLen );
```

*Example 2: To push left/right of 3D stereo video frame with cropping in broadcasting*

```
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_L_UNCOMPRESSION_BUFFER_EX( pServer,  
    0, QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720,  
    pFrameBuffer,  
    nFrameBufferLen,  
    100, 150,  
    720, 480,  
    QCAP_SCALE_STYLE_STRETCH );  
  
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_R_UNCOMPRESSION_BUFFER_EX( pServer,  
    0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720, pFrameBuffer, nFrameBufferLen,  
    100, 150, 720, 480, QCAP_SCALE_STYLE_STRETCH );
```

## 11.9.3 Server 3D Stereo Buffer Functions

### 11.9.3.1 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_STEREO\_UNCOMPRESSION\_BUFFER

### 11.9.3.2 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_SERVER\_STEREO\_UNCOMPRESSION\_BUFFER\_EX

#### Introduction

The user can use this function to push 3D uncompressed video buffer into broadcasting. The buffer will pass to the encoder to generate its bitstream, then sent to the network as the broadcasting source.

The cropping parameters can crop a display region and scale to the target video size.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8, max is 0-8
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the 3D input source buffer's width
ULONG	nHeight	IN	Specify the 3D input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the 3D input source buffer
ULONG	nFrameBufferLen	IN	Specify the 3D input source buffer's size
ULONG	nCropX	IN	Specify the x-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropW	IN	Specify the width of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nCropH	IN	Specify the height of crop of 3D video <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nScaleStyle	IN	<b>default QCAP_SCALE_STYLE_STRETCH</b> specify the scale styles: QCAP_SCALE_STYLE_STRETCH QCAP_SCALE_STYLE_FIT QCAP_SCALE_STYLE_FILL <b>Only in</b> <b>QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nStereoBufferType	IN	<b>default QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE</b> Specify 3D stereo mode: QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE QCAP_3D_STEREO_BUFFER_TOP_BOTTOM QCAP_3D_STEREO_BUFFER_LINE_BY_LINE

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Push a 3D stereo video stream to broadcast engine, and output 3D Side-by-Side format.*

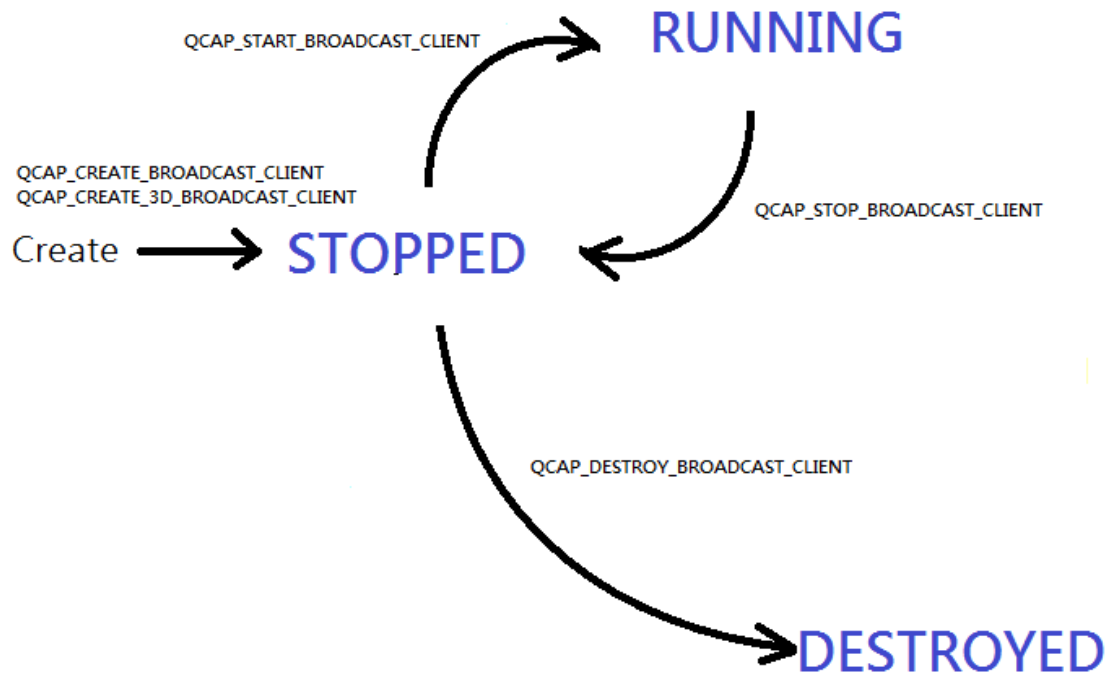
```
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER( pServer, 0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720,  
    pFrameBuffer,  
    nFrameBufferLen,  
    QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE );  
  
QCAP_SET_VIDEO_3D_BROADCAST_SERVER_STEREO_UNCOMPRESSION_BUFFER_EX(pServer, 0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720,  
    pFrameBuffer,  
    nFrameBufferLen,  
    100, 150,  
    720, 480,  
    QCAP_SCALE_STYLE_STRETCH,  
    QCAP_3D_STEREO_BUFFER_SIDE_BY_SIDE);
```

## 11.10 Client Major Functions

### Introduction

The client-side major function can create a broadcasting client object (include 3D client), and start the client to receiving audio/video streams, stop after client video is done, and destroy the client object to release the system resource.

The life cycle of the client object is:



# 11.10.1 QCAP\_CREATE\_BROADCAST\_CLIENT

# 11.10.2 QCAP\_CREATE\_BROADCAST\_CLIENT\_EX

## Introduction

The user can use this function to create a broadcast client object. The associated windows handle can be used to display the live broadcasting video. Currently, the client creates function supports for streaming servers:

- **RTSP** server
- **RTMP** server (build with QCAP SDK)
- **TS over TCP** server
- **TS over UDP** server



For more detailed parameters descriptions, please refer to **QCAP\_CREATE()**.

## Parameters

type	parameter	I/O	descriptions
UINT	iCliNum	IN	Specify broadcast client number to get client parameters start from 0 to 63
CHAR *	pszURL	IN	The URL string for client to access the media server e.g. <i>"rtsp://root:1234@127.0.0.1:554/session0.mpg"</i>
PVOID *	ppClient	OUT	Handle of the client object
ULONG	nDecoderType	IN	<b>default QCAP_DECODER_TYPE_SOFTWARE</b> Specify query the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
CHAR *	pszNetworkAdapterIP	IN	The IP address for the client network adapter to do transmission <b>Only in QCAP_CREATE_BROADCAST_CLIENT_EX()</b>
HWND	hAttachedWindow	IN	<b>default NULL</b> Handle of the window to show the broadcasting video
BOOL	bThumbDraw	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer
BOOL	bMaintainAspectRatio	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example: Create an RTSP client to a server URL*

```
QCAP_CREATE_BROADCAST_CLIENT( 0,  
    QCAP_DECODER_TYPE_SOFTWARE,  
    "rtsp://root:1234@127.0.0.1:554/session0.mpg",  
    &pClient,  
    hWnd,  
    TRUE,  
    FALSE );
```

```
QCAP_CREATE_BROADCAST_CLIENT_EX( 0,  
    QCAP_DECODER_TYPE_SOFTWARE,  
    "rtsp://root:1234@127.0.0.1:554/session0.mpg",  
    &pClient,  
    "10.26.64.1"  
    hWnd,  
    TRUE,  
    FALSE );
```

## 11.10.3 QCAP\_CREATE\_3D\_BROADCAST\_CLIENT

## 11.10.4 QCAP\_CREATE\_3D\_BROADCAST\_CLIENT\_EX

### Introduction

The user can use this function to create a 3D broadcast client object. The associated Left/Right windows handles can be used to display the 3D live broadcasting stereo video.



For more detailed parameters descriptions, please refer to **QCAP\_CREATE()**.

### Parameters

type	parameter	I/O	descriptions
UINT	iCliNum	IN	Specify broadcast client number to get client parameters start from 0 to 63
CHAR *	pszURL	IN	The URL string for client to access the media server e.g. <i>"rtsp://root:1234@127.0.0.1:554/session0.mpg"</i>
PVOID *	ppClient	OUT	Handle of the client object
ULONG	nDecoderType	IN	<b>default QCAP_DECODER_TYPE_SOFTWARE</b> Specify query the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
CHAR *	pszNetworkAdapterIP	IN	The IP address for the client network adapter to do transmission <b>Only in QCAP_CREATE_3D_BROADCAST_CLIENT_EX()</b>
HWND	hAttachedWindowL	IN	<b>default NULL</b> Handle of the window to show the 3D left-side video
BOOL	bThumbDrawL	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer of 3D left-side video
BOOL	bMaintainAspectRatioL	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio of 3D left-side
HWND	hAttachedWindowR	IN	<b>default NULL</b> Handle of the window to show the 3D right-side video
BOOL	bThumbDrawR	IN	<b>default FALSE</b> Enable/Disable the thumb draw renderer of 3D right-side video
BOOL	bMaintainAspectRatioR	IN	<b>default FALSE</b> Enable/Disable the maintain aspect ratio of 3D right-side

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Create an RTSP 3D client to a server URL, and display stereo video in 2 windows*

```
QCAP_CREATE_3D_BROADCAST_CLIENT( 0,
    "rtsp://root:1234@127.0.0.1:554/session0.mpg",
    &pClient,
    QCAP_DECODER_TYPE_SOFTWARE,
    hWnd_L, TRUE, FALSE,
    hWnd_R, TRUE, FALSE );

QCAP_CREATE_3D_BROADCAST_CLIENT_EX( 0,
    "rtsp://root:1234@127.0.0.1:554/session0.mpg",
    &pClient,
    QCAP_DECODER_TYPE_SOFTWARE,
    "10.26.64.10",
    hWnd_L, TRUE, FALSE,
    hWnd_R, TRUE, FALSE );
```

# 11.10.5 QCAP\_START\_BROADCAST\_CLIENT

## Introduction

This function can start a broadcast client to receive audio/video stream from a server.

The user can select which Internet Protocol to connect to the broadcasting server.

If the bit rate of video broadcasting property is larger than 12Mbps, we recommend user to use TCP protocol to streaming.



If DelayQueueDuration is set to 0, we offer .TS file to One VOD for HLS streaming.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG	nProtocol	IN	<b>default QCAP_BROADCAST_PROTOCOL_TCP</b> Broadcasting protocols: QCAP_BROADCAST_PROTOCOL_UDP QCAP_BROADCAST_PROTOCOL_TCP QCAP_BROADCAST_PROTOCOL_HTTP <b>If BPS &gt;= 12Mbps is suggested to use QCAP_BROADCAST_PROTOCOL_TCP</b>
ULONG	nReconnectionTimeout	IN	<b>default 3000</b> Specify the timeout duration (ms) of the reconnection
ULONG	nDelayQueueDuration	IN	<b>default 0</b> Specify delay queue duration time (ms)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Start a broadcasting client object by using UDP protocol*

```
QCAP_START_BROADCAST_CLIENT( pClient, QCAP_BROADCAST_PROTOCOL_UDP );
```



# 11.10.6 QCAP\_STOP\_BROADCAST\_CLIENT

## Introduction

This function can stop a broadcasting client from running. The receiving audio/video streams process will be stopped. The user can start it again by **QCAP\_START\_BROADCAST\_CLIENT()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To stop a broadcasting client from running*

```
QCAP_STOP_BROADCAST_CLIENT( pClient );
```

# 11.10.7 QCAP\_DESTROY\_BROADCAST\_CLIENT

## Introduction

This function can destroy broadcast client object and release its system resources.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Destroy a broadcasting client object*

```
QCAP_DESTROY_BROADCAST_CLIENT( pClient );
```

# 11.11 Client Custom Property Functions

## Introduction

This section provides the function that can set a custom property in client-side, the property name is a string identify a certain property name and its corresponding custom value.

### 11.11.1 QCAP\_SET\_BROADCAST\_CLIENT\_CUSTOM\_PROPERTY

#### Introduction

This function can set or change the value of a certain custom property to broadcast client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
CHAR *	pszProperty	IN	Specify client custom property interface
CHAR *	pszValue	IN	Specifies the property value

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Change the custom property "Record\_Start" value to "Start111"*

```
QCAP_SET_BROADCAST_CLIENT_CUSTOM_PROPERTY( pClient, "Record_Start", "Start111" );
```

### 11.11.2 QCAP\_GET\_BROADCAST\_CLIENT\_CUSTOM\_PROPERTY

#### Introduction

This function can get the value of a certain custom property from a broadcast client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
CHAR *	pszProperty	IN	Specify get client custom property interface
CHAR **	ppszValue	OUT	Pointer the specify property value. It cannot be NULL.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get the value of custom property "Record\_Start"*

```
char *szValue = NULL;

QCAP_GET_BROADCAST_CLIENT_CUSTOM_PROPERTY( pClient, "Record_Start", &szValue );
```

# 11.11.3 QCAP\_SET\_BROADCAST\_CLIENT\_OUTPUT\_STREAMS

## Introduction

This function can set the Audio / Video stream's PID to a broadcast client.



This function is used for **TS over IP** only!

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG	nVideoStream_PID	IN	Specify the video PID for client side
ULONG	nAudioStream_PID	IN	Specify the audio PID for client side

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the Audio / Video PID for a broadcast client*

```
QCAP_SET_BROADCAST_CLIENT_OUTPUT_STREAMS( pClient, 0x01, 0x02 );
```

# 11.11.4 QCAP\_GET\_BROADCAST\_CLIENT\_OUTPUT\_STREAMS

This function can get the Audio / Video stream's PID from a broadcast client.



This function is used for **TS over IP** only!

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	
ULONG *	pVideoStream_PID	OUT	Specify the pointer to video PID for client side
ULONG *	pAudioStream_PID	OUT	Specify the pointer to video PID for client side

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the Audio / Video PID for a broadcast client*

```
QCAP_GET_BROADCAST_CLIENT_OUTPUT_STREAMS( &pVideoStream_PID, &pAudioStream_PID );
```

# 11.12 Client Video Functions

## Introduction

This section provides functions that can diagnostic video streaming status, set the clipping area or crop rectangle, and control the video flipping mode in both vertical and horizontal way of video broadcast client.

### 11.12.1 QCAP\_DIAGNOSE\_VIDEO\_BROADCAST\_CLIENT\_STREAM\_STATUS

#### Introduction

This function can retrieve and diagnostic the video stream output status.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
BOOL *	plsStill	OUT	Indicate if the video stream is still available

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : To get the video streaming status in client side*

```
BOOL status=0;

QCAP_DIAGNOSE_VIDEO_BROADCAST_CLIENT_STREAM_STATUS( pClient, &status );
```

# 11.12.1 QCAP\_SET\_VIDEO\_BROADCAST\_CLIENT\_REGION\_DISPLAY

## Introduction

This function can set the clipping area of client-side video, the crop region will be scaled to capture device display output.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG	nCropX	IN	Specify the x-coordinate of the crop region display
ULONG	nCropY	IN	Specify the y-coordinate of the crop region display
ULONG	nCropW	IN	Specify the width of crop region display
ULONG	nCropH	IN	Specify the height of crop region display

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set crop region x,y(0,0) to w,h(1280x720) to be displayed in client video*

```
QCAP_SET_VIDEO_BROADCAST_CLIENT_REGION_DISPLAY( pClient, 0, 0, 1280, 720 );
```

# 11.12.2 QCAP\_GET\_VIDEO\_BROADCAST\_CLIENT\_REGION\_DISPLAY

## Introduction

This function can get the current region clipping area parameters of client video.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG *	pCropX	OUT	Pointer to the x-coordinate of the crop region display
ULONG *	pCropY	OUT	Pointer to the y-coordinate of the crop region display
ULONG *	pCropW	OUT	Pointer to the horizontal width of crop region display
ULONG *	pCropH	OUT	Pointer to the vertical height of crop region display

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the clipping area setting in client video*

```
QCAP_GET_VIDEO_BROADCAST_CLIENT_REGION_DISPLAY( pClient, &nCropX, &nCropY, &nCropW, &nCropH );
```

### 11.12.3 QCAP\_SET\_VIDEO\_BROADCAST\_CLIENT\_MIRROR

## Introduction

This function can set the video flipping mode in both vertical and horizontal way of video broadcast client.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
BOOL	bHorizontalMirror	IN	Enable/Disable horizontal of mirror
BOOL	bVerticalMirror	IN	Enable/Disable vertical of mirror

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example :Flipping video up-down & left-right in client video*

```
QCAP_SET_VIDEO_BROADCAST_CLIENT_MIRROR( pClient, TRUE, TRUE );
```

#### 11.12.4 QCAP\_GET\_VIDEO\_BROADCAST\_CLIENT\_MIRROR

## Introduction

This function can get the flipping mode of the video display of video broadcast client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
BOOL *	pHorizontalMirror	OUT	Returned the current horizontal of mirror
BOOL *	pVerticalMirror	OUT	Returned the current vertical of mirror

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the video flipping mode in client video*

[illegible]

# 11.13 Client Audio Functions

## Introduction

This is section contains the audio property to diagnostic audio streaming status, set in the device, such as sound renderer can decide which output device to play the sound, and it's volume setting in the broadcast client receiving.

## 11.13.1 QCAP\_DIAGNOSE\_AUDIO\_BROADCAST\_CLIENT\_STREAM\_STATUS

### Introduction

This function can retrieve and diagnostic the audio stream output status.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
double *	pVolumeDB_L	OUT	Indicate the audio L volume in dB, range -100-0 (dB)
double *	pVolumeDB_R	OUT	Indicate the audio L volume in dB, range -100-0 (dB)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To get the audio streaming status in client side*

```
BOOL L_vol=0, R_vol=0;

QCAP_DIAGNOSE_AUDIO_BROADCAST_CLIENT_STREAM_STATUS( pClient, &l_vol, &r_vol );
```

# 11.13.1 QCAP\_SET\_AUDIO\_BROADCAST\_CLIENT\_SOUND\_RENDERER

## Introduction

This function can set current sound renderer of sound output device used in client streaming.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_AUDIO\_SOUND\_RENDERER()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
UINT	iSoundNum	IN	Sound Renderer, default Renderer is 0

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the current audio recording input device to Sound Device #1 in client-side*

```
UINT nSounder = 1;

QCAP_SET_AUDIO_BROADCAST_CLIENT_SOUND_RENDERER( pClient, nSounder );
```

# 11.13.2 QCAP\_GET\_AUDIO\_BROADCAST\_CLIENT\_SOUND\_RENDERER

## Introduction

This function can get current sound renderer from available sound output device in client streaming.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
UINT *	pSoundNum	OUT	Pointer to the Sound number

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the current audio recording input device in client-side*

```
QCAP_GET_AUDIO_BROADCAST_CLIENT_SOUND_RENDERER( pClient, &nSounder );
```



# 11.13.3 QCAP\_SET\_AUDIO\_BROADCAST\_CLIENT\_VOLUME

## Introduction

This function can set current audio volume value in client streaming.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG	nVolume	IN	Audio volume, from 0-100, 0 is mute

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set current audio volume output to 50% in client-side*

```
QCAP_SET_AUDIO_BROADCAST_CLIENT_VOLUME( pClient, 50 );
```

# 11.13.4 QCAP\_GET\_AUDIO\_BROADCAST\_CLIENT\_VOLUME

## Introduction

This function can get current audio volume value in client streaming, range from 0 to 100.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG *	pVolume	OUT	Pointer to the audio volume

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get current audio volume output in client-side*

```
QCAP_GET_AUDIO_BROADCAST_CLIENT_VOLUME( pClient, &nVolume );
```

## 11.14 Client Recording Functions

### Introduction

When the client has connected to the broadcasting server, the video is ready to stream remotely and display on client window. This section provides recording functions to save video from the broadcasting source to video file.

### 11.14.1 QCAP\_START\_BROADCAST\_CLIENT\_RECORD

#### Introduction

When the client has connected to the broadcasting server, a user can use this function to start recording of broadcasting video in client-side. Unlike **Channel Recording**, the audio/video property or quality options..etc. were decided by the broadcasting server and cannot be changed in client-side.

By the way, if user call **Client OSD Functions** to draw OSD on client screen, those visual effects will not be recorded.



For more detailed parameters descriptions, please refer to **QCAP\_START\_SHARE\_RECORD()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
CHAR *	pszFilePathName	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3</b>
double	dVideoDelayTime	IN	<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time
double	dSegmentDurationTime	IN	<b>default 0.0</b> Specify the video segment duration time to split in recording file (in seconds)
ULONG	nSegmentDurationSizeKB	IN	<b>default 0</b> Specify the video segment duration to split int recording file (in Kilo-Bytes)

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Start to record a broadcasting video in client-side*

```
QCAP_START_BROADCAST_CLIENT_RECORD( pClient, "C:/REC.AVI" );

QCAP_START_BROADCAST_CLIENT_RECORD ( pClient,
    "D:/REC_$Y_$M_$D_$h_$m_$s_$i.mp4",
    0.0,
    0.0,
    10.0,    //split video by 10s duration
    0 );
```

# 11.14.2 QCAP\_START\_BROADCAST\_CLIENT\_TIMEShift\_RECORD

## Introduction

The user can use this function to start time-shifting broadcasting recording in client-side. That means a user can continue recording one broadcasting video while playing back the same video recording file. In another word, a user can playback the video in broadcasting recording on-the-fly, just like time-shifted.

This time-shifted functionality current supported **MP4** format only.



For more detailed parameters descriptions, please refer to:  
\* **QCAP\_START\_TIMEShift\_SHARE\_RECORD()**,  
\* **QCAP\_OPEN\_TIMEShift\_FILE\_EX()**

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
CHAR *	pszFilePathName	IN	Specify the file name for recording. 1. The file name supported input field descriptors: \$Y, \$M, \$D, \$h, \$m, \$s, \$i for start recording time %Y, %M, %D, %h, %m, %s, %i for stop recording time 2. The file extension decides the recording video format: AVI, <b>MP4</b> , ASF, WMV, FLV, TS, M3U8, SCF, WAV, MP3
PVOID *	ppPhysicalFileWriter	OUT	Handle of Physical File Writer object
double	dVideoDelayTime	IN	<b>default 0.0</b> Specify the video delay time
double	dAudioDelayTime	IN	<b>default 0.0</b> Specify the audio delay time

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Start a time-shifted broadcasting recording in client-side, and playback the video (still in recording)*

```
PVOID pPhysicalFileWriter = NULL;

PVOID pFile = NULL;

QCAP_START_BROADCAST_CLIENT_TIMEShift_RECORD( pClient,
                                               "D:/TIMESHIFT_RECORD.mp4",
                                               ppPhysicalFileWriter,
                                               0.0,
                                               0.0 );

QCAP_OPEN_TIMEShift_FILE_EX( pPhysicalFileWriter,
                             &pFile, QCAP_DECODER_TYPE_SOFTWARE, &nVideoFormat,
                             &nVideoWidth, &nVideoHeight, &dVideoFrameRate,
                             &nAudioFormat, &nAudioChannels,
                             &nAudioBitsPerSample, &nAudioSampleFrequency,
                             &dFileTotalDuationTimes,
                             &nFileTotalVideoFrames, &nFileTotalAudioFrames,
                             &nFileTotalMetadataFrames,
                             hWnd, 1 );

QCAP_PLAY_FILE( pFile ); //playback the video (still in time-shifted broadcasting recording)
```

## 11.14.3 QCAP\_PAUSE\_BROADCAST\_CLIENT\_RECORD

### Introduction

The user can use this function to pause broadcasting recording in client video.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Pause the broadcasting recording in client-side*

```
QCAP_PAUSE_BROADCAST_CLIENT_RECORD( pClient );
```

## 11.14.4 QCAP\_RESUME\_BROADCAST\_CLIENT\_RECORD

### Introduction

The user can use this function to resume a previously paused broadcasting recording.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To resume a previously paused broadcasting recording in client-side*

```
QCAP_RESUME_BROADCAST_CLIENT_RECORD( pClient );
```

# 11.14.5 QCAP\_STOP\_BROADCAST\_CLIENT\_RECORD

## Introduction

The user can use this function to stop broadcasting recording from a client.

This function is designed for both synchronous and asynchronous operations.



For more detailed parameters descriptions, please refer to **QCAP\_STOP\_SHARE\_RECORD()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Stop a broadcasting recording in client-side*

```
QCAP_STOP_BROADCAST_CLIENT_RECORD( pClient, 0 );
```

# 11.15 Client Snapshot Functions

## Introduction

This chapter will help to take a snapshot of your current broadcast client video stream. Follow this guide to take a snapshot of your whole video display, or any section of the video you want. The user can save a snapshot to a **BMP/JPG**, or to trigger a snapshot to get the image stream buffer from a callback function without saving it to disk.

### 11.15.1 QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_BMP

### 11.15.2 QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_BMP\_EX

## Introduction

This function takes a snapshot of video and saves to **BMP** 24bit or 32bit file in client-side.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_BMP\_EX()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.BMP24" → save to 24bit <b>BMP</b> "Filename.BMP32 or BMP" → save to 32bit <b>BMP</b> "BMP24" → To snapshot stream in callback only (no save to file.) "BMP32"/"BMP" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Take a snapshot and save to file in client-side video*

```
QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP( pClient,  
                                     "C:/PICTURE1.BMP24" ); // PICTURE1.BMP24 ->PICTURE1.BMP  
  
QCAP_SNAPSHOT_BROADCAST_CLIENT_BMP_EX( pClient,  
                                         "C:/PICTURE1.BMP",  
                                         10, 40,  
                                         1900, 1000,  
                                         720, 480 );
```

### 11.15.3 QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_JPG

### 11.15.4 QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_JPG\_EX

#### Introduction

This function takes a snapshot of video and saves to **JPEG** image file format in client-side.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_JPG\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.JPG" → To snapshot to <b>JPEG</b> image file "JPG" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX()</b>
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Take a snapshot in client video, cropping a rectangle area and scale to 720x480 in JPEG*

```
QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG ( pClient,  
    "C:/PICTURE1.JPG",  
    100 );  
  
QCAP_SNAPSHOT_BROADCAST_CLIENT_JPG_EX( pClient,  
    "C:/PICTURE1.JPG",  
    10, 40,  
    1900, 1000,  
    720, 480,  
    100 );
```



## 11.16 Client OSD Functions

### Introduction

An on-screen display (OSD) are control functions on a video screen that allows you to draw text fields, overlapped pictures, or put customer image buffer with blending or color key effect.

For more detailed parameters descriptions, please refer to **Chapter 7 OSD Function API**.

### 11.16.1 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_TEXT

### 11.16.2 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_TEXT\_EX

### 11.16.3 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_TEXT\_W

### 11.16.4 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_TEXT\_EX\_W

### Introduction

The user can use this function to create a text field objects used for on-screen display in any video session.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_TEXT()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify they-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to auto calculate width.
INT	h	IN	Specify the height of OSD output Set -1 to auto calculate height.
CHAR * WSTRING	pszString pwszString	IN	Specify to display the text of OSD output Support wide character string
CHAR * WSTRING	pszFontFamilyName pwszFontFamilyName	IN	Specify the font name used to display the text of OSD output Support wide character string
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output : QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
DWORD	dwFontColor	IN	Specify the font color used to display the text of OSD output
DWORD	dwBackgroundColor	IN	Specify the background color used to display the text of OSD output
DWORD	dwBorderColor	IN	Specify the border color of the text <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_TEXT_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width in pixel, set 0 to disable border. <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_TEXT_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
INT	nTextStartPosX	IN	<b>default 0</b> Specify the text scrolling start position X of the upper-left corner of OSD text



### 11.16.6 QCAP\_GET\_OSD\_BROADCAST\_CLIENT\_TEXT\_BOUNDARY\_W

```
QCAP_GET_OSD_BROADCAST_CLIENT_TEXT_BOUNDARY( pClient, 0,
"CH01",
"Arial",
QCAP_FONT_STYLE_BOLD,
12,
&BoundaryWidth,
&BoundaryHeight );
```

# 11.16.7 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_PICTURE

## Introduction

This OSD function displays one or more **BMP/JPG/PNG/GIF/EDL.INI** on top of any video stream.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_PICTURE()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to use original picture width.
INT	h	IN	Specify the height of OSD output Set -1 to use original picture height.
CHAR *	pszFilePathName	IN	Specify the image file name to display in OSD Supported extensions: " <b>BMP</b> " as 24/32-bit, " <b>JPG</b> " and " <b>PNG</b> " Supported animation by <b>GIF,EDL.INI</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a half-transparent PNG on the top of captured video stream in client-side*

```
QCAP_SET_OSD_BROADCAST_CLIENT_PICTURE( pClient, 0,
    0,
    0,
    -1,
    -1,
    "C:/SAMPLE.PNG",
    128,
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

## 11.16.8 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_BUFFER

## 11.16.8 QCAP\_SET\_OSD\_BROADCAST\_CLIENT\_BUFFER\_EX

### Introduction

The user can use this function to create a framebuffer object used for on-screen display in any session.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_BUFFER()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output
INT	h	IN	Specify the height of OSD output
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameWidth	IN	Specify the width of frame contained in a buffer
ULONG	nFrameHeight	IN	Specify the height of frame contained in a buffer
ULONG	nFramePitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate by width and color space format.
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX()</b>
DWORD	dwBorderColor	IN	Specify the border color <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width <b>Only in QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	<b>default 0xFFFFFFFF</b> Specify the key color of OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY) 2. 0x00FF0000 (MASK BLUE) 3. 0x0000FF00 (MASK GREEN)
ULONG	nKeyColorThreshold	IN	<b>default 25</b> Specify the threshold of key color, the range from 0 to 128
ULONG	nKeyColorBlurLevel	IN	<b>default 2</b> Specify the blur level, range 0-2
BOOL	bKeyColorSpillSuppress	IN	<b>default TRUE</b> Specify the color spill suppress value
ULONG	nKeyColorSpillSuppressThreshold	IN	

			<b>default 22</b> Specify the threshold value of color spill suppress
BYTE *	pMaskBuffer	IN	<b>default NULL</b> Specify a mask OSD background buffer, default null
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a picture directly from a framebuffer on the video display in client-side*

```

QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER( pClient, 0,
    0,
    0,
    1920, 1080,
    QCAP_COLORSPACE_TYEP_YUY2,
    pFrameBuffer,
    800, 600,
    0,
    128,
    0xFFFFFFFF,
    25, 2,
    NULL,
    QCAP_SEQUENCE_STYLE_FOREMOST );

QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER_EX( pClient, 0,
    0,
    0,
    1920, 1080,
    QCAP_COLORSPACE_TYEP_YUY2,
    pFrameBuffer,
    800, 600,
    0,
    0,0,0,0, 0, 0,
    128,
    0xFFFFFFFF,
    25, 2, FALSE,
    NULL,
    QCAP_SEQUENCE_STYLE_FOREMOST );

```

### 11.16.9 QCAP\_MOVE\_OSD\_BROADCAST\_CLIENT\_OBJECT

## Introduction

The user can use this function to move the OSD object around the video window. It is useful to scroll the text string or picture on the video display window.



For more detailed parameters descriptions, please refer to **QCAP\_MOVE\_OSD\_OBJECT()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To scroll the OSD object from left to right in client-side video*

[illegible]

6



# 11.17 Client 3D Functions

## Introduction

There are many 3D display formats in video rendering. This section provides functions for the 3D display mode function in client-side.

### 11.17.1 QCAP\_SET\_VIDEO\_3D\_BROADCAST\_CLIENT\_DISPLAY\_MODE

#### Introduction

The user can use this function to set current 3D display mode in a broadcasting client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG	nStereoDisplayMode	IN	<b>default QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE</b> Specify 3D display mode: QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE QCAP_3D_STEREO_DISPLAY_MODE_LEFT_ONLY QCAP_3D_STEREO_DISPLAY_MODE_RIGHT_ONLY
BOOL	bLeftRightSwap	IN	<b>default FALSE</b> Swap the client video left/right outputs

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set current 3D display mode to Top-Bottom in client-side video*

```
QCAP_SET_VIDEO_3D_BROADCAST_CLIENT_DISPLAY_MODE( pClient, QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM, FALSE );
```

### 11.17.2 QCAP\_GET\_VIDEO\_3D\_BROADCAST\_CLIENT\_DISPLAY\_MODE

#### Introduction

The user can use this function to get current 3D display mode in a broadcasting client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
ULONG *	pStereoDisplayMode	OUT	Pointer to the current Stereo Display Mode
BOOL *	pLeftRightSwap	OUT	Pointer to the current Left Right Swap

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

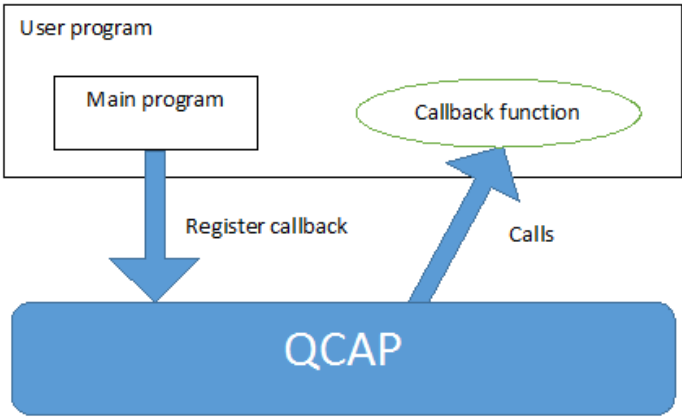
#### Examples

*Example : Get current 3D display mode in client-side video*

```
QCAP_GET_VIDEO_3D_BROADCAST_CLIENT_DISPLAY_MODE( pClient, &nStereoDisplayMode, &nLeftRightSwap );
```

# 11.18 Broadcast Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
<b>Server Callback Functions</b>	
QCAP_REGISTER_BROADCAST_SERVER_GET_CUSTOM_PROPERTY_CALLBACK	PF_BROADCAST_SERVER_GET_CUSTOM_PROPERTY_CALLBACK
QCAP_REGISTER_BROADCAST_SERVER_SET_CUSTOM_PROPERTY_CALLBACK	PF_BROADCAST_SERVER_SET_CUSTOM_PROPERTY_CALLBACK
QCAP_REGISTER_BROADCAST_SERVER_SNAPSHOT_DONE_CALLBACK	PF_BROADCAST_SERVER_SNAPSHOT_DONE_CALLBACK
QCAP_REGISTER_BROADCAST_SERVER_SNAPSHOT_STREAM_CALLBACK	PF_BROADCAST_SERVER_SNAPSHOT_STREAM_CALLBACK
QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK	PF_VIDEO_BROADCAST_SERVER_CALLBACK
QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK	PF_AUDIO_BROADCAST_SERVER_CALLBACK
QCAP_REGISTER_AUDIO_MX_BROADCAST_SERVER_CALLBACK	PF_AUDIO_MX_BROADCAST_SERVER_CALLBACK
QCAP_REGISTER_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK	PF_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK
QCAP_REGISTER_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK	PF_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK
QCAP_REGISTER_AUDIO_DECODER_MX_BROADCAST_SERVER_CALLBACK	PF_AUDIO_DECODER_MX_BROADCAST_SERVER_CALLBACK
QCAP_REGISTER_VIDEO_BROADCAST_SERVER_MEDIA_TIMER_CALLBACK	PF_VIDEO_BROADCAST_SERVER_MEDIA_TIMER_CALLBACK
QCAP_REGISTER_AUDIO_BROADCAST_SERVER_MEDIA_TIMER_CALLBACK	PF_AUDIO_BROADCAST_SERVER_MEDIA_TIMER_CALLBACK
<b>Client Callback Functions</b>	
QCAP_REGISTER_BROADCAST_CLIENT_CONNECTED_CALLBACK	PF_BROADCAST_CLIENT_CONNECTED_CALLBACK
QCAP_REGISTER_BROADCAST_CLIENT_RECORD_DONE_CALLBACK	PF_BROADCAST_CLIENT_RECORD_DONE_CALLBACK
QCAP_REGISTER_BROADCAST_CLIENT_RECORD_FAIL_CALLBACK	PF_BROADCAST_CLIENT_RECORD_FAIL_CALLBACK
QCAP_REGISTER_BROADCAST_CLIENT_SNAPSHOT_DONE_CALLBACK	PF_BROADCAST_CLIENT_SNAPSHOT_DONE_CALLBACK
QCAP_REGISTER_BROADCAST_CLIENT_SNAPSHOT_STREAM_CALLBACK	PF_BROADCAST_CLIENT_SNAPSHOT_STREAM_CALLBACK
QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK	PF_VIDEO_BROADCAST_CLIENT_CALLBACK
QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK	PF_AUDIO_BROADCAST_CLIENT_CALLBACK
QCAP_REGISTER_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK	PF_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK
QCAP_REGISTER_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK	PF_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK
QCAP_REGISTER_VIDEO_DECODER_3D_BROADCAST_CLIENT_CALLBACK	PF_VIDEO_DECODER_3D_BROADCAST_CLIENT_CALLBACK
QCAP_REGISTER_BROADCAST_CLIENT_MEDIAINFO_CALLBACK	PF_BROADCAST_CLIENT_MEDIAINFO_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

### 11.18.1 QCAP\_REGISTER\_BROADCAST\_SERVER\_GET\_CUSTOM\_PROPERTY\_CALLBACK

## Introduction

This callback function is called when a client sends a command to the server to query the custom property.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
<b><i>PF_BROADCAST_SERVER_GET_CUSTOM_PROPERTY_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_SERVER\_GET\_CUSTOM\_PROPERTY\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
CHAR *	pszProperty	The client custom property interface name
CHAR *	pszValue	<b>This is a OUTPUT parameter</b> The value of custom property
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function when client requests custom property information*

[illegible]

### 11.18.2 QCAP\_REGISTER\_BROADCAST\_SERVER\_SET\_CUSTOM\_PROPERTY\_CALLBACK

## Introduction

This callback function is called when a client sends a command to the server to change the custom property .

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
<b><i>PF_BROADCAST_SERVER_SET_CUSTOM_PROPERTY_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_SERVER\_SET\_CUSTOM\_PROPERTY\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
CHAR *	pszProperty	The client custom property interface name
CHAR *	pszValue	The value of custom property
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to know when the client changes the custom property.*

[illegible]

### 11.18.3 QCAP\_REGISTER\_BROADCAST\_SERVER\_SNAPSHOT\_DONE\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_BMP/JPG()** is completed. The user can get the path filename of the snapshot in the callback.

When uses asynchronous snapshot (*b/sAsync* = TRUE), it is useful to know when snapshot file is ready.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	The index of the i-th sessions, range 0-8
<b><i>PF_BROADCAST_SERVER_SNAPSHOT_DONE_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_SERVER\_SNAPSHOT\_DONE\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
CHAR *	pszFilePathName	pointer to the snapshot filename
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for snapshot done in server-side*

[illegible]

#### 11.18.4 QCAP\_REGISTER\_BROADCAST\_SERVER\_SNAPSHOT\_STREAM\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_BMP/JPG()** image stream is generated, then user can get image stream in buffer directly.

Fo. The users who want a snapshot without saving to a file, call **QCAP\_SNAPSHOT\_BROADCAST\_SERVER\_BMP/JPG()** by passing *pszFilePathName* file extension only(e.g. "BMP"), then a user can use this callback to retrieve the snapshot image in the buffer.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_BROADCAST_SERVER_SNAPSHOT_STREAM_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_SERVER\_SNAPSHOT\_STREAM\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
CHAR *	pszFilePathName	pointer to the snapshot filename
BYTE *	pStreamBuffer	Pointer to the image framebuffer
ULONG	nStreamBufferLen	Specify the length of image framebuffer
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for snapshot stream completion in server-side*

[illegible]

### 11.18.5 QCAP\_REGISTER\_VIDEO\_BROADCAST\_SERVER\_CALLBACK

## Introduction

This callback function will provide video compressed **H.264** data for the broadcast server developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_VIDEO_BROADCAST_SERVER_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_VIDEO\_BROADCAST\_SERVER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
BOOL	blsKeyFrame	Specify the input source's frame is keyframe or not
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

QCAP_RT_OK	callback already processed
QCAP_RT_FAIL	The framebuffer will be dropped

## Examples

*Example : Register a callback function to get broadcast server video callback*

[illegible]

### 11.18.6 QCAP\_REGISTER\_AUDIO\_BROADCAST\_SERVER\_CALLBACK

## Introduction

This callback function will provide audio compressed data for the broadcast server developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_AUDIO_BROADCAST_SERVER_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_BROADCAST\_SERVER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
PVOID	pUserData	Pointer to custom user data

***Return value of Callback***

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and its sound will be muted

## Examples

*Example : Register a callback function to get broadcast server audio callback*

[illegible]



### 11.18.21 QCAP\_REGISTER\_AUDIO\_MX\_BROADCAST\_SERVER\_CALLBACK

## Introduction

This callback function will provide audio mixer callback for broadcast server developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_AUDIO_MX_BROADCAST_SERVER_CALLBACK</i></b>	pCB	IN	Callback function

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_MX\_BROADCAST\_SERVER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
UINT	iTrackNum	The index of the i-th audio tracks, range 0-3
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Specify the input source buffer
ULONG	nStreamBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to get audio mixer callback in server-side*

[illegible]

### 11.18.7 QCAP\_REGISTER\_VIDEO\_DECODER\_BROADCAST\_SERVER\_CALLBACK

## Introduction

This callback function will provide video uncompressed data **after** decoding for the broadcast server developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_VIDEO\_DECODER\_BROADCAST\_SERVER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

***Return value of Callback***

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	the framebuffer will be dropped and will not be displayed on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	The video preview data will not be displayed on the window of broadcast server

## Examples

*Example : Register a callback function to get the decoded audio uncompressed data in server-side*

[illegible]

### 11.18.8 QCAP\_REGISTER\_AUDIO\_DECODER\_BROADCAST\_SERVER\_CALLBACK

## Introduction

This callback function will provide audio uncompressed data **after** decoding for broadcast server developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_DECODER\_BROADCAST\_SERVER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

***Return value of Callback***

QCAP_RT_OK	callback already processed
QCAP_RT_FAIL	The framebuffer will be dropped and its sound will be muted
QCAP_RT_SKIP_DISPLAY	The audio uncompressed data will not be played on the window of broadcast server.

## Examples

*Example : Register a callback function to get the decoded audio uncompressed data in server-side*

[illegible]

### 11.18.22 QCAP\_REGISTER\_AUDIO\_DECODER\_MX\_BROADCAST\_SERVER\_CALLBACK

## Introduction

This callback function will provide audio decoder mixer buffer for broadcast client developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_AUDIO_DECODER_MX_BROADCAST_SERVER_CALLBACK</i></b>	pCB	IN	Callback function

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_DECODER\_MX\_BROADCAST\_SERVER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
UINT	iTrackNum	The index of the i-th audio tracks, range 0-3
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to get audio mixer buffer in server side.*

[illegible]

### 11.18.9 QCAP\_REGISTER\_VIDEO\_BROADCAST\_SERVER\_MEDIA\_TIMER\_CALLBACK

## Introduction

This callback function will auto provide a high-resolution media timer function into video broadcast server.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_VIDEO_BROADCAST_SERVER_MEDIA_TIMER_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_VIDEO\_BROADCAST\_SERVER\_MEDIA\_TIMER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
double	dSampleTime	The sampling time in seconds
double	dDelayTime	Specify the video delay time
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to use video media timer in server-side*

[illegible]

### 11.18.10 QCAP\_REGISTER\_AUDIO\_BROADCAST\_SERVER\_MEDIA\_TIMER\_CALLBACK

## Introduction

This callback function will auto provide a high-resolution media timer function into an audio broadcast server.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
UINT	iSessionNum	IN	Specify the index of the i-th sessions, range 0-8
<b><i>PF_AUDIO_BROADCAST_SERVER_MEDIA_TIMER_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_BROADCAST\_SERVER\_MEDIA\_TIMER\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iSessionNum	The index of the i-th sessions, range 0-8
double	dSampleTime	The sampling time in seconds
double	dDelayTime	Specify the audio delay time
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to use audio media timer in server-side*

[illegible]

### 11.18.11 QCAP\_REGISTER\_BROADCAST\_CLIENT\_CONNECTED\_CALLBACK

## Introduction

This callback function will be called when the broadcast client is successfully connected. The user can use this function to get the broadcasting audio/video format, such as video width, video height, video interleaved, frame per second, audio channel number, audio sample width and audio sampling frequency. The user can implement its own function to handle this information.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_BROADCAST_CLIENT_CONNECTED_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_CLIENT\_CONNECTED\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
ULONG	nVideoWidth	The video frame width
ULONG	nVideoHeight	The video frame height
BOOL	bVideosInterleaved	The video interleaved flag
double	dVideoFrameRate	The video frames per second
ULONG	nAudioChannels	The total audio channels (e.g. stereo or mono)
ULONG	nAudioBitsPerSample	The audio bits per sample (e.g. 8bits, 16bits)
ULONG	nAudioSampleFrequency	The audio sampling rate (e.g. 44kHz, 16kHz)
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to get the broadcasting audio/video information*

[illegible]

### 11.18.12 QCAP\_REGISTER\_BROADCAST\_CLIENT\_RECORD\_DONE\_CALLBACK

## Introduction

The user can register a **`PF_BROADCAST_CLIENT_RECORD_DONE_CALLBACK`** function to get notification when the recording process is completed in client-side. When **`QCAP_STOP_BROADCAST_CLIENT_RECORD()`** have done video files processing, then user-defined callback function will be called.

When calling `QCAP_STOP_BROADCAST_CLIENT_RECORD()` asynchronously, or when some video format (e.g. **MP4**) may take a while to complete, this callback can help to know when the video file is ready.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_BROADCAST_CLIENT_RECORD_DONE_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

***PF\_BROADCAST\_CLIENT\_RECORD\_DONE\_CALLBACK***

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
CHAR *	pszFilePathName	pointer to the video record filename
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to notify recording completion in client-side*

[illegible]



### 11.18.13 QCAP\_REGISTER\_BROADCAST\_CLIENT\_RECORD\_FAIL\_CALLBACK

## Introduction

The user can register a ***PF\_BROADCAST\_CLIENT\_RECORD\_FAIL\_CALLBACK*** function to get notification when the recording process is fail for some reason in client-side. When an error occurred during recording process, then user-defined callback function will be called.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_BROADCAST_CLIENT_RECORD_FAIL_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_CLIENT\_RECORD\_FAIL\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
CHAR *	pszFilePathName	pointer to the video record filename
QRESULT	nErrorStatus	The error status of record fail
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

**Example : Register a callback function to notify recording fail in client-side**

[illegible]

### 11.18.14 QCAP\_REGISTER\_BROADCAST\_CLIENT\_SNAPSHOT\_DONE\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_BMP/JPG()** is completed. The user can get the path filename of the snapshot in the callback.

When uses asynchronous snapshot (*b/sAsync* = TRUE), it is useful to know when snapshot file is ready.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_BROADCAST_CLIENT_SNAPSHOT_DONE_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_CLIENT\_SNAPSHOT\_DONE\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
CHAR *	pszFilePathName	pointer to the snapshot filename
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for snapshot done in client-side*

[illegible]

### 11.18.15 QCAP\_REGISTER\_BROADCAST\_CLIENT\_SNAPSHOT\_STREAM\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_BMP/JPG()** image stream is generated in client-side, then a user can get image stream in buffer directly.

For. The users who want a snapshot without saving to a file, call **QCAP\_SNAPSHOT\_BROADCAST\_CLIENT\_BMP/JPG()** by passing *pszFilePathName* file extension only(e.g. "BMP"), then a user can use this callback to retrieve the snapshot image in the buffer.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_BROADCAST_CLIENT_SNAPSHOT_STREAM_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_CLIENT\_SNAPSHOT\_STREAM\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
CHAR *	pszFilePathName	Pointer to the snapshot filename
BYTE *	pStreamBuffer	Pointer to the image framebuffer
ULONG	nStreamBufferLen	Specify the length of image framebuffer
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

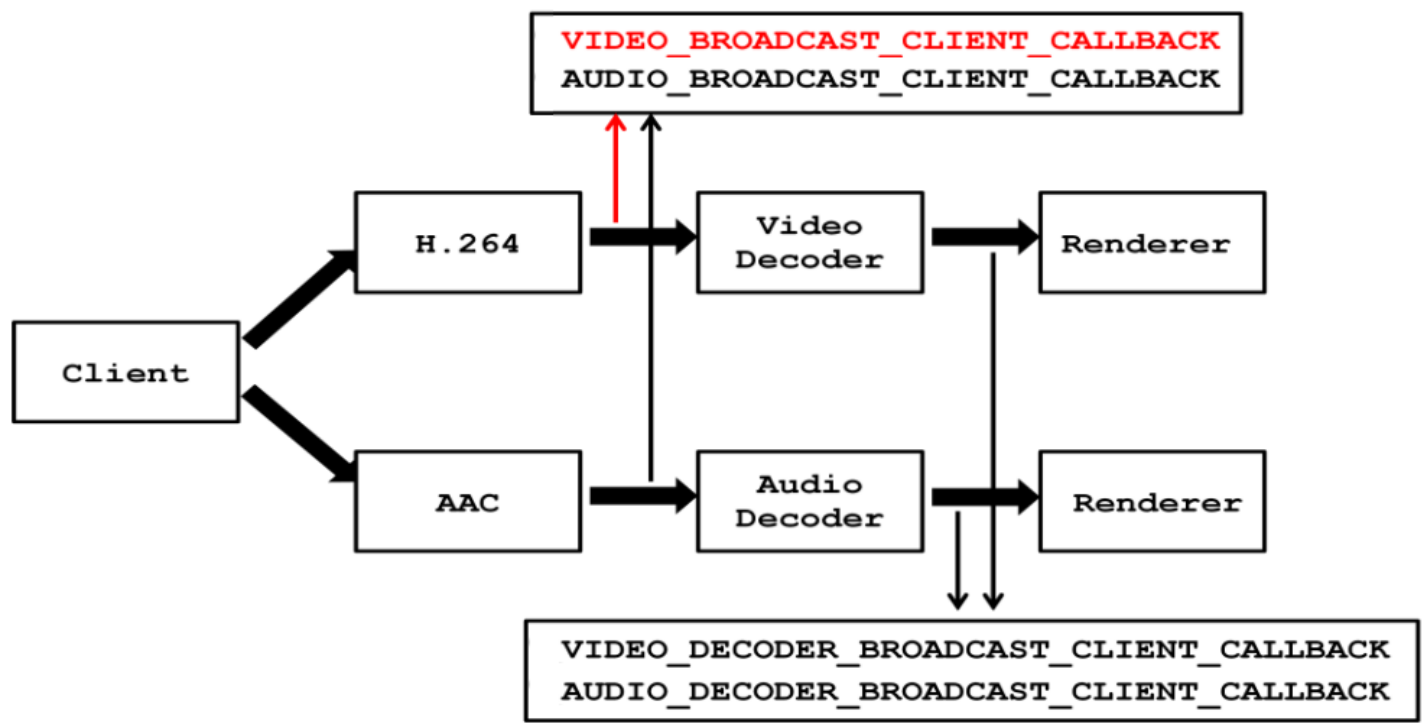
*Example : Register a callback function for snapshot stream completion in client-side*

[illegible]

# 11.18.16 QCAP\_REGISTER\_VIDEO\_BROADCAST\_CLIENT\_CALLBACK

## Introduction

The flowchart of the Audio / Video recording approach of a broadcast client is shown in the figure below. This callback function will provide video compressed **H.264** data *before* video decoding for broadcast client developer to use.



## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b>PF_VIDEO_BROADCAST_CLIENT_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_VIDEO\_BROADCAST\_CLIENT\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
PVOID	pClient	Handle of the broadcast client object
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
BOOL	bIsKeyFrame	Specify the input source's frame is keyframe or not
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and will not be displayed on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	The video preview data ( <b>YUY2</b> or <b>YV12</b> ) will not be displayed on the window of broadcast client
<b>QCAP_RT_SKIP_RECORD_NUM_00</b>	To select which channel record number to skip the framebuffer

## Examples

*Example : Register a callback function to get video callback in client-side*

[illegible]

### 11.18.17 QCAP\_REGISTER\_AUDIO\_BROADCAST\_CLIENT\_CALLBACK

## Introduction

This callback function will provide **PCM / AAC** audio data frames *before* audio decoding for broadcast client developer to use.

The flowchart of the Audio / Video recording approach of the broadcast client please refer to **QCAP\_REGISTER\_VIDEO\_BROADCAST\_CLIENT\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_AUDIO_BROADCAST_CLIENT_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_BROADCAST\_CLIENT\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
double	dSampleTime	The sampling time in seconds
BYTE *	pStreamBuffer	Pointer to the source framebuffer
ULONG	nStreamBufferLen	Specify the length of source framebuffer
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and its sound will be muted
<b>QCAP_RT_SKIP_DISPLAY</b>	The audio uncompressed data will not be played on the window of broadcast client
<b>QCAP_RT_SKIP_RECORD_NUM_00</b>	To select which channel record number to skip the framebuffer

## Examples

*Example : Register a callback function to get audio callback in client-side*

[illegible]

### 11.18.18 QCAP\_REGISTER\_VIDEO\_DECODER\_BROADCAST\_CLIENT\_CALLBACK

## Introduction

This callback function will provide video recording's uncompressed data **after** video decoding for broadcast client developer to use.

The flowchart of the Audio / Video recording approach of the broadcast client please refer to **QCAP\_REGISTER\_VIDEO\_BROADCAST\_CLIENT\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_VIDEO\_DECODER\_BROADCAST\_CLIENT\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and will not be displayed on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	The video preview data will not be displayed on the window of broadcast client

## Examples

*Example : Register a callback function to get video frame after decoding in client-side*

[illegible]

### 11.18.19 QCAP\_REGISTER\_AUDIO\_DECODER\_BROADCAST\_CLIENT\_CALLBACK

## Introduction

This callback function will provide audio uncompressed data **after** audio decoding for broadcast client developer to use.

The flowchart of the Audio / Video recording approach of the broadcast client please refer to **QCAP\_REGISTER\_VIDEO\_BROADCAST\_CLIENT\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_AUDIO\_DECODER\_BROADCAST\_CLIENT\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and its sound will be muted
<b>QCAP_RT_SKIP_DISPLAY</b>	The audio uncompressed data will not be played on the window of broadcast client

## Examples

*Example : Register a callback function to get audio frame after decoding in client-side*

[illegible]



### 11.18.20 QCAP\_REGISTER\_VIDEO\_DECODER\_3D\_BROADCAST\_CLIENT\_CALLBACK

## Introduction

This callback function will provide video recording's 3D uncompressed data **after** video decoding for broadcast client developer to use.

The flowchart of the Audio / Video recording approach of the broadcast client please refer to **QCAP\_REGISTER\_VIDEO\_BROADCAST\_CLIENT\_CALLBACK()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_VIDEO_DECODER_3D_BROADCAST_CLIENT_CALLBACK</i></b>	pCB	IN	Callback function

### Return values

QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

## PF\_VIDEO\_DECODER\_3D\_BROADCAST\_CLIENT\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
UINT	iChNum	The channel number to get <b>SCF</b> file parameters, start from 0
double	dSampleTime	The sampling time in seconds
BYTE *	pFrameBuffer	Specify the input source buffer
ULONG	nFrameBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

<b>QCAP_RT_OK</b>	callback already processed
<b>QCAP_RT_FAIL</b>	The framebuffer will be dropped and will not be displayed on the window
<b>QCAP_RT_SKIP_DISPLAY</b>	The video preview data will not be displayed on the window of 3D player

## Examples

*Example : Register a callback function to get 3D video frame after decoding in client-side*

[illegible]

### 11.18.21 QCAP\_REGISTER\_BROADCAST\_CLIENT\_MEDIAINFO\_CALLBACK

## Introduction

This callback function will provide video / audio media information for broadcast client developer to use.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the broadcast client object
<b><i>PF_BROADCAST_CLIENT_MEDIAINFO_CALLBACK</i></b>	pCB	IN	Callback function

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_BROADCAST\_CLIENT\_MEDIAINFO\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pClient	Handle of the broadcast client object
ULONG	nTotalStreams	Indicate the total number of streams
UINT	iStreamNum	Indicate the stream number
ULONG	nStream_PID	Indicate the Stream PID
ULONG	nProgram_PID	Indicate the Program PID
ULONG	nVideoWidth	The video frame width
ULONG	nVideoHeight	The video frame height
BOOL	bVideosIsInterleaved	The video interleaved flag
double	dVideoFrameRate	The video frames per second
ULONG	nAudioChannels	The total audio channels (e.g. stereo or mono)
ULONG	nAudioBitsPerSample	The audio bits per sample (e.g. 8bits, 16bits)
ULONG	nAudioSampleFrequency	The audio sampling rate (e.g. 44kHz, 16kHz)
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to get media information after decoding in client-side*

[illegible]



# 12 Communication ONVIF Function API

## Introduction



ONVIF is an open industry standard for the IP-based security products and it is committed to the adoption of IP in the security market.

The ONVIF specification defines a common protocol for exchanging information between network video devices including automatic device discovery, video streaming.

In this chapter, a user can easily use ONVIF function to build client/server applications.

## 12.1 ONVIF Server Function API

### 12.1.1 QCAP\_CREATE\_COMMUNICATION\_ONVIF\_SERVER

#### Introduction

This function can create and set ONVIF communication server information such as *Manufacturer name* and *Model name*.

#### Parameters

type	parameter	I/O	descriptions
ULONG	nTotalVideoSources	IN	Specify the total communication video sources
ULONG	nTotalAudioSources	IN	Specify the total communication audio sources
ULONG	nTotalVideoEncoders	IN	Specify the total video encoders
ULONG	nTotalAudioEncoders	IN	Specify the total audio encoders
ULONG	nTotalProfiles	IN	Specify the total communication profile
PVOID *	ppServer	OUT	Handle of the communication server object
CHAR *	pszName	IN	<b>default NULL</b> Specify the name string
CHAR *	pszLocation	IN	<b>default NULL</b> Specify the location string
CHAR *	pszManufacturer	IN	<b>default NULL</b> Specify the manufacturer name
CHAR *	pszModel	IN	<b>default NULL</b> Specify the model name
CHAR *	pszHardwareVersion	IN	<b>default NULL</b> Specify the hardware version
CHAR *	pszFirmwareVersion	IN	<b>default NULL</b> Specify the firmware version
CHAR *	pszDeviceID	IN	<b>default NULL</b> Specify the Device ID

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Create an ONVIF communication server*

```
PVOID pServer = NULL;
```

```
QCAP_CREATE_COMMUNICATION_ONVIF_SERVER( 4, 4,  
                                         4, 4,  
                                         4,  
                                         &pServer );
```

C

## 12.1.2 QCAP\_START\_COMMUNICATION\_SERVER

### Introduction

This function can start an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
ULONG	nNetworkPort	IN	<b>default 8001</b> Specify the port number for the ONVIF protocol on server

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To start an ONVIF communication server.*

```
QCAP_START_COMMUNICATION_SERVER( pServer, 8001 );
```

## 12.1.3 QCAP\_STOP\_COMMUNICATION\_SERVER

### Introduction

This function can stop ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## 12.1.4 QCAP\_DESTROY\_COMMUNICATION\_SERVER

### Introduction

This functions destroy ONVIF communication server object and release its resource.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Stop and release an ONVIF communication server's resources*

```
QCAP_STOP_COMMUNICATION_SERVER( pServer );

QCAP_DESTROY_COMMUNICATION_SERVER( pServer );
```

### 12.1.5 QCAP\_SET\_COMMUNICATION\_SERVER\_VIDEO\_SOURCE

## Introduction

This function can set video parameters of the i-th video stream session in an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iVidSrcNum	IN	Specify the index of the i-th video source
ULONG	nWidth	IN	Specify the width
ULONG	nHeight	IN	Specify the height
ULONG	nFrameRate	IN	Specify the frame rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To set the resolution of an ONVIF communication server*

```
QCAP_SET_COMMUNICATION_SERVER_VIDEO_SOURCE( pServer,  
0,  
1280, 720,  
30 );
```

### 12.1.6 QCAP\_GET\_COMMUNICATION\_SERVER\_VIDEO\_SOURCE

## Introduction

This function is used to get video parameters of the i-th video stream session in an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iVidSrcNum	IN	Specify the index of the i-th video source
ULONG *	pWidth	OUT	Pointer to the video width
ULONG *	pHeight	OUT	Pointer to the video height
ULONG *	pFrameRate	OUT	Pointer to the video frame rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Retrieve the parameters of the  $i$ -th ONVIF video session*

[illegible]

### 12.1.7 QCAP\_SET\_COMMUNICATION\_SERVER\_AUDIO\_SOURCE

## Introduction

This function can set audio format of the i-th audio stream session in an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iAudSrcNum	IN	Specify the index of the i-th audio source
ULONG	nChannels	IN	Specify the total audio channels
ULONG	nBitsPerSample	IN	Specify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set the audio format to Stereo, 16bit, 48KHz of the i-th session*

```
QCAP_SET_COMMUNICATION_SERVER_AUDIO_SOURCE( pServer,  
0,  
2, 16, 48000 );
```

### 12.1.8 QCAP\_GET\_COMMUNICATION\_SERVER\_AUDIO\_SOURCE

## Introduction

This function can get audio parameters of the i-th audio stream session in an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iAudSrcNum	IN	Specify the index of the i-th audio source
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the audio format from the i-th session*

```
QCAP_GET_COMMUNICATION_SERVER_AUDIO_SOURCE( pServer,  
0,  
&nChannels,  
&nBitsPerSample,  
&nSampleFrequency );
```



# 12.1.9 QCAP\_SET\_COMMUNICATION\_SERVER\_VIDEO\_ENCODER

## Introduction

This function can set additional encoder parameters of the i-th video stream session in an ONVIF communication server.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iVidEncNum	IN	Specify the index of the i-th video encoder
ULONG	nEncoderFormat	IN	Specify video encoder format: QCAP_ENCODER_FORMAT_H264 = 0,
ULONG	nWidth	IN	Specify the width
ULONG	nHeight	IN	Specify the height
ULONG	nFrameRate_Min	IN	Specify the minimum frame rate
ULONG	nFrameRate_Max	IN	Specify the maximum frame rate
ULONG	nFrameRate_Default	IN	Specify the default frame rate
ULONG	nRecordMode_Min	IN	Specify the minimum record mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nRecordMode_Max	IN	Specify the maximum frame rate: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nRecordMode_Default	IN	Specify the default frame rate: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality_Min	IN	Specify the minimum quality is 0. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nQuality_Max	IN	Specify the maximum quality is 10000. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nQuality_Default	IN	Specify the default quality value. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate_Min	IN	Specify the minimum bit rate value.
ULONG	nBitRate_Max	IN	Specify the maximum bit rate value.
ULONG	nBitRate_Default	IN	Specify the default bit rate value.
ULONG	nGOP_Min	IN	Specify the minimum GOP is 0.
ULONG	nGOP_Max	IN	Specify the maximum GOP is 255.
ULONG	nGOP_Default	IN	Specify the default GOP value.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set additional video encoder parameters for the session*

```
QCAP_SET_COMMUNICATION_SERVER_VIDEO_ENCODER( pServer,
    0,
    QCAP_ENCODER_FORMAT_H264,
    1920, 1080,
    0, 30, 30,
    0, 2, 1,
    0, 10000, 8000,
    1*1024*1024, 16*1024*1024, 4*1024*1024,
    0, 255, 30 );
```

# 12.1.10 QCAP\_GET\_COMMUNICATION\_SERVER\_VIDEO\_ENCODER

## Introduction

This function can get additional parameters of the i-th video stream session in an ONVIF communication server.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iVidEncNum	IN	Specify the index of the i-th video encoder
ULONG *	pEncoderFormat	OUT	Pointer to the current communication encoder format
ULONG *	pWidth	OUT	Pointer to the video width
ULONG *	pHeight	OUT	Pointer to the video height
ULONG *	pFrameRate_Min	OUT	Pointer to the min communication frame rate
ULONG *	pFrameRate_Max	OUT	Pointer to the max communication frame rate
ULONG *	pFrameRate_Default	OUT	Pointer to the default communication frame rate
ULONG *	pRecordMode_Min	OUT	Pointer to the min communication record mode
ULONG *	pRecordMode_Max	OUT	Pointer to the max communication record mode
ULONG *	pRecordMode_Default	OUT	Pointer to the default communication record mode
ULONG *	pQuality_Min	OUT	Pointer to the min communication quality
ULONG *	pQuality_Max	OUT	Pointer to the max communication quality
ULONG *	pQuality_Default	OUT	Pointer to the default communication quality
ULONG *	pBitRate_Min	OUT	Pointer to the min communication bit rate
ULONG *	pBitRate_Max	OUT	Pointer to the max communication bit rate
ULONG *	pBitRate_Default	OUT	Pointer to the default communication bit rate
ULONG *	pGOP_Min	OUT	Pointer to the min communication GOP
ULONG *	pGOP_Max	OUT	Pointer to the max communication GOP
ULONG *	pGOP_Default	OUT	Pointer to the default communication GOP

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get additional video encoder parameters from the session*

```
QCAP_GET_COMMUNICATION_SERVER_VIDEO_ENCODER( pServer,
    0,
    &nEncoderFormat,
    &nWidth,
    &nHeight,
    &nFrameRate_Min,
    &nFrameRate_Max,
    &nFrameRate_Default,
    &nRecordMode_Min,
    &nRecordMode_Max,
    &nRecordMode_Default,
    &nQuality_Min,
    &nQuality_Max,
    &nQuality_Default,
    &nBitRate_Min,
    &nBitRate_Max,
    &nBitRate_Default,
    &nGOP_Min,
    &nGOP_Max,
    &nGOP_Default );
```

# 12.1.11 QCAP\_SET\_COMMUNICATION\_SERVER\_AUDIO\_ENCODER

## Introduction

This function can set additional parameters of the i-th audio stream session in an ONVIF communication server.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iAudEncNum	IN	Specify the index of the i-th audio encoder
ULONG	nEncoderFormat	IN	Specify audio encoder format: QCAP_ENCODER_FORMAT_PCM QCAP_ENCODER_FORMAT_AAC QCAP_ENCODER_FORMAT_AAC_RAW QCAP_ENCODER_FORMAT_AAC_ADTS QCAP_ENCODER_FORMAT_MP2 QCAP_ENCODER_FORMAT_MP3 QCAP_ENCODER_FORMAT_OPUS
ULONG	nChannels	IN	Set client total audio channels
ULONG	nBitsPerSample	IN	Set client audio bits per sample
ULONG	nSampleFrequency	IN	Set client audio sample frequency
ULONG	nBitRate_Min	IN	Specify the minimum bit rate
ULONG	nBitRate_Max	IN	Specify the maximum bit rate
ULONG	nBitRate_Default	IN	Specify the default bit rate

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# 12.1.12 QCAP\_GET\_COMMUNICATION\_SERVER\_AUDIO\_ENCODER

## Introduction

This function can get audio encoder parameters of the i-th audio stream session in an ONVIF communication server.

## Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iAudEncNum	IN	Specify the index of the i-th audio encoder
ULONG *	pEncoderFormat	OUT	Pointer to the current communication encoder format
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency
ULONG *	pBitRate_Min	OUT	Pointer to the min communication bit rate
ULONG *	pBitRate_Max	OUT	Pointer to the max communication bit rate
ULONG *	pBitRate_Default	OUT	Pointer to the default communication bit rate

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set/get additional audio encoder parameters from the session*

```
QCAP_SET_COMMUNICATION_SERVER_AUDIO_ENCODER( pServer, 0,
                                              QCAP_ENCODER_FORMAT_AAC,
                                              0, 12*1024*1024, 4*1024*1024 );

QCAP_GET_COMMUNICATION_SERVER_AUDIO_ENCODER( pServer, 0,
                                              &nEncoderFormat,
                                              &nBitRate_Min, &nBitRate_Max, &nBitRate_Default );
```

## 12.1.13 QCAP\_SET\_COMMUNICATION\_SERVER\_PROFILE

### Introduction

This function can set profile parameters of the i-th session in an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iProNum	IN	Specify the index of the i-th profile
CHAR *	pszURL	IN	Specify the URL for user to access
UINT	iVidSrcNum	IN	Specify the index of the i-th client video source
UINT	iAudSrcNum	IN	Specify the index of the i-th client audio source
UINT	iVidEncNum	IN	Specify the video encoder number
UINT	iAudEncNum	IN	Specify the audio encoder number

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set audio/video profile for the session*

```
QCAP_SET_COMMUNICATION_SERVER_PROFILE( pServer,
    0,
    "rtsp://root:root@10.10.80.24:800/session0.mpg",
    0, 0, 0, 0 );
```

## 12.1.14 QCAP\_GET\_COMMUNICATION\_SERVER\_PROFILE

### Introduction

This function can get the profile of the i-th profile in an ONVIF communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the communication server object
UINT	iProNum	IN	Specify the index of the i-th portal
CHAR **	ppszURL	OUT	Pointer to the URL string
UINT *	pVidSrcNum	OUT	Pointer to the video source number
UINT *	pAudSrcNum	OUT	Pointer to the audio source number
UINT *	pVidEncNum	OUT	Pointer to the video encoder number
UINT *	pAudEncNum	OUT	Pointer to the audio encoder number

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get profile parameters from the session*

```
QCAP_GET_COMMUNICATION_SERVER_PROFILE( pServer,
    0,
    &pszURL,
    &nVidSrcNum,
    &nAudSrcNum,
    &nVidEncNum,
    &nAudEncNum, );
```

# 12.2 ONVIF Emulator Function API

## 12.2.1 QCAP\_CREATE\_COMMUNICATION\_ONVIF\_EMULATOR

### Introduction

This function can create an ONVIF communication emulator object.

### Parameters

type	parameter	I/O	descriptions
PVOID *	ppEmulator	OUT	Handle of the ONVIF emulator object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Create an ONVIF communication emulator.*

```
QCAP_CREATE_COMMUNICATION_ONVIF_EMULATOR( &pEmulator );
```

## 12.2.2 QCAP\_START\_COMMUNICATION\_EMULATOR

### Introduction

This function can start ONVIF communication emulator.

### Parameters

type	parameter	I/O	descriptions
PVOID	pEmulator	IN	Handle of the ONVIF emulator object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Start an ONVIF communication emulator.*

```
QCAP_START_COMMUNICATION_EMULATOR( pEmulator );
```

## 12.2.3 QCAP\_STOP\_COMMUNICATION\_EMULATOR

### Introduction

This function can stop ONVIF communication emulator.

### Parameters

type	parameter	I/O	descriptions
PVOID	pEmulator	IN	Handle of the ONVIF emulator object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Stop a ONVIF communication emulator.*

```
QCAP_STOP_COMMUNICATION_EMULATOR( pEmulator );
```

## 12.2.4 QCAP\_DESTROY\_COMMUNICATION\_EMULATOR

### Introduction

This function can destroy ONVIF communication emulator object and release its resource.

### Parameters

type	parameter	I/O	descriptions
PVOID	pEmulator	IN	Handle of the ONVIF emulator object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Release an ONVIF emulator object and free resources*

```
QCAP_DESTROY_COMMUNICATION_EMULATOR( pEmulator );
```

### 12.2.5 QCAP\_GET\_COMMUNICATION\_EMULATOR\_SERVER\_INFO

## Introduction

This function can get communication emulator information.

### Parameters

type	parameter	I/O	descriptions
PVOID	pEmulator	IN	Handle of the ONVIF emulator object
UINT	iSvrNum	IN	Specify the index of the i-th server
CHAR * *	ppsNetworkIP	OUT	Pointer to the network IP address
ULONG *	pNetworkPort	OUT	Pointer to the network port number CHAR * *

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the ONVIF emulator network information*

[illegible]

# 12.3 ONVIF Client Function API

## 12.3.1 QCAP\_CREATE\_COMMUNICATION\_ONVIF\_CLIENT

### Introduction

This function can create a communication ONVIF client.

```
QCAP_CREATE_COMMUNICATION_ONVIF_CLIENT(
```

### Parameters

type	parameter	I/O	descriptions
CHAR *	pszNetworkURL	IN	Specify the URL for client to connect
PVOID *	ppClient	OUT	Handle of the client object
CHAR *	pszAccount	IN	<b>default NULL</b> Specify the server log in account, for example: "root"
CHAR *	pszPassword	IN	<b>default NULL</b> Specify the server log in password, for example: "root"

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Create a communication ONVIF client*

```
QCAP_CREATE_COMMUNICATION_ONVIF_CLIENT( "test.com.tw", &pClient, "root", "root" );
```

## 12.3.2 QCAP\_START\_COMMUNICATION\_CLIENT

### Introduction

This function can start communication ONVIF client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : start a communication ONVIF client*

```
QCAP_START_COMMUNICATION_CLIENT( pClient );
```



### 12.3.3 QCAP\_STOP\_COMMUNICATION\_CLIENT

**Introduction**

This function can stop communication ONVIF client.

**Parameters**

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : Stop a communication ONVIF client.*

```
QCAP_STOP_COMMUNICATION_CLIENT( pClient );
```

### 12.3.4 QCAP\_DESTROY\_COMMUNICATION\_CLIENT

**Introduction**

This function can destroy communication ONVIF client object and release its resource.

**Parameters**

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object

**Return value**

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

**Examples**

*Example : Release a communication client and release resources*

```
QCAP_DESTROY_COMMUNICATION_CLIENT( pClient );
```

### 12.3.5 QCAP\_GET\_COMMUNICATION\_CLIENT\_VIDEO\_SOURCE\_INFO

## Introduction

This function can get video source information from a communication client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iVidSrcNum	IN	Specify the index of the i-th client video source
ULONG *	pWidth	OUT	Pointer to the client video width
ULONG *	pHeight	OUT	Pointer to the client video height
ULONG *	pFrameRate	OUT	Pointer to the client video frame rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get video source information of a communication client*

```
QCAP_GET_COMMUNICATION_CLIENT_VIDEO_SOURCE_INFO( pClient,
0,
&nWidth,
&nHeight,
&nFrameRate );
```

### 12.3.6 QCAP\_GET\_COMMUNICATION\_CLIENT\_AUDIO\_SOURCE\_INFO

## Introduction

This function can get audio source information from a communication client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iAudSrcNum	IN	Specify the index of the i-th client audio source
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the audio sample frequency

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get audio source information of a communication client*

```
QCAP_GET_COMMUNICATION_CLIENT_AUDIO_SOURCE_INFO( pClient,
0,
&nChannels,
&nBitsPerSample,
&nSampleFrequency );
```

### 12.3.7 QCAP\_GET\_COMMUNICATION\_CLIENT\_VIDEO\_ENCODER\_INFO

## Introduction

This function can get video encoder parameters of the i-th video stream session in communication client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iVidEncNum	IN	Specify the index of the i-th client video encoder
ULONG *	pTotalVidEncOptions	OUT	,
UINT *	pVidEncOptionNum_Default	OUT	

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get video encoder parameters from a communication client.*

[illegible]

### 12.3.8 QCAP\_GET\_COMMUNICATION\_CLIENT\_AUDIO\_ENCODER\_INFO

## Introduction

This function can get audio encoder parameters of the i-th audio stream session in communication ONVIF client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iAudEncNum	IN	Specify the index of the i-th client audio encoder
ULONG *	pTotalVidEncOptions	OUT	,
UINT *	pVidEncOptionNum_Default	OUT	

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get audio encoder parameters from a communication client.*

[illegible]

### 12.3.9 QCAP\_GET\_COMMUNICATION\_CLIENT\_VIDEO\_ENCODER\_OPTION\_INFO

#### Introduction

This function can get video encoder options from the i-th video stream session in communication client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iVidEncNum	IN	Specify the index of the i-th client video encoder
UINT	iVidEncOptionNum	IN	Specify the the number of video encoder options
ULONG *	pEncoderFormat	OUT	Pointer to the communication client encoder format
ULONG *	pWidthList	OUT	Pointer to a list of the client video width
ULONG *	pWidthListSize	OUT	Pointer to the size of a list of the client video width
ULONG *	pWidth_Default	OUT	Pointer to the default width
ULONG *	pHeightList	OUT	Pointer to a list of the client video height
ULONG *	pHeightListSize	OUT	Pointer to the size of a list of the client video height
ULONG *	pHeight_Default	OUT	Pointer to the default height
ULONG *	pFrameRate_Min	OUT	Pointer to the min communication frame rate
ULONG *	pFrameRate_Max	OUT	Pointer to the max communication frame rate
ULONG *	pFrameRate_Default	OUT	Pointer to the default communication frame rate
ULONG *	pRecordMode_Min	OUT	Pointer to the min communication record mode
ULONG *	pRecordMode_Max	OUT	Pointer to the max communication record mode
ULONG *	pRecordMode_Default	OUT	Pointer to the default communication record mode
ULONG *	pQuality_Min	OUT	Pointer to the min communication quality
ULONG *	pQuality_Max	OUT	Pointer to the max communication quality
ULONG *	pQuality_Default	OUT	Pointer to the default communication quality
ULONG *	pBitRate_Min	OUT	Pointer to the min communication bit rate
ULONG *	pBitRate_Max	OUT	Pointer to the max communication bit rate
ULONG *	pBitRate_Default	OUT	Pointer to the default communication bit rate
ULONG *	pGOP_Min	OUT	Pointer to the min communication GOP
ULONG *	pGOP_Max	OUT	Pointer to the max communication GOP
ULONG *	pGOP_Default	OUT	Pointer to the default communication GOP

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get video encoder options from a communication client.*

```
QCAP_GET_COMMUNICATION_CLIENT_VIDEO_ENCODER_OPTION_INFO(pClient,
    1,
    1,
    &pEncoderFormat,
    &pWidthList,
    &pWidthListSize,
    &pWidth_Default,
    &pHeightList,
    &pHeightListSize,
    &pHeight_Default,
    &pFrameRate_Min,
    &pFrameRate_Max,
    &pFrameRate_Default,
    &pRecordMode_Min,
    &pRecordMode_Max,
    &pRecordMode_Default,
    &pQuality_Min,
    &pQuality_Max,
    &pQuality_Default,
    &pBitRate_Min,
    &pBitRate_Max,
    &pBitRate_Default,
    &pGOP_Min,
    &pGOP_Max,
    &pGOP_Default );
```

## 12.3.10 QCAP\_GET\_COMMUNICATION\_CLIENT\_AUDIO\_ENCODER\_OPTION\_INFO

### Introduction

This function can get audio encoder options from the i-th audio stream session in communication ONVIF client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iAudEncNum	IN	Specify the index of the i-th client audio encoder
UINT	iAudEncOptionNum	IN	Specify the the number of audio encoder options
ULONG *	pEncoderFormat	OUT	Pointer to the communication client encoder format
ULONG *	pChannelsList	OUT	Pointer to a list of the communication audio channels
ULONG *	pChannelsListSize	OUT	Pointer to the size of a list of the communication audio channels
ULONG *	pChannels_Default	OUT	Pointer to the default channels
ULONG *	pBitsPerSampleList	OUT	Pointer to a list of the communication audio samples
ULONG *	pBitsPerSampleListSize	OUT	Pointer to the size of a list of the communication audio samples
ULONG *	pBitsPerSample_Default	OUT	Pointer to the default bits per sample
ULONG *	pSampleFrequencyList	OUT	Pointer to a list of the communication audio sample frequency
ULONG *	pSampleFrequencyListSize	OUT	Pointer to the size of a list of the communication audio frequency
ULONG *	pSampleFrequency_Default	OUT	Pointer to the default frequency
ULONG *	pBitRate_Min	OUT	Pointer to the min communication client bit rate
ULONG *	pBitRate_Max	OUT	Pointer to the max communication client bit rate
ULONG *	pBitRate_Default	OUT	Pointer to the default communication client bit rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Get audio encoder options from a communication client.*

```
QCAP_GET_COMMUNICATION_CLIENT_AUDIO_ENCODER_OPTION_INFO( pClient,
    1,
    1,
    &pEncoderFormat,
    &pChannelsList,
    &pChannelsListSize,
    &pChannels_Default,
    &pBitsPerSampleList,
    &pBitsPerSampleListSize,
    &pBitsPerSample_Default,
    &pSampleFrequencyList,
    &pSampleFrequencyListSize,
    &pSampleFrequency_Default,
    &pBitRate_Min,
    &pBitRate_Max,
    &pBitRate_Default );
```

### 12.3.11 QCAP\_GET\_COMMUNICATION\_CLIENT\_PROFILE\_INFO

## Introduction

This function can get profile parameters from the communication ONVIF client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iProNum	IN	Specify the index of the i-th portal
CHAR * *	ppszURL	OUT	Pointer to the URL
UINT *	pVidSrcNum	OUT	Pointer to the video source number
UINT *	pAudSrcNum	OUT	Pointer to the audio source number
UINT *	pVidEncNum	OUT	Pointer to the video encoder number
UINT *	pAudEncNum	OUT	Pointer to the audio encoder number

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get profile parameters from a communication ONVIF client.*

```
QCAP_GET_COMMUNICATION_CLIENT_PROFILE_INFO( pClient,
0,
&pszURL,
&nVidEncNum,
&nVidSrcNum,
&nAudEncNum,
&nAudSrcNum );
```

### 12.3.12 QCAP\_SET\_COMMUNICATION\_CLIENT\_PROFILE\_VIDEO\_PROPERTY

#### Introduction

This function can set video parameters to the i-th profile video property in communication ONVIF client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iProNum	IN	Specify the index of the i-th portal
ULONG	nEncoderFormat	IN	Specify video encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800
ULONG	nWidth	IN	Specify the width
ULONG	nHeight	IN	Specify the height
ULONG	nFrameRate	IN	Specify the minimum frame rate
ULONG	nRecordMode	IN	Specify the record mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify the quality value. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify the bit rate value.
ULONG	nGOP	IN	Specify the GOP value.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set video parameters to a communication ONVIF client.*

```
QCAP_SET_COMMUNICATION_CLIENT_PROFILE_VIDEO_PROPERTY( pClient,
0,
QCAP_ENCODER_FORMAT_H264,
1920, 1080,
30,
QCAP_RECORD_MODE_CBR,
8000,
4*1024*1024,
30 );
```

### 12.3.13 QCAP\_GET\_COMMUNICATION\_CLIENT\_PROFILE\_VIDEO\_PROPERTY

#### Introduction

This function can get video parameters to the i-th profile video property in communication ONVIF client.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iProNum	IN	Specify the index of the i-th portal
ULONG *	pEncoderFormat	OUT	Pointer to the client encoder format
ULONG *	pWidth	OUT	Pointer to the client video width
ULONG *	pHeight	OUT	Pointer to the client video height
ULONG *	pFrameRate	OUT	Pointer to the client video frame rate
ULONG *	pRecordMode	OUT	Pointer to the client video record mode
ULONG *	pQuality	OUT	Pointer to the client video quality
ULONG *	pBitRate	OUT	Pointer to the client video bit rate
ULONG *	pGOP	OUT	Pointer to the client video GOP

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Get video parameters from a ONVIF communication client.*

```
QCAP_GET_COMMUNICATION_CLIENT_PROFILE_VIDEO_PROPERTY( pClient,
    0,
    &nEncoderFormat,
    &nWidth, &nHeight,
    &nFrameRate,
    &nRecordMode,
    &nQuality,
    &nBitRate,
    &nGOP );
```



## 12.3.14 QCAP\_SET\_COMMUNICATION\_CLIENT\_PROFILE\_AUDIO\_PROPERTY

### Introduction

This function can set audio parameters to the i-th profile audio property in communication ONVIF client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iProNum	IN	Specify the index of the i-th portal
ULONG	nEncoderFormat	IN	Specify audio encoder format: QCAP_ENCODER_FORMAT_PCM QCAP_ENCODER_FORMAT_AAC QCAP_ENCODER_FORMAT_AAC_RAW QCAP_ENCODER_FORMAT_AAC_ADTS QCAP_ENCODER_FORMAT_MP2 QCAP_ENCODER_FORMAT_MP3 QCAP_ENCODER_FORMAT_OPUS
ULONG	nChannels	IN	Specify the total audio channels
ULONG	nBitsPerSample	IN	Specify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency
ULONG	nBitRate	IN	Specify the audio bit rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## 12.3.15 QCAP\_GET\_COMMUNICATION\_CLIENT\_PROFILE\_AUDIO\_PROPERTY

### Introduction

This function can get audio parameters to the i-th profile audio property in communication ONVIF client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
UINT	iProNum	IN	Specify the index of the i-th portal
ULONG *	pEncoderFormat	OUT	Pointer to the current communication encoder format
ULONG *	pChannels	OUT	Pointer to the communication audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the communication audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the communication audio sample frequency
ULONG *	pBitRate	OUT	Pointer to the communication bit rate

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Set/get audio parameters from a communication ONVIF client.*

```
QCAP_SET_COMMUNICATION_CLIENT_PROFILE_AUDIO_PROPERTY( pClient,
    0,
    QCAP_ENCODER_FORMAT_AAC,
    2, 16, 48000,
    4*1024*1024 );

QCAP_GET_COMMUNICATION_CLIENT_PROFILE_AUDIO_PROPERTY( pClient,
    0,
    &nEncoderFormat,
    &nChannels,
    &nBitsPerSample, &nSampleFrequency,
    &nBitRate );
```

### 12.3.16 QCAP\_SET\_COMMUNICATION\_CLIENT\_CUSTOM\_PROPERTY

## Introduction

This function is used to set/change custom properties to a communication client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
CHAR *	pszProperty	IN	Specify client custom property interface
CHAR *	pszValue	IN	Specifies the property value

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set/change custom properties to a communication client.*

[illegible]

### 12.3.17 QCAP\_GET\_COMMUNICATION\_CLIENT\_CUSTOM\_PROPERTY

## Introduction

This function is used to get custom properties to a communication client.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the ONVIF client object
CHAR *	pszProperty	IN	Specify get client custom property interface
CHAR * *	ppszValue	OUT	Pointer the specify property value. It cannot be NULL. The range of value is dependent on its property

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

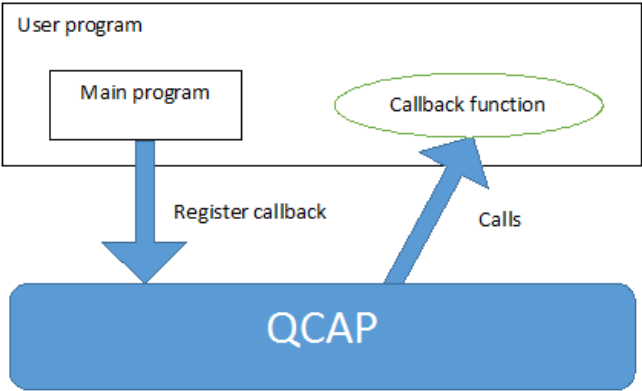
## Examples

*Example : Get custom properties to a communication client.*

[illegible]

# 12.4 ONVIF Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_COMMUNICATION_SERVER_GET_CUSTOM_PROPERTY_CALLBACK	PF_COMMUNICATION_SERVER_GET_CUSTOM_PROPERTY_CALLBACK
QCAP_REGISTER_COMMUNICATION_SERVER_SET_CUSTOM_PROPERTY_CALLBACK	PF_COMMUNICATION_SERVER_SET_CUSTOM_PROPERTY_CALLBACK
QCAP_REGISTER_COMMUNICATION_SERVER_PROFILE_VIDEO_SETUP_CALLBACK	PF_COMMUNICATION_SERVER_PROFILE_VIDEO_SETUP_CALLBACK
QCAP_REGISTER_COMMUNICATION_SERVER_PROFILE_AUDIO_SETUP_CALLBACK	PF_COMMUNICATION_SERVER_PROFILE_AUDIO_SETUP_CALLBACK
QCAP_REGISTER_COMMUNICATION_EMULATOR_SCAN_DONE_CALLBACK	PF_COMMUNICATION_EMULATOR_SCAN_DONE_CALLBACK
QCAP_REGISTER_COMMUNICATION_CLIENT_CONNECTED_CALLBACK	PF_COMMUNICATION_CLIENT_CONNECTED_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

### 12.4.1 QCAP\_REGISTER\_COMMUNICATION\_SERVER\_GET\_CUSTOM\_PROPERTY\_CALLBACK

## Introduction

This callback function is called when getting custom property from communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
<b><i>PF_COMMUNICATION_SERVER_GET_CUSTOM_PROPERTY_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_COMMUNICATION\_SERVER\_GET\_CUSTOM\_PROPERTY\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
PVOID	pServer	Handle of the broadcast server object
CHAR *	pszProperty	The client custom property interface name
CHAR *	pszValue	<b>This is a OUTPUT parameter</b> The value of customer property
PVOID	pUserData	Pointer to custom user data

***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to notify getting custom property*

[illegible]

## 12.4.2 QCAP\_REGISTER\_COMMUNICATION\_SERVER\_SET\_CUSTOM\_PROPERTY\_CALLBACK

## Introduction

This callback function is called when setting custom property command from communication server.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
<b><i>PF_COMMUNICATION_SERVER_SET_CUSTOM_PROPERTY_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### PF\_COMMUNICATION\_SERVER\_SET\_CUSTOM\_PROPERTY\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
CHAR *	pszProperty	The client custom property interface name
CHAR *	pszValue	To specify the property value
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to notify getting custom property*

[illegible]

### 12.4.3 QCAP\_REGISTER\_COMMUNICATION\_SERVER\_PROFILE\_VIDEO\_SETUP\_CALLBACK

## Introduction

When the profile of video setup property is changed in a communication server, the callback function will be called.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
<b><i>PF_COMMUNICATION_SERVER_PROFILE_VIDEO_SETUP_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_COMMUNICATION\_SERVER\_PROFILE\_VIDEO\_SETUP\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iProNum	The index of the i-th profile
ULONG	nEncoderFormat	The video encoder format
ULONG	nWidth	video frame width
ULONG	nHeight	
video frame height	ULONG	nFrameRate
video frames per second	ULONG	nRecordMode
The record mode	ULONG	nQuality
The quality value. It is used for <b>VBR</b> and <b>ABR</b> .	ULONG	nBitRate
The bit rate value	ULONG	nGOP
The GOP value	PVOID	pUserData

***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to notify profile of video setup property is changed*

[illegible]

#### 12.4.4 QCAP\_REGISTER\_COMMUNICATION\_SERVER\_PROFILE\_AUDIO\_SETUP\_CALLBACK

## Introduction

When the profile of audio setup property is changed in a communication server, the callback function will be called.

### Parameters

type	parameter	I/O	descriptions
PVOID	pServer	IN	Handle of the broadcast server object
<b><i>PF_COMMUNICATION_SERVER_PROFILE_AUDIO_SETUP_CALLBACK</i></b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_COMMUNICATION\_SERVER\_PROFILE\_AUDIO\_SETUP\_CALLBACK

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pServer	Handle of the broadcast server object
UINT	iProNum	The index of the i-th profile
ULONG	nEncoderFormat	The audio encoder format
ULONG	nChannels	The total audio channels (e.g. stereo or mono)
ULONG	nBitsPerSample	The audio bits per sample (e.g.8bits, 16bits)
ULONG	nSampleFrequency	The audio sampling rate (e.g.44kHz, 16kHz)
ULONG	nBitRate	The bit rate value
PVOID	pUserData	Pointer to custom user data

### ***Return value of Callback***

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to notify profile of audio setup property is changed*

[illegible]

#### 12.4.5 QCAP\_REGISTER\_COMMUNICATION\_EMULATOR\_SCAN\_DONE\_CALLBACK

## Introduction

This callback function is called when the communication emulator scanning is complete.

### Parameters

type	parameter	I/O	descriptions
PVOID	pEmulator	IN	Handle of the emulator object
<b>PF_COMMUNICATION_EMULATOR_SCAN_DONE_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

***PF\_COMMUNICATION\_EMULATOR\_SCAN\_DONE\_CALLBACK***

### Parameters of Callback

<i>type</i>	<i>parameter</i>	<i>callback descriptions</i>
PVOID	pEmulator	The ONVIF emulator object
ULONG	nTotalServers	The number of total servers
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to know when the communication emulator scanning is complete.*

[illegible]



# 12.4.6 QCAP\_REGISTER\_COMMUNICATION\_CLIENT\_CONNECTED\_CALLBACK

## Introduction

When the broadcasting client connection is built, the callback function will be called. The user can use this function to get audio/video information from a communication client. The user can implement its own function to handle this information.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClient	IN	Handle of the communication client object
<b>PF_COMMUNICATION_CLIENT_CONNECTED_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_COMMUNICATION\_CLIENT\_CONNECTED\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pClient	Handle of the client object
ULONG	nTotalVideoSources	The total communication video sources
ULONG	nTotalAudioSources	The total communication audio sources
ULONG	nTotalVideoEncoders	The total video encoders
ULONG	nTotalAudioEncoders	The total audio encoders
ULONG	nTotalProfiles	Total communication profile
CHAR *	pszName	The name string
CHAR *	pszLocationr	The Location string
CHAR *	pszManufacturer	The manufacturer name
CHAR *	pszModel	The model name
CHAR *	pszHardwareVersion	The hardware version
CHAR *	pszFirmwareVersion	The firmware version
CHAR *	pszDeviceID	The hardware device ID
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to know when client is connected*

```
QRETURN onvif_client_connected( PVOID pClient,
                                ULONG nTotalVideoSources,
                                ULONG nTotalAudioSources,
                                ULONG nTotalVideoEncoders,
                                ULONG nTotalAudioEncoders,
                                ULONG nTotalProfiles,
                                CHAR * pszName,
                                CHAR * pszLocationr,
                                CHAR * pszManufacturer,
                                CHAR * pszModel,
                                CHAR * pszHardwareVersion,
                                CHAR * pszFirmwareVersion,
                                CHAR * pszDeviceID,
                                PVOID pUserData )
{
    return QCAP_RT_OK;
}

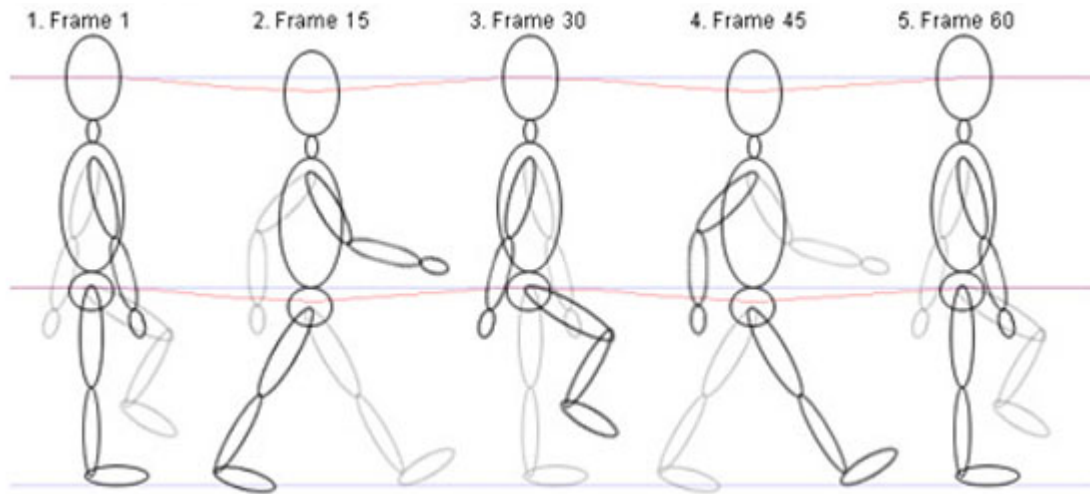
void test_callback()
{
    PF_COMMUNICATION_CLIENT_CONNECTED_CALLBACK pCB = onvif_client_connected;

    QCAP_REGISTER_COMMUNICATION_CLIENT_CONNECTED_CALLBACK( pClient, pCB, pUserData );
}
```

# 13 Animation Function API

---

## Introduction



This chapter provides functions that run animation sprites on a live video stream. For example, by using animation clip and sprites functions, a user can control alpha blending, put a picture on any position, scaling the output, or loads a color buffer to your live video source.



```

QCAP_STEP_ANIMATION_CLIP( pClip,
                           &iFrameNum,
                           &pFrameBuffer,
                           &nFrameBufferLen,
                           TRUE );

QCAP_STOP_ANIMATION_CLIP( pClip );

QCAP_DESTROY_ANIMATION_CLIP( pClip );

```

## 13.2 QCAP\_START\_ANIMATION\_CLIP

### Introduction

The user can use this function to start an animation clip.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Start a animation clip*

```

QCAP_START_ANIMATION_CLIP( pClip );

```

## 13.3 QCAP\_STOP\_ANIMATION\_CLIP

### Introduction

The user can use this function to stop animation clip.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Stop a animation clip*

```
QCAP_STOP_ANIMATION_CLIP( pClip );
```

## 13.4 QCAP\_DESTROY\_ANIMATION\_CLIP

### Introduction

This function can destroy animation clip and release its resource.



If you have use **QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_SOURCE()**, you must destroy the animation clip before device destroys or to set the source to NULL.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Destroy a animation clip*

```
QCAP_DESTROY_ANIMATION_CLIP( pClip );
```

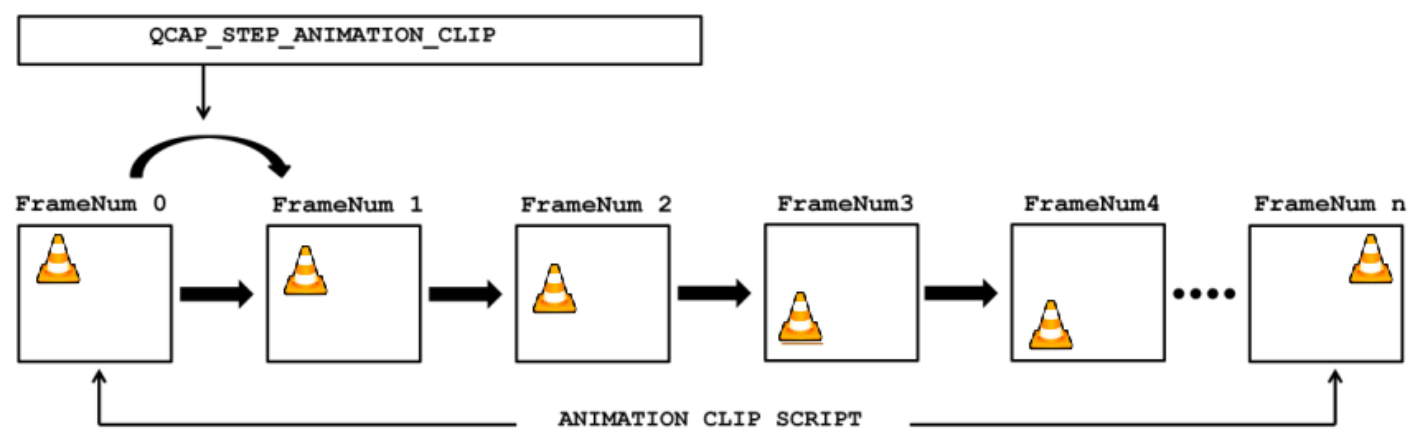
# 13.5 QCAP\_STEP\_ANIMATION\_CLIP

## Introduction

The user can use this function to execute an animation clip step by step.

When the function is called each time, the animation clip output current frame and go to next frame.

The result as shown in this figure:



## Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT *	pFrameNum	OUT	Pointer to the frame number
BYTE **	ppFrameBuffer	OUT	Pointer to the framebuffer
ULONG *	pFrameBufferLen	OUT	Pointer to the framebuffer length
BOOL	bClearBackground	IN	<b>default TRUE</b> Enable/Disable the Clear Background

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Play an animation clip step by step*

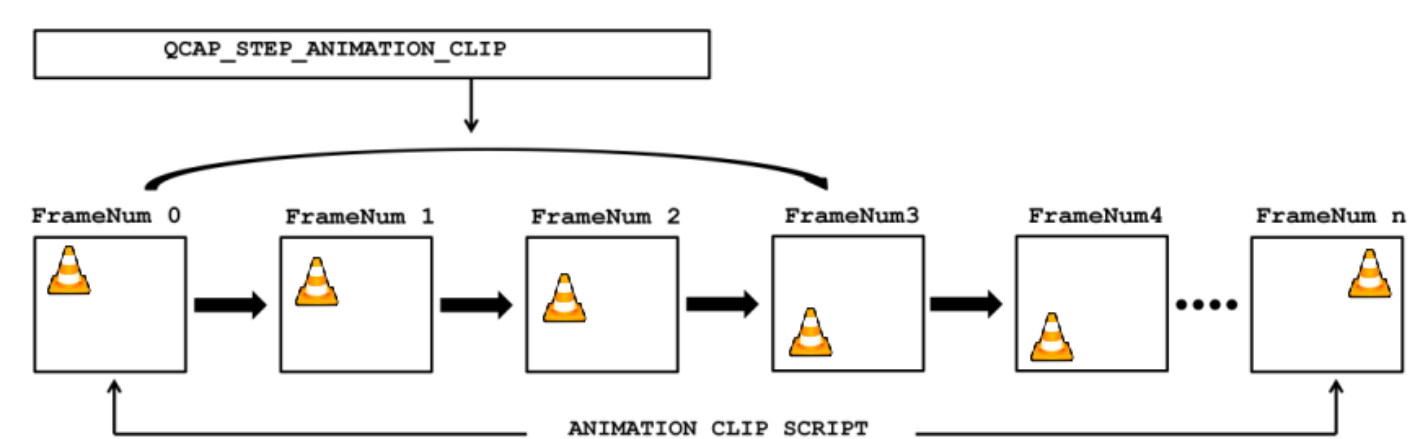
```
QCAP_STEP_ANIMATION_CLIP( pClip, &pFrameNum, &pFrameBuffer, &nFrameBufferLen, TRUE );
```

# 13.6 QCAP\_SEEK\_ANIMATION\_CLIP

## Introduction

The user can use this function to execute animation clip and seek to next frame. When the function is called each time, the animation clip output current frame and then jump to any frame.

The result as shown in figure:



## Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iFrameNum	IN	Specify the next index of frame number
BYTE **	ppFrameBuffer	OUT	Pointer to the input source buffer
ULONG *	pFrameBufferLen	OUT	Pointer to the input source buffer's size
BOOL	bClearBackground	IN	<b>default TRUE</b> Enable/Disable the background clear

## Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

## Examples

*Example : Play an animation clip and seek to next position*

```
QCAP_SEEK_ANIMATION_CLIP( pClip, iFrameNum, &pFrameBuffer, &nFrameBufferLen, TRUE );
```

# 13.7 Animation Sprite Functions

## Introduction

This section provides function that can control the sprite in an animation clip.

### 13.7.1 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_TRANSFORM\_PROPERTY

#### Introduction

The user can use this function to set transformation setting of animation clip sprite.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
UINT	iFrameNum	IN	Specify the index of frame number
INT	nPositionX	IN	Specify the horizontal positionX
INT	nPositionY	IN	Specify the vertical position Y
INT	nScaleW	IN	Specify the target scaling width
INT	nScaleH	IN	Specify the target scaling height

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set transformation setting of animation clip sprite.*

```
QCAP_SET_ANIMATION_CLIP_SPRITE_TRANSFORM_PROPERTY( pClip,
                                                    0,
                                                    100,
                                                    150, 150,    //Position X,Y
                                                    100, 100 ); //Scale W,H
```



### 13.7.2 QCAP\_GET\_ANIMATION\_CLIP\_SPRITE\_TRANSFORM\_PROPERTY

## Introduction

The user can use this function to get transformation/scaling setting of animation clip sprite.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
UINT	iFrameNum	IN	Specify the index of frame number
INT *	pPositionX	OUT	Pointer to the horizontal position X
INT *	pPositionY	OUT	Pointer to the vertical position Y
INT *	pScaleW	OUT	Pointer to the target scaling width
INT *	pScaleH	OUT	Pointer to the target scaling height

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get transformation setting of animation clip sprite.*

[illegible]

# 13.7.3 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_ALPHA\_PROPERTY

## Introduction

The user can use this function to set the alpha value of animation clip sprite.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
UINT	iFrameNum	IN	Specify the index of frame number
double	nAlpha	IN	Specify the alpha value

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# 13.7.4 QCAP\_GET\_ANIMATION\_CLIP\_SPRITE\_ALPHA\_PROPERTY

## Introduction

The user can use this function to get the alpha value of animation clip sprite.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
UINT	iFrameNum	IN	Specify the index of frame number
double *	pAlpha	OUT	Pointer to the current alpha value

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set/Get the alpha value of animation clip sprite.*

```
QCAP_SET_ANIMATION_CLIP_SPRITE_ALPHA_PROPERTY( pClip, 0, 100, 50 );

QCAP_GET_ANIMATION_CLIP_SPRITE_ALPHA_PROPERTY( pClip, 0, 100, &nAlpha );
```

# 13.7.5 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_PICTURE

## Introduction

The user can use this function to set a picture on an animation clip sprite.

## Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
CHAR *	pszFilePathName	IN	Specify the image file name Supported format: <b>BMP, JPG, PNG,GIF</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set a picture on a sprite in animation clip.*

```
QCAP_SET_ANIMATION_CLIP_SPRITE_PICTURE( pClip, 0, "C:/PICTURE1.BMP" );
```

## 13.7.6 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_BUFFER

## 13.7.7 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_BUFFER\_EX

### Introduction

The user can use this function to set each channels buffer of animation clip.

This function update sprites buffer (such as video file update) to buffer of animation clip.

The enhanced version of this function can set the crop parameters of each channels buffer of animation clip.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify the width of frame
ULONG	nHeight	IN	Specify the height of frame
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
ULONG	nCropX	IN	Specify the x-coordinate of the crop region display <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop region display <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX()</b>
BOOL	bCloneCopy	IN	<b>default FALSE</b> Enable/Disable the Clone copy

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To set each channels buffer of animation clip.*

```
QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER( pClip,  
    0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720,  
    FrameBuffer,  
    FrameBufferLen );  
  
QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX( pClip,  
    0,  
    QCAP_COLORSPACE_TYEP_YUY2,  
    1280, 720,  
    FrameBuffer,  
    FrameBufferLen,  
    0, 0,  
    720, 480 );
```

## 13.7.8 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_SOURCE

## 13.7.9 QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_SOURCE\_EX

### Introduction

The user can use this function to set the source of each channel for animation clip.

It can help a user to use complex **QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_BUFFER()** function to attach one live device channel on it. When device's preview data coming, the **QCAP\_SET\_ANIMATION\_CLIP\_SPRITE\_BUFFER()** will be called automatically. The function has a enhanced version to set the crop parameters to each channel of animation clip.

### Parameters

type	parameter	I/O	descriptions
PVOID	pClip	IN	Handle of the clip object
UINT	iSpriteNum	IN	Specify the index of sprite, start from 0
PVOID	pDevice	IN	Handle of the capture card object
ULONG	nCropX	IN	Specify the x-coordinate of the crop region display <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop region display <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in</b> <b>QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX()</b>
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
BOOL	bCloneCopy	IN	<b>default FALSE</b> Enable/Disable the clone copy

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

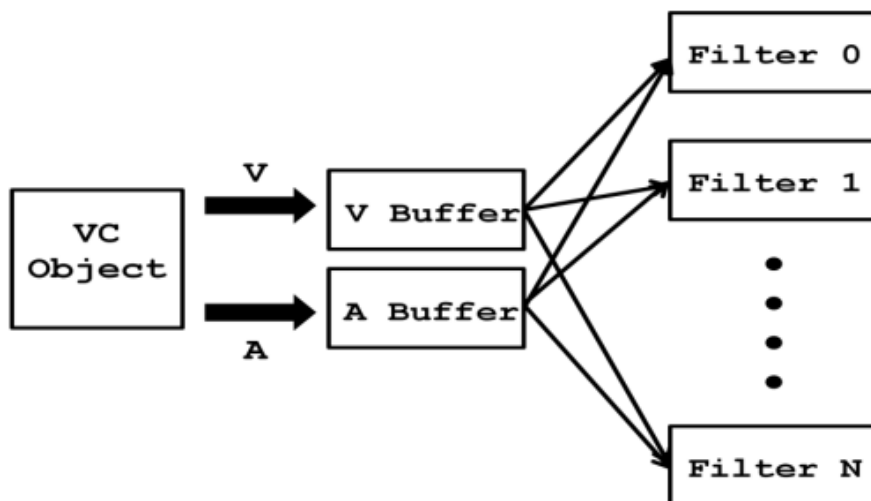
*Example : Set the animation source*

```
QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE( pClip,  
                                         0,  
                                         Device,  
                                         QCAP_COLORSPACE_TYEP_YUY2,  
                                         FALSE );  
  
QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX( pClip,  
                                             0,  
                                             10,  
                                             40,  
                                             1900,  
                                             1000,  
                                             QCAP_COLORSPACE_TYEP_YUY2,  
                                             FALSE );
```

# 14 Virtual Camera Function API

---

## Introduction



This chapter provides a virtual camera (VC) function. The user can use any camera device to simulate a real audio/video stream come from the capture card. The virtual camera allows you to create one share mapping of video framebuffer to another 3rd party software, such as **QQ** or **Skype**.

The virtual camera object acts like one streaming sender and the **Custom Virtual Camera Filter** act likes a streaming receiver. The virtual camera object has audio/video buffer, data stream connected to many filters internally. The user can mix the data with buffers, and push the uncompressed buffer at desired time-stamp, also snapshot function and OSD effect are supported.

For a user who wants to use a virtual camera, must register the **VIRCAM.X86.AX** filter into your windows system. We call it as **Custom Virtual Camera Filter**. You can find it from our **QCAPI\TOOLS** folder.



# 14.1 Virtual Camera Device Functions

## Introduction

In this section describes the major function to initialize / release a virtual camera object:

functions	description
QCAP_CREATE_VIRTUAL_CAMERA()	This function can create one virtual camera object.
QCAP_START_VIRTUAL_CAMERA()	This function can start virtual camera.
QCAP_STOP_VIRTUAL_CAMERA()	This function can stop virtual camera.
QCAP_DESTROY_VIRTUAL_CAMERA()	This function can destroy camera object and release its resource.



The user must set `QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY()` / `QCAP_SET_AUDIO_VIRTUAL_CAMERA_PROPERTY()` before calls to `QCAP_START_VIRTUAL_CAMERA()` function.

## 14.1.1 QCAP\_CREATE\_VIRTUAL\_CAMERA

### Introduction

The create function can create one virtual camera (VC) object.

### Parameters

type	parameter	I/O	descriptions
UINT	iCamNum	IN	Specify the camera number, start from 0 max is 0-63
PVOID *	ppCamera	OUT	of the camera object

### Return value

Returns `QCAP_RS_SUCCESSFUL` if OK, otherwise an error occurred.

### Examples

*Example : The programming flow of using a virtual camera*

```
QCAP_CREATE_VIRTUAL_CAMERA( 0, &pCamera );

QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY( pCamera,
                                         QCAP_COLORSPACE_TYEP_YV12,
                                         nWidth, nHeight,
                                         dFrameRate );

QCAP_SET_AUDIO_VIRTUAL_CAMERA_PROPERTY( pCamera,
                                         2,
                                         16,
```

```

44000);

QCAP_START_VIRTUAL_CAMERA( pCamera );

QCAP_SET_VIDEO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER_EX( pCamera,
    QCAP_COLORSPACE_TYEP_YUY2,
    1280, 720,
    &pFrameBuffer,
    1280 * 720 * 2,
    0, 0,
    720, 480,
    0 );

QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER_EX( pCamera,
    2, 16, 48000,
    pFrameBuffer,
    nFrameBufferLen,
    0 );

QCAP_STOP_VIRTUAL_CAMERA( pCamera );

QCAP_DESTROY_VIRTUAL_CAMERA( pCamera );

```

## 14.1.2 QCAP\_START\_VIRTUAL\_CAMERA

### Introduction

This function can start a virtual camera (VC) object to running state.

### Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To set a virtual camera object to running state*

```

QCAP_START_VIRTUAL_CAMERA( pCamera );

```

# 14.1.3 QCAP\_STOP\_VIRTUAL\_CAMERA

## Introduction

This function can stop a virtual camera (VC) object to output data streams.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : After stop a virtual camera object, no data stream will output*

```
QCAP_STOP_VIRTUAL_CAMERA( pCamera );
```

# 14.1.4 QCAP\_DESTROY\_VIRTUAL\_CAMERA

## Introduction

This function can destroy a virtual camera (VC) object and free its system resources.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To destroy a virtual camera object*

```
QCAP_DESTROY_VIRTUAL_CAMERA( pCamera );
```

# 14.2 Virtual Camera Property Functions

## Introduction

In this section provides functions that can set the video resolutions, color format, and audio format, sample rates for the virtual camera.

### 14.2.1 QCAP\_SET\_VIDEO\_VIRTUAL\_CAMERA\_PROPERTY

#### Introduction

This function can set video format of the virtual camera. It will change the output format of Virtual Camera Filter's video output pin.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nWidth	IN	Specify video encoder width.
ULONG	nHeight	IN	Specify video encoder height.
double	dFrameRate	IN	Specify video encoder frame rate.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Set video format to **YV12** of a virtual camera.*

```
QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY( pCamera,
                                         QCAP_COLORSPACE_TYEP_YV12,
                                         nWidth, nHeight,
                                         dFrameRate );
```

# 14.2.2 QCAP\_GET\_VIDEO\_VIRTUAL\_CAMERA\_PROPERTY

## Introduction

This function can get the video format of the virtual camera.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
ULONG *	pColorSpaceType	OUT	Pointer to the current color space type
ULONG *	pWidth	OUT	Pointer to the current video width
ULONG *	pHeight	OUT	Pointer to the current video height
double *	pFrameRate	OUT	Pointer to the current frame rate

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Query the video format of a virtual camera.*

```
QCAP_GET_VIDEO_VIRTUAL_CAMERA_PROPERTY( pCamera,
                                         &nColorSpaceType,
                                         &nWidth, &nHeight, &nFrameRate );
```

### 14.2.3 QCAP\_SET\_AUDIO\_VIRTUAL\_CAMERA\_PROPERTY

## Introduction

This function can set audio format of the virtual camera. It will change the output format of Virtual Camera Filter's audio output pin.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	of the camera object
ULONG	nChannels	IN	Specify the total audio channels
ULONG	nBitsPerSample	IN	SSpecify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency

### Return value

Returns **QCAP RS SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Set audio format to (Stereo, 16bits, 44KHz) of a virtual camera*

```
QCAP_SET_AUDIO_VIRTUAL_CAMERA_PROPERTY( pCamera,  
                                         2, 16, 44000);
```

#### 14.2.4 QCAP\_GET\_AUDIO\_VIRTUAL\_CAMERA\_PROPERTY

## Introduction

This function can get the audio format of the virtual camera.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
ULONG *	pChannels	OUT	Pointer to the total audio channels
ULONG *	pBitsPerSample	OUT	Pointer to the current audio bits per sample
ULONG *	pSampleFrequency	OUT	Pointer to the current audio sample frequency

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get audio format of a virtual camera.*

[illegible]

# 14.3 Virtual Camera Data Functions

## Introduction

This section provides functions to push audio/video uncompressed buffers into virtual camera stream.



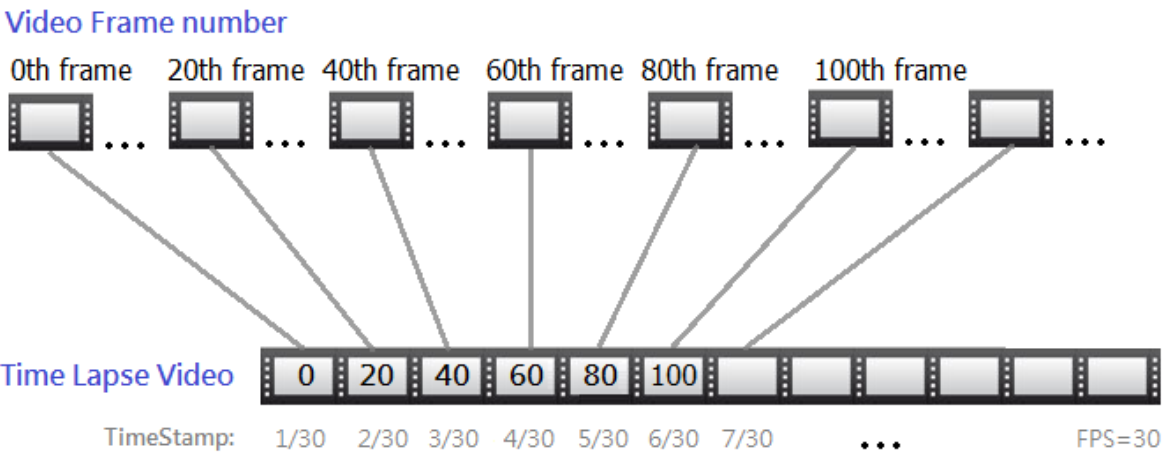
The performance of **RGB** type is faster than **BGR** in our alpha blending algorithm.

### 14.3.1 QCAP\_SET\_VIDEO\_VIRTUAL\_CAMERA\_UNCOMPRESSION\_BUFFER

### 14.3.2 QCAP\_SET\_VIDEO\_VIRTUAL\_CAMERA\_UNCOMPRESSION\_BUFFER\_EX

## Introduction

This function can push uncompressed video buffer into virtual camera's share memory. The users can use *Sample Time* to control time-stamp of video, and will result a Time-Lapse video, as shown below:



## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420





### 14.3.3 QCAP\_SET\_AUDIO\_VIRTUAL\_CAMERA\_UNCOMPRESSION\_BUFFER

#### 14.3.4 QCAP\_SET\_AUDIO\_VIRTUAL\_CAMERA\_UNCOMPRESSION\_BUFFER\_EX

## Introduction

This function can push uncompressed audio buffer into virtual camera's share memory.



The SampleTime of audio must same with SampleTime of video.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
ULONG	nChannels	IN	Specify the total audio channels <b>Only in</b> <b>QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nBitsPerSample	IN	Specify the audio bits per sample <b>Only in</b> <b>QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nSampleFrequency	IN	Specify the audio sample frequency <b>Only in</b> <b>QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER_EX()</b>
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame. Set 0 to auto generate by system clock.

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To push uncompressed audio buffer into virtual camera's share memory.*

```
QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( pCamera,
                                                    &nFrameBuffer,
                                                    1280 * 720 * 2,
                                                    0 );

QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER_EX( pCamera,
                                                       2, 16, 48000,
                                                       pFrameBuffer,
                                                       nFrameBufferLen,
                                                       0 );
```

# 14.4 Virtual Camera Snapshot Functions

## Introduction

This chapter will help to take a snapshot of your virtual camera video stream. Follow this guide to take a snapshot of your whole video display, or any section of the video you want. The user can save a snapshot to a **BMP/JPG**, or to trigger a snapshot to get the image stream buffer from a callback function without saving it to disk.

### 14.4.1 QCAP\_SNAPSHOT\_VIRTUAL\_CAMERA\_BMP

### 14.4.2 QCAP\_SNAPSHOT\_VIRTUAL\_CAMERA\_BMP\_EX

## Introduction

This function takes a snapshot of video from a virtual camera and saves to **BMP** 24bit or 32bit file.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_BMP\_EX()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.BMP24" → save to 24bit BMP "Filename.BMP32 or BMP" → save to 32bit BMP "BMP24" → To snapshot stream in callback only (no save to file.) "BMP32"/"BMP" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX()</b>
BOOL	blsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Take a snapshot and save to file via virtual camera*

```
QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP( pCamera,  
                                   "C:/PICTURE1.BMP24" );  
  
QCAP_SNAPSHOT_VIRTUAL_CAMERA_BMP_EX( pCamera,  
                                       "C:/PICTURE1.BMP",  
                                       10, 40,  
                                       1900, 1000,  
                                       720, 480 );
```

### 14.4.3 QCAP\_SNAPSHOT\_VIRTUAL\_CAMERA\_JPG

### 14.4.4 QCAP\_SNAPSHOT\_VIRTUAL\_CAMERA\_JPG\_EX

#### Introduction

This function takes a snapshot of video from a virtual camera and saves to **JPEG** image file format.



For more detailed parameters descriptions, please refer to **QCAP\_SNAPSHOT\_JPG\_EX()**.

#### Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
CHAR *	pszFilePathName	IN	Specify the file type to store: "Filename.JPG" → To snapshot to <b>JPEG</b> image file "JPG" → To snapshot stream in callback only (no save to file.)
ULONG	nCropX	IN	Specify the x-coordinate of the crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX()</b>
ULONG	nCropY	IN	Specify the y-coordinate of the crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX()</b>
ULONG	nCropW	IN	Specify the width of crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX()</b>
ULONG	nCropH	IN	Specify the height of crop <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX()</b>
ULONG	nDstW	IN	Specify the width of downscale <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX()</b>
ULONG	nDstH	IN	Specify the height of downscale <b>Only in QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX()</b>
ULONG	nQuality	IN	Specify the quality of <b>JPEG</b> file, from 0-100
BOOL	bIsAsync	IN	<b>default TRUE</b> Set the asynchronous operation flag
ULONG	nMilliseconds	IN	<b>default 0</b> Time-out interval in ms. (synchronous mode only): 1. set 0 to returns immediately. 2. set INFINITE the time-out interval never elapses.

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Take a snapshot and save to **JPEG** via virtual camera*

```
QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG( pCamera,
                                   "C:/PICTURE1.JPG",
                                   85 );

QCAP_SNAPSHOT_VIRTUAL_CAMERA_JPG_EX( pCamera,
                                       "C:/PICTURE1.JPG",
                                       10, 40,
                                       1900, 1000,
                                       720, 480,
                                       85 );
```

# 14.5 Virtual Camera OSD Functions

## Introduction

An on-screen display (OSD) are control functions on a video screen that allows you to draw text fields, overlapped pictures, or put customer image buffer with blending or color key effect.

For more detailed parameters descriptions, please refer to **Chapter 7 OSD Function API**.

### 14.5.1 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_TEXT

### 14.5.2 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_TEXT\_EX

### 14.5.3 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_TEXT\_W

### 14.5.4 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_TEXT\_EX\_W

## Introduction

The user can use this function to create a text field objects used for on-screen display in virtual camera video stream.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_TEXT()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output Set -1 to auto calculate width.
INT	h	IN	Specify the height of OSD output Set -1 to auto calculate height.
CHAR * WSTRING	pszString pwszString	IN	Specify to display the text of OSD output Support wide character string
CHAR * WSTRING	pszFontFamilyName pwszFontFamilyName	IN	Specify the font name used to display the text of OSD output Support wide character string
ULONG	nFontStyle	IN	Specify the font style used to display the text of OSD output Available values are QCAP_FONT_STYLE_REGULAR QCAP_FONT_STYLE_BOLD QCAP_FONT_STYLE_ITALIC QCAP_FONT_STYLE_BOLDITALIC QCAP_FONT_STYLE_UNDERLINE QCAP_FONT_STYLE_STRIKEOUT
ULONG	nFontSize	IN	Specify the font size used to display the text of OSD output
DWORD	dwFontColor	IN	Specify the font color used to display the text of OSD output
DWORD	dwBackgroundColor	IN	Specify the background color used to display the text of OSD output
DWORD	dwBorderColor	IN	Specify the border color of the text Only in QCAP_SET_OSD_VIRTUAL_CAMERA_TEXT_EX()

ULONG	nBorderWidth	IN	Specify the border width in pixel, set 0 to disable border. <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_TEXT_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
INT	nTextStartPosX	IN	<b>default 0</b> Specify the text scrolling start position X of the upper-left corner of OSD text
INT	nTextStartPosY	IN	<b>default 0</b> Specify the text scrolling start position Y of the upper-left corner of OSD text
ULONG	nStringAlignmentStyle	IN	<b>default QCAP_STRING_ALIGNMENT_STYLE_LEFT</b> The alignment styles are: QCAP_STRING_ALIGNMENT_STYLE_LEFT QCAP_STRING_ALIGNMENT_STYLE_NEAR QCAP_STRING_ALIGNMENT_STYLE_CENTER QCAP_STRING_ALIGNMENT_STYLE_RIGHT QCAP_STRING_ALIGNMENT_STYLE_FAR <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_TEXT_EX()</b>
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

#### Examples

*Example : Put a string "CH01" on the top in virtual camera video*

```
QCAP_SET_OSD_VIRTUAL_CAMERA_TEXT( pCamera,
    0,
    0, 0,
    -1, -1,
    "CH01", "Arial",
    QCAP_FONT_STYLE_BOLD,
    12,
    0xFF000000,
    0xFFFFFFFF,
    128,
    0, 0,
    QCAP_SEQUENCE_STYLE_FOREMOST );

QCAP_SET_OSD_VIRTUAL_CAMERA_TEXT_EX( pCamera,
    0,
    0, 0,
    -1, -1,
    "CH01", "Arial",
    QCAP_FONT_STYLE_BOLD,
    12,
    0xFF000000,
    0xFFFFFFFF,
    0,0, //border color & width
    128,
    0, 0,
    QCAP_STRING_ALIGNMENT_STYLE_LEFT,
    QCAP_SEQUENCE_STYLE_FOREMOST );
```

```
QCAP_GET_OSD_VIRTUAL_CAMERA_TEXT_BOUNDARY( pCamera,  
0,  
"CH01",  
"Arial", QCAP_FONT_STYLE_BOLD,  
12,  
&BoundaryWidth,  
&BoundaryHeight );
```

### 14.5.7 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_PICTURE

## Introduction

This OSD function displays one or more **BMP/JPG/PNG/GIF/EDL.INI** on top of virtual camera video stream.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_PICTURE()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	Specify the width of OSD output. Set -1 to use original picture width.
INT	h	IN	Specify the height of OSD output. Set -1 to use original picture height.
CHAR *	pszFilePathName	IN	Specify the image file name to display in OSD Supported extensions: " <b>BMP</b> " as 24/32-bit, " <b>JPG</b> " and " <b>PNG</b> " Supported animation by <b>GIF</b> , <b>EDL</b> , <b>INI</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Place a half-transparent PNG in virtual camera video*

```
QCAP_SET_OSD_VIRTUAL_CAMERA_PICTURE( pCamera,
0,
0, 0,
-1, -1,
"C:/SAMPLE.PNG", 128,
QCAP_SEQUENCE_STYLE_FOREMOST );
```



## 14.5.8 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_BUFFER

## 14.5.8 QCAP\_SET\_OSD\_VIRTUAL\_CAMERA\_BUFFER\_EX

### Introduction

The user can use this function to create a framebuffer object used for on-screen display in camera video stream.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_OSD\_BUFFER()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
INT	w	IN	The horizontal width of OSD output
INT	h	IN	The vertical height of OSD output
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameWidth	IN	Specify the width of frame contained in a buffer
ULONG	nFrameHeight	IN	Specify the height of frame contained in a buffer
ULONG	nFramePitch	IN	Specify the number of bytes in a scan-line. Set 0 to auto calculate by width and color space format.
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_BUFFER_EX()</b>
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_BUFFER_EX()</b>
ULONG	nCropW	IN	The width of the crop rectangle <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_BUFFER_EX()</b>
ULONG	nCropH	IN	The height of the crop rectangle <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_BUFFER_EX()</b>
DWORD	dwBorderColor	IN	Specify the border color <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_BUFFER_EX()</b>
ULONG	nBorderWidth	IN	Specify the border width <b>Only in QCAP_SET_OSD_VIRTUAL_CAMERA_BUFFER_EX()</b>
ULONG	nTransparent	IN	Specify the transparent ratio of whole OSD output, Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	<b>default 0xFFFFFFFF</b> Specify the key color of OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY) 2. 0x00FF0000 (MASK BLUE) 3. 0x0000FF00 (MASK GREEN)
ULONG	nKeyColorThreshold	IN	



### 14.5.9 QCAP\_MOVE\_OSD\_VIRTUAL\_CAMERA\_OBJECT

## Introduction

The user can use this function to move the OSD object around the video window. It is useful to scroll the text string or picture on the video display window.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
UINT	iOsdNum	IN	Specify the OSD layer number to output, range 0-511
INT	x	IN	Specify the x-coordinate of the upper-left corner of OSD output
INT	y	IN	Specify the y-coordinate of the upper-left corner of OSD output
ULONG	nSequenceStyle	IN	<b>default QCAP_SEQUENCE_STYLE_FOREMOST</b> Specify OSD sequence style type: QCAP_SEQUENCE_STYLE_FOREMOST QCAP_SEQUENCE_STYLE_BEFORE_ENCODE QCAP_SEQUENCE_STYLE_AFTERMOST
double	dLifeTime	IN	<b>default 0.0</b> Specify the OSD display duration in seconds (0 to display forever)

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To scroll the OSD object from left to right in virtual camera video*

[illegible]

# 14.6 Virtual Camera Audio Functions

## Introduction

In this section provides the audio buffer mixer functions. The user can use these function to mix a audio buffer directly with the camera audio stream.

### 14.6.1 QCAP\_SET\_AUDIO\_MX\_VIRTUAL\_CAMERA\_MIXING\_UNCOMPRESSION\_BUFFER

### 14.6.1 QCAP\_SET\_AUDIO\_MX\_VIRTUAL\_CAMERA\_MIXING\_UNCOMPRESSION\_BUFFER\_EX

## Introduction

This function can mix audio virtual uncompressed buffer on audio preview callback. The user must set **QCAP\_SET\_AUDIO\_MX\_VIRTUAL\_CAMERA\_MIXING\_UNCOMPRESSION\_BUFFER()** to mix audio record first then call **QCAP\_SET\_AUDIO\_MX\_VIRTUAL\_CAMERA\_UNCOMPRESSION\_BUFFER()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
UINT	iMixNum	IN	Specify the audio mixer number
ULONG	nChannels	IN	Specify the total audio channels <b>Only in QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_MIXING_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nBitsPerSample	IN	Specify the audio bits per sample <b>Only in QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_MIXING_UNCOMPRESSION_BUFFER_EX()</b>
ULONG	nSampleFrequency	IN	Specify the audio sample frequency <b>Only in QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_MIXING_UNCOMPRESSION_BUFFER_EX()</b>
BYTE *	pFrameBuffer	IN	Specify the audio input source buffer
ULONG	nFrameBufferLen	IN	Specify the audio input source buffer's size

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To mix audio uncompressed buffer in virtual camera*

```
QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_MIXING_UNCOMPRESSION_BUFFER( pCamera, 0,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( pCamera, 0.0 );
```

# 14.6.2 QCAP\_SET\_AUDIO\_MX\_VIRTUAL\_CAMERA\_UNCOMPRESSION\_BUFFER

## Introduction

This function can set audio virtual uncompressed buffer on audio preview callback.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the virtual camera object
double	dSampleTime	IN	<b>default 0.0</b> Specify the time-stamp of this frame.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

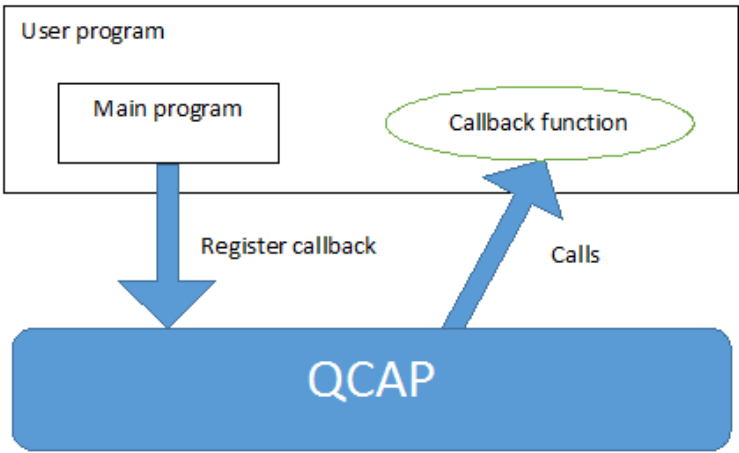
*Example : To mix audio uncompressed buffer in virtual camera*

```
QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_MIXING_UNCOMPRESSION_BUFFER( pCamera,
                                                                pFrameBuffer,
                                                                nFrameBufferLen );

QCAP_SET_AUDIO_MX_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( pCamera, 0.0 );
```

# 14.7 Virtual Camera Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_VIRTUAL_CAMERA_SNAPSHOT_DONE_CALLBACK	PF_VIRTUAL_CAMERA_SNAPSHOT_DONE_CALLBACK
QCAP_REGISTER_VIRTUAL_CAMERA_SNAPSHOT_STREAM_CALLBACK	PF_VIRTUAL_CAMERA_SNAPSHOT_STREAM_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

# 14.7.1 QCAP\_REGISTER\_VIRTUAL\_CAMERA\_SNAPSHOT\_DONE\_CALLBACK

## Introduction

This callback function is called when the snapshot file of **QCAP\_SNAPSHOT\_VIRTUAL\_CAMERA\_BMP()/JPG()** is completed. For users who use asynchronous mode, can use this function to know when snapshot file is ready.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
<b>PF_VIRTUAL_CAMERA_SNAPSHOT_DONE_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_VIRTUAL\_CAMERA\_SNAPSHOT\_DONE\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pCamera	Handle of the camera object
CHAR *	pszFilePathName	pointer to the snapshot filename
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for snapshot done in virtual camera*

```
QRETURN virtual_camera_snapshot_done( PVOID pCamera,
                                     CHAR * pszFilePathName,
                                     PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIRTUAL_CAMERA_SNAPSHOT_DONE_CALLBACK pCB = virtual_camera_snapshot_done;

    QCAP_REGISTER_VIRTUAL_CAMERA_SNAPSHOT_DONE_CALLBACK( pClient, pCB, pUserData );
}
```

# 14.7.2 QCAP\_REGISTER\_VIRTUAL\_CAMERA\_SNAPSHOT\_STREAM\_CALLBACK

## Introduction

This callback function will be called when **QCAP\_SNAPSHOT\_VIRTUAL\_CAMERA\_\*BMP/JPG()**\* image stream is generated, then a user can get image stream in buffer directly.

## Parameters

type	parameter	I/O	descriptions
PVOID	pCamera	IN	Handle of the camera object
<b>PF_VIRTUAL_CAMERA_SNAPSHOT_STREAM_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# PF\_VIRTUAL\_CAMERA\_SNAPSHOT\_STREAM\_CALLBACK

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pCamera	Handle of the camera object
CHAR *	pszFilePathName	pointer to the snapshot filename
BYTE *	pStreamBuffer	Pointer to the image framebuffer
ULONG	nStreamBufferLen	Specify the length of image framebuffer
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function for snapshot stream completion in virtual camera*

```
QRETURN virtual_camera_snapshot_stream( PVOID pCamera,
                                       CHAR * pszFilePathName
                                       BYTE * pStreamBuffer,
                                       ULONG nStreamBufferLen,
                                       PVOID pUserData )
{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_VIRTUAL_CAMERA_SNAPSHOT_STREAM_CALLBACK pCB = virtual_camera_snapshot_stream;

    QCAP_REGISTER_VIRTUAL_CAMERA_SNAPSHOT_STREAM_CALLBACK( pClient, pCB, pUserData );
}
```



# 15 CD&DVD Burning Function API

---

## Introduction



This chapter provides a CD, DVD, and Blu-ray, HD burning API functions. QCAP offers a well-rounded and easy-to-use API for fast and easy implementation of burning solutions. This is especially useful when a user wants to backup capture video files to external discs for security / medical / broadcasting applications.

### Burning programming guides

The practice Burning example:

- STEP 01 **QCAP\_CREATE\_BURNING\_DRIVE**( 'E', "TEST", &pDrive );
- STEP 02 **QCAP\_START\_BURNING\_DRIVE**( pDrive );
- STEP 03 **QCAP\_START\_RECORD**( pDevice, 0, "E:/CH01.TS" );
- STEP 04 **QCAP\_STOP\_RECORD**( pDevice, 0, FALSE, INFINE );
- STEP 05 **QCAP\_STOP\_BURNING\_DRIVE**( pDrive );
- STEP 06 **QCAP\_UNLOAD\_DISC**( pDrive );
- STEP 07 **QCAP\_DESTROY\_BURNING\_DRIVE**( pDrive );

# 15.1 QCAP\_CREATE\_BURNING\_DRIVE

## Introduction

The user can use this function to create a burning drive.



The user can find driver name of DVD-RW drive on Devices with Removable Storage.

## Parameters

type	parameter	I/O	descriptions
CHAR	cDriveName	IN	Specify the driver name
CHAR *	pVolumeName	IN	Specify the file name
PVOID *	ppDrive	OUT	Pointer to the address of CD&DVD driver

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Create a burning drive*

```
PVOID pDrive = 0;

QCAP_CREATE_BURNING_DRIVE( 'E', "TEST", &pDrive );

QCAP_START_BURNING_DRIVE( pDrive );

QCAP_START_RECORD( pDevice, 0, "E:/CH01.TS" );

QCAP_STOP_RECORD( pDevice, 0, FALSE, INFINITE );

//Note: Need to wait for file closing completion

QCAP_STOP_BURNING_DRIVE( pDrive );

QCAP_UNLOAD_DISC( pDrive );

QCAP_DESTROY_BURNING_DRIVE( pDrive );
```

## 15.2 QCAP\_START\_BURNING\_DRIVE

### Introduction

The user can use this function to start burning drive.



The video record files are only support **TS** and **FLV** format.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the driver

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Start a burning drive*

```
QCAP_START_BURNING_DRIVE( pDrive );
```

## 15.3 QCAP\_STOP\_BURNING\_DRIVE

### Introduction

The user can use this function to stop burning drive.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the driver

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Stop a burning drive*

```
QCAP_STOP_BURNING_DRIVE( pDrive );
```

## 15.4 QCAP\_LOAD\_DISC

### Introduction

The user can use this function to load a disc.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the driver

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To load a disc.*

```
QCAP_LOAD_DISC( pDrive );
```

---

## 15.5 QCAP\_UNLOAD\_DISC

### Introduction

The user can use this function to unload disc.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the driver

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To unload a disc.*

```
QCAP_UNLOAD_DISC( pDrive );
```

# 15.6 QCAP\_GET\_DISC\_INFO

## Introduction

The user can use this function to get burn disc type information,

the disc type as show in below table.

INFO Value	DISC Type	INFO Value	DISC Type
0x0008	CD-ROM	0x0020	DDCD-ROM
0x0009	CD-R	0x0021	DDCD-R
0x000A	CD-RW	0x0022	DDCD-RW
0x0010	DVD-ROM	0x0040	BD-ROM
0x0011	DVD-R	0x0041	BD-R
0x0012	DVD-RAM	0x0042	BD-R
0x0013	DVD-RW	0x0043	BD-RE
0x0014	DVD-RW	0x0050	HD-ROM
0x0015	DVD-R DL	0x0051	HD-R
0x0016	DVD-R DL	0x0052	HD-RAM
0x001A	DVD+RW	0x0053	HD-RW
0x0017	DVD-RW DL	0x005A	HD-RW DL
0x001B	DVD+R		
0x002A	DVD+RW DL		
0x002B	DVD+R DL		
0x0052	DVD-R DL		
0X1314	DVD-RW	0x0000	Errors DISC

## Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the driver
BOOL *	pIsDiscBlank	OUT	Pointer of disc blank
BOOL *	pIsDiscWriteable	OUT	Pointer of disc writable
DWORD *	pDiscType	OUT	Pointer of disc type
ULONGLONG *	pDiscRemainCapability	OUT	Pointer of disc remain capability

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get Disc type information*

```
QCAP_GET_DISC_INFO( pDrive,  
                    &nIsDiscBlank,  
                    &nIsDiscWriteable,  
                    &nDiscType,  
                    &nDiscRemainCapability );
```

C

## 15.7 QCAP\_ERASE\_DISC

### Introduction

The user can use this function to erase the disc.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the driver

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To erase a ReWritable disc*

```
QCAP_ERASE_DISC( pDrive );
```

---

## 15.8 QCAP\_DESTROY\_BURNING\_DRIVE

### Introduction

The user can use this function to destroy burning drive.

### Parameters

type	parameter	I/O	descriptions
PVOID	pDrive	IN	Handle of the CD&DVD RW driver

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To destroy a burning drive.*

```
QCAP_DESTROY_BURNING_DRIVE( pDrive );
```

# 16 Camera Function API

---

## Introduction



This chapter provides functions to enumerate system camera attached on the platform, and can also set the default video input on a system camera.



# 16.1 QCAP\_CAMERA\_ENUMERATION

## Introduction

This function enumerates all system camera devices on the platform.

## Parameters

type	parameter	I/O	descriptions
CHAR **	ppszCameraDevName	OUT	Pointer to the address of all camera devices
BOOL	bNext	IN	<b>default FALSE</b> Specify to not use first camera on the platform. * Set FALSE to use the first camera available. * Set TRUE, to use any camera other than the first one.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To enumerates all system camera attached*

```
CHAR *camera_name = NULL;

//get the 1st camera available
QRESULT qr = QCAP_CAMERA_ENUMERATION( &camera_name, FALSE );

printf("name: %s\n", camera_name );

//get the other camera available
while( qr == QCAP_RS_SUCCESSFUL )
{
    qr = QCAP_CAMERA_ENUMERATION( &camera_name, TRUE );

    printf("name: %s\n", camera_name );
}
```

## 16.2 QCAP\_SET\_DEFAULT\_CAMERA

### Introduction

The user can use this function to modify the default DirectShow Video Capture Filter's name. In Windows 7, the usual friendly name of VideoCapture Filter is "XXX" in different language systems; So QCAP uses string, "Camera", as default.

This function is to modify the default name of *DirectShow Video Capture Filter*.

In Windows 7, the name of *VideoCapture Filter* is usually **"XYZ Camera"** no matter what system language is.

So in order to match a name in many languages of windows system, QCAP use keyword string "Camera" as default.

### Parameters

type	parameter	I/O	descriptions
CHAR *	pszCameraDevName	IN	<b>default "Camera"</b> Specify capture filter keyword of camera name

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : To set the default video input from system camera*

```
QCAP_SET_DEFAULT_CAMERA( "Camera1" ); // For Windows [ENG] Edition

QCAP_SET_DEFAULT_CAMERA( "攝影機" ); // For Windows [CHT] Edition

QCAP_SET_DEFAULT_CAMERA( "Camera" ); // Default
```

# 17 Sound Card Function API

---

## Introduction



This chapter provides functions to enumerate system sound cards on the platform, and can also set the default sound input on a system sound card.

# 17.1 QCAP\_SOUND CARD\_ENUMERATION

## Introduction

This function enumerates all system sound card devices on the platform.

## Parameters

type	parameter	I/O	descriptions
CHAR **	ppszSoundCardDevName	OUT	Pointer to the address of all sound card devices
BOOL	bNext	IN	<b>default FALSE</b> Specify to not use first sound card on the platform. * Set FALSE to use the first sound card available. * Set TRUE, to use any sound card other than the first one.

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To enumerates all system sound card devices*

```
CHAR *soundcard_name = NULL;

//get the 1st sound card available
QRESULT qr = QCAP_SOUND CARD_ENUMERATION( soundcard_name, FALSE );

printf("name: %s\n", soundcard_name );

//get the other sound cards available
while( qr == QCAP_RS_SUCCESSFUL )
{
    qr = QCAP_SOUND CARD_ENUMERATION( soundcard_name, TRUE );

    printf("name: %s\n", soundcard_name );
}
```

# 17.2 QCAP\_SET\_DEFAULT\_SOUND CARD

## Introduction

This function is to modify the default name of *DirectShow Audio Capture Filter*.  
In Windows 7, the name of *AudioCapture Filter* is usually "ABC ( ABC High Definition Audio )" no matter what system language is.  
So in order to match a name in many languages of windows system, QCAP use keyword string "High" as default.

## Parameters

type	parameter	I/O	descriptions
CHAR *	pszMicrophoneDevName	IN	<b>default "High"</b> Specify capture filter keyword of microphone name
CHAR *	pszLineInDevName	IN	<b>default "High"</b> Specify capture filter keyword of line-in name

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To set the default sound input from system sound card*

```
QCAP_SET_DEFAULT_SOUND CARD( "Microphone", "Line In" ); // For Windows [ENG] Edition

QCAP_SET_DEFAULT_SOUND CARD( "麥克風", "線路輸入" );      // For Windows [CHT] Edition

QCAP_SET_DEFAULT_SOUND CARD( "麦克风", "线路输入" );      // For Windows [CHS] Edition

QCAP_SET_DEFAULT_SOUND CARD( "High", "High" );           // Default
```

# 17.3 QCAP\_SOUNDRENDERER\_ENUMERATION

## Introduction

This function can enumerate the sounder renderer available on the system.

## Parameters

type	parameter	I/O	descriptions
CHAR **	ppszSoundRendererDevName	OUT	Specify the pointer of the sound renderer string output
BOOL	bNext	IN	<b>default FALSE</b> The flag to enumerate next sound renderer or not

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To enumerate the sound renderer*

```
QCAP_SOUNDRENDERER_ENUMERATION( &SoundRenender, false );
```

C

# 18 Helper Function API

---

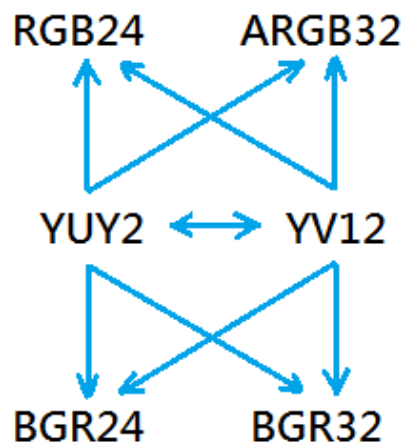
## Introduction



Helper Functions

The helper function is functions that perform part of the computation of another function. In this chapter, helper functions are useful to make your programs easier converts between framebuffers, resize an image, rotate buffers, re-scaling, audio re-sampling...etc. They also let you reuse computations, just as with functions in general.

Different color format can convert to other buffer format, the supported color space converting graph:



# 18.1 QCAP\_GET\_H264\_BUFFER\_LAYER\_ID

## Introduction

This function for a user to get ID layer number of **H.264**.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pStreamBuffer	IN	Specify the source framebuffer
ULONG	nStreamBufferLen	IN	Specify the source width
ULONG *	pLayerID	OUT	Returned the current ID layer number of <b>H.264</b>

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get H.264 ID layer number*

```
ULONG nLayerID = 0;
BYTE *pStreamBuffer;    //H.264 data stream
ULONG nStreamBufferLen; //length of H.264 stream

QCAP_GET_H264_BUFFER_LAYER_ID( pStreamBuffer, nStreamBufferLen, &nLayerID );
```



# 18.4 QCAP\_COLORSPACE\_YUY2\_TO\_YV12

## Introduction

This function can convert color buffer from **YUY2** to **YV12**.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YUY2** to **YV12***

```
QCAP_COLORSPACE_YUY2_TO_YV12( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                FALSE,
                                FALSE );
```

# 18.7 QCAP\_COLORSPACE\_YV12\_TO\_YUY2

## Introduction

This function can convert color buffer from **YV12** to **YUY2**.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from YV12 to YUY2*

```
QCAP_COLORSPACE_YV12_TO_YUY2 ( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                FALSE,
                                FALSE );
```

# 18.2 QCAP\_COLORSPACE\_YUY2\_TO\_ABGR32

## Introduction

This function can convert color buffer from **YUY2** to ABGR32.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BYTE	bAlpah	IN	<b>default 0x00</b> Specify the alpha of target frame, the default 0x00
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YUY2** to ABGR32*

```
QCAP_COLORSPACE_YUY2_TO_ABGR32( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                0x00,
                                FALSE,
                                FALSE );
```

# QCAP\_COLORSPACE\_YUY2\_TO\_ARGB32

## Introduction

This function can convert color buffer from **YUY2** to ARGB32.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BYTE	bAlpah	IN	<b>default 0x00</b> Specify the alpha of target frame, the default 0x00
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YUY2** to ARGB32*

```
QCAP_COLORSPACE_YUY2_TO_ARGB32( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                0x00,
                                FALSE,
                                FALSE );
```

# 18.3 QCAP\_COLORSPACE\_YUY2\_TO\_BGR24

## Introduction

This function can convert color buffer from **YUY2** to BGR24



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YUY2** to **BGR24***

```
QCAP_COLORSPACE_YUY2_TO_BGR24( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                FALSE,
                                FALSE );
```

# QCAP\_COLORSPACE\_YUY2\_TO\_RGB24

## Introduction

This function can convert color buffer from **YUY2** to RGB24.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BYTE	bAlpah	IN	<b>default 0x00</b> Specify the alpha of target frame, the default 0x00
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YUY2** to RGB24*

```
QCAP_COLORSPACE_YUY2_TO_RGB24( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                0x00,
                                FALSE,
                                FALSE );
```

# 18.5 QCAP\_COLORSPACE\_YV12\_TO\_ABGR32

## Introduction

This function can convert color buffer from **YV12** to **ABGR32**.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BYTE	bAlpah	IN	<b>default 0x00</b> Specify the alphah of target frame.
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YV12** to **ABGR32***

```
QCAP_COLORSPACE_YV12_TO_ABGR32( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                0x00,
                                FALSE,
                                FALSE );
```

# QCAP\_COLORSPACE\_YV12\_TO\_ARGB32

## Introduction

This function can convert color buffer from **YV12** to ARGB32.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BYTE	bAlpah	IN	<b>default 0x00</b> Specify the alpha of target frame, the default 0x00
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YV12** to ARGB32*

```
QCAP_COLORSPACE_YV12_TO_ARGB32( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                0x00,
                                FALSE,
                                FALSE );
```



# 18.6 QCAP\_COLORSPACE\_YV12\_TO\_BGR24

## Introduction

This function can convert color buffer from **YV12** to **BGR24**.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from YV12 to BGR24*

```
QCAP_COLORSPACE_YV12_TO_BGR24( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                FALSE,
                                FALSE );
```

# QCAP\_COLORSPACE\_YV12\_TO\_RGB24

## Introduction

This function can convert color buffer from **YV12** to RGB24.



The buffer size must match between input source buffer and the output source buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of target frame
ULONG	nDstHeight	IN	Specify the height of target frame
ULONG	nDstPitch	IN	Specify the pitch of target frame
BYTE	bAlpah	IN	<b>default 0x00</b> Specify the alpha of target frame, the default 0x00
BOOL	bHorizontalMirror	IN	<b>default FALSE</b> Enable/Disable HorizontalMirror
BOOL	bVerticalMirror	IN	<b>default FALSE</b> Enable/Disable VerticalMirror

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a color buffer from **YV12** to RGB24*

```
QCAP_COLORSPACE_YV12_TO_RGB24( pSrcFrameBuffer,
                                nSrcWidth,
                                nSrcHeight,
                                nSrcPitch,
                                &nDstFrameBuffer,
                                nDstWidth,
                                nDstHeight,
                                nDstPitch,
                                0x00,
                                FALSE,
                                FALSE );
```

# QCAP\_GET\_AUDIO\_BUFFER\_FAST\_FOURIER\_TRANSFORM\_DATA

## Introduction

Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. This function performs Fourier analysis to an audio framebuffer and returns the peak frequency and its amplitude.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer size
ULONG	nChannels	IN	The total audio channels (e.g. stereo or mono)
ULONG	nBitsPerSample	IN	The audio bits per sample (e.g. 8bits, 16bits)
ULONG	nSampleFrequency	IN	The audio sampling rate (e.g. 44kHz, 16kHz)
UINT	iChNum	IN	The channel number to get <b>SCF</b> file parameters, start from 0
double *	pPeakFrq	OUT	TBD the peak frequency of FFT result
double *	pPeakFrqAmp	OUT	TBD the amplitude value of the peak frequency of FFT result

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Do a FFT to a audio framebuffer and get it's amplitude value*

```
QCAP_GET_AUDIO_BUFFER_FAST_FOURIER_TRANSFORM_DATA( pFrameBuffer,
nFrameBufferLen,
2, 16, 48000,
0,
&pPeakFrq,
&pPeakFrqAmp );
```

9

# 18.8 QCAP\_CONVERT\_3D\_STEREO\_BUFFER

## Introduction

This function can help to convert 3D stereo buffers format within Side-by-Side (SBS), Top-Bottom (TB), Line-by-Line (LBL)...etc.

## Parameters

type	parameter	I/O	descriptions
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
ULONG	nSrcStereoDisplayMode	IN	Specify source 3D stereo display mode: QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE QCAP_3D_STEREO_DISPLAY_MODE_LEFT_ONLY. QCAP_3D_STEREO_DISPLAY_MODE_RIGHT_ONLY.
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
ULONG	nDstStereoDisplayMode	IN	Specify another 3D stereo display mode: QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM QCAP_3D_STEREO_DISPLAY_MODE_LINE_BY_LINE QCAP_3D_STEREO_DISPLAY_MODE_LEFT_ONLY. QCAP_3D_STEREO_DISPLAY_MODE_RIGHT_ONLY.
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of destination frame
ULONG	nDstHeight	IN	Specify the height of destination frame
ULONG	nDstPitch	IN	Specify the pitch of destination frame
BOOL	bLeftRightSwap	IN	<b>default FALSE</b> Swap the video left/right outputs from source to destination

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert a 3D buffer format from Side-by-Side to Top-Bottom*

```
QCAP_CONVERT_3D_STEREO_BUFFER( QCAP_COLORSPACE_TYEP_YUY2,  
                                QCAP_3D_STEREO_DISPLAY_MODE_SIDE_BY_SIDE,  
                                &nSrcFrameBuffer,  
                                nSrcWidth,  
                                nSrcHeight,  
                                nSrcPitch,  
                                QCAP_3D_STEREO_DISPLAY_MODE_TOP_BOTTOM,  
                                nDstWidth,  
                                nDstHeight,  
                                nDstPitch,  
                                FALSE );
```

# 18.9 QCAP\_RESIZE\_VIDEO\_BUFFER

## Introduction

This function scales a video buffer in any color space.

## Parameters

type	parameter	I/O	descriptions
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of destination frame
ULONG	nDstHeight	IN	Specify the height of destination frame
ULONG	nDstPitch	IN	Specify the pitch of destination frame

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Scale down an ARGB color buffer to 1920x1080 to 720x480*

```
QCAP_RESIZE_VIDEO_BUFFER( QCAP_COLORSPACE_TYEP_ARGB32,
    &nSrcFrameBuffer,
    1920,
    1080,
    1080*4,
    &nDstFrameBuffer,
    720,
    480,
    480*4 );
```

# 18.10 QCAP\_GET\_ROTATE\_VIDEO\_BUFFER\_BOUNDARY

## Introduction

This function is used to get boundary the result of rotation.  
This function can peek the new resolution (W/H) after a rotation of **QCAP\_ROTATE\_VIDEO\_BUFFER()**. Then a user can prepare the right buffer size that can fit the result buffer.

## Parameters

type	parameter	I/O	descriptions
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG *	pDstWidth	OUT	Pointer to the horizontal width of destination frame
ULONG *	pDstHeight	OUT	Pointer to the vertical height of destination frame
double	dAngle	IN	Specify the rotate angle of source, range is 0-360

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Peek the resolution of a buffer after rotation.*

```
QCAP_GET_ROTATE_BOUNDARY( nSrcWidth,
                          nSrcHeight,
                          nSrcPitch,
                          &nDstWidth,
                          &nDstHeight,
                          dAngle );
```

# 18.11 QCAP\_ROTATE\_VIDEO\_BUFFER

## Introduction

This can rotate an image in source buffer use the desired angle, and the result will save to a destination buffer.

This function can peek the new resolution (W/H) after a rotation of **QCAP\_ROTATE\_VIDEO\_BUFFER()**. Then a user can prepare the right buffer size that can fit the result buffer.

## Parameters

type	parameter	I/O	descriptions
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
BYTE *	pDstFrameBuffer	OUT	Specify the destination framebuffer
ULONG	nDstWidth	IN	Specify the width of destination frame
ULONG	nDstHeight	IN	Specify the height of destination frame
ULONG	nDstPitch	IN	Specify the pitch of destination frame
double	dAngle	IN	Specify the rotate angle of source, range is 0-360
BYTE *	pSrcTempFrameBuffer	IN	<b>default NULL</b> Specify the temp source framebuffer
BYTE *	pDstTempFrameBuffer	IN	<b>default NULL</b> Specify the temp Frame another buffer
BOOL	bClearBackground	IN	<b>default TRUE</b> If true, the background color will be remove

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.



## Examples

*Example : Rotate the color buffer by a 45 degree*

```
double dAngle = 45.0;

ULONG nDstWidth = 0;

ULONG nDstHeight = 0;

ULONG nDstPitch = nSrcPitch;    //src and dest pitch are the same

QCAP_GET_ROTATE_BOUNDARY( nSrcWidth,
                          nSrcHeight,
                          nSrcPitch,
                          &nDstWidth,
                          &nDstHeight,
                          dAngle );

//allocate a new size buffer that can fit the rotation result

BYTE *nDstFrameBuffer = malloc( nDstHeight x nSrcPitch );

QCAP_ROTATE_VIDEO_BUFFER( QCAP_COLORSPACE_TYEP_ARGB32,
                          &nSrcFrameBuffer,
                          nSrcWidth,
                          nSrcHeight,
                          nSrcPitch,
                          &nDstFrameBuffer,
                          nDstWidth,
                          nDstHeight,
                          nDstPitch,
                          dAngle,
                          NULL,
                          NULL,
                          TRUE );
```

## 18.12 QCAP\_LOAD\_PICTURE\_BUFFER

### Introduction

This function can load an image file into picture buffer.

The user can set **pFrameBuffer** to NULL to query image buffer desired size in *nFrameBufferSize*, then allocate a buffer size that can fit the resulting image. Then call **QCAP\_LOAD\_PICTURE\_BUFFER()** again to actual load the image file.

### Parameters

type	parameter	I/O	descriptions
CHAR *	pszFilePathName	IN	Specify the picture file name, supported format: <b>BMP,PNG,JPG</b>
ULONG *	pColorSpaceType	OUT	Pointer to the color space type for the picture
BYTE *	pFrameBuffer	OUT	Pointer to the picture buffer Set NULL to query the result buffer's size and stores in <b>pFrameBufferSize</b>
ULONG *	pFrameBufferSize	IN/OUT	Pointer to the picture buffer's size
ULONG *	pFrameWidth	OUT	Pointer to the picture width
ULONG *	pFrameHeight	OUT	Pointer to the picture Height
ULONG *	pFramePitch	OUT	Pointer to the picture pitch

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : Load a **BMP** image file to a memory buffer*

```
ULONG nFrameBufferSize=0;

//Query the desire buffer size for image file

QCAP_LOAD_PICTURE_BUFFER( "Pitcure.bmp",
                          &nColorSpaceType,
                          NULL,
                          &nFrameBufferSize,
                          &nFrameWidth,
                          &nFrameHeight,
                          &nFramePitch );

//allocate enough buffer size to fit the result

BYTE *nFrameBuffer = malloc( nFrameBufferSize );

//Then load the image file to buffer

QCAP_LOAD_PITCURE_BUFFER( "Pitcure.bmp",
```

```
&nColorSpaceType,  
&nFrameBuffer,  
nFrameBufferSize,  
&nFrameWidth,  
&nFrameHeight,  
&nFramePitch );
```

## 18.13 QCAP\_CALCULATE\_CHROMAKEY

### Introduction

This function can help to compute the chroma-key from a source framebuffer to a destination alpha-blending buffer.

### Parameters

type	parameter	I/O	descriptions
ULONG	nColorSpaceType	IN	Specify encoder color space type: QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_Y800 QCAP_COLORSPACE_TYEP_H264 QCAP_COLORSPACE_TYEP_H265
BYTE*	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcWidth	IN	Specify the source width
ULONG	nSrcHeight	IN	Specify the source height
ULONG	nSrcPitch	IN	Specify the source pitch
ULONG	nCropX	IN	The x-coordinate of the upper-left corner of the crop rectangle
ULONG	nCropY	IN	The y-coordinate of the upper-left corner of the crop rectangle
ULONG	nCropW	IN	The width of the crop rectangle
ULONG	nCropH	IN	The height of the crop rectangle
BYTE *	pDstAlpahBuffer	OUT	Specify the destination A alpha-blending framebuffer
BYTE *	pDstYBuffer	OUT	Specify the destination Y luminate framebuffer
BYTE *	pDstCbBuffer	OUT	Specify the destination Cb color framebuffer
BYTE *	pDstCrBuffer	OUT	Specify the destination Cr color framebuffer
ULONG	nDstWidth	IN	Specify the width of destination frame
ULONG	nDstHeight	IN	Specify the height of destination frame
ULONG	nDstPitch	IN	Specify the pitch of destination frame
ULONG	nTransparent	IN	Specify the transparent value Set 255 means no transparent, range 0-255
DWORD	dwKeyColor	IN	<b>default 0xFFFFFFFF</b> Specify the key color of OSD in color type <b>ARGB</b> : 1. 0xFFFFFFFF (NO COLORKEY) 2. 0x00FF0000 (MASK BLUE) 3. 0x0000FF00 (MASK GREEN)
ULONG	nKeyColorThreshold	IN	<b>default 25</b> Specify the threshold of key color, the range from 0 to 128
ULONG	nKeyColorBlurLevel	IN	<b>default 2</b> Specify the blur level, range 0-2
BOOL	bKeyColorSpillSuppress	IN	<b>default TRUE</b> Specify the color spill suppress value
ULONG	nKeyColorSpillSuppressThreshold	IN	<b>default 22</b> Specify the threshold value of color spill suppress

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : get the chroma-key alpha-blending buffer from a source framebuffer*

```
ULONG nFramebufferSize=0;

//Query the desire buffer size for image file

QCAP_CALCULATE_CHROMAKEY( pSrcFramebuffer, 1920, 1080, 1920*4,
                          0,0,0,0,
                          &DstABuffer,&DstYBuffer,&DstCbBuffer,&DstCrBuffer,
                          1920, 1080, 1920*4,
                          0,
                          0xFFFFFFFF,
                          25, 2, TRUE);
```

C

# 18.14 QCAP\_GET\_AUDIO\_BUFFER\_VOLUME\_DB

## Introduction

This function help to get channel volume of audio buffer in *The Decibel (dB) Scale*.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
ULONG	nChannels	IN	Specify the total audio channels
ULONG	nBitsPerSample	IN	Specify the audio bits per sample
ULONG	nSampleFrequency	IN	Specify the audio sample frequency
UINT	iChNum	IN	Specify the index of channel 0 is the left channel 1 is the right channel
double *	pVolumeDB	OUT	<b>RANGE = -100 to 0</b> Pointer to the volume value

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Get the Left/Right channel volume in dB from an audio buffer*

```
QCAP_GET_AUDIO_BUFFER_VOLUME_DB( pFrameBuffer,
                                nFrameBufferLen,
                                2, 16, 48000,
                                0,
                                &VolumeDB_L ); //Left Channel volume in dB

QCAP_GET_AUDIO_BUFFER_VOLUME_DB( pFrameBuffer,
                                nFrameBufferLen,
                                2, 16, 48000,
                                1,
                                &VolumeDB_R ); //Right Channel volume in dB
```

# 18.15 QCAP\_RESAMPLE\_AUDIO\_BUFFER

## Introduction

This function can re-sampling an audio buffer to another format.

For example, transform an audio buffer from (2CH 16Bit 48000HZ) to (1Ch 8Bit 44100HZ).

The user can set *pFrameBuffer* to NULL to query audio buffer desired size in *nFrameBufferSize*, then allocate a buffer size that can fit the resulting audio. Then call **QCAP\_RESAMPLE\_AUDIO\_BUFFER()** again to actual re-sampling the audio buffer.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pSrcFrameBuffer	IN	Specify the source framebuffer
ULONG	nSrcFrameBufferLen	IN	Specify the source buffer's size
ULONG	nSrcChannels	IN	Specify the source total audio channels
ULONG	nSrcBitsPerSample	IN	Specify the source audio bits per sample
ULONG	nSrcSampleFrequency	IN	Specify the source audio sample frequency
BYTE *	pDstFrameBuffer	OUT	Pointer to the target framebuffer <a href="#">Set NULL to query the result buffer's size and stores in pDstFrameBufferLen</a>
ULONG *	pDstFrameBufferLen	IN/OUT	Specify the target framebuffer's size
ULONG	nDstChannels	IN	Specify the target total audio channels
ULONG	nDstBitsPerSample	IN	Specify the target audio bits per sample
ULONG	nDstSampleFrequency	IN	Specify the target audio sample frequency

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Convert audio buffer format from (Stereo,16bits,44Khz) to (Mono,8bit,44Khz)*

```
ULONG nDstAudioBufferLen = 0;

BYTE *nDstAudioBuffer = NULL;

//Query the desire buffer size for audio re-sampling

QCAP_RESAMPLE_AUDIO_BUFFER( pFrameBuffer,
                             nFrameBufferLen,
                             2, 16, 48000,
                             NULL,
                             &nDstAudioBufferLen,
                             1, 8, 44100 );
```

```
//allocate enough buffer size to fit the result

BYTE *nDstAudioBuffer = malloc( nDstAudioBufferLen );

//Then performs the audio re-sampling

QCAP_RESAMPLE_AUDIO_BUFFER( pFrameBuffer,
                             nFrameBufferLen,
                             2, 16, 48000,
                             nDstAudioBuffer,
                             &nDstAudioBufferLen,
                             1, 8, 44100 );
```



# 18.16 QCAP\_RESCALE\_AUDIO\_BUFFER

## Introduction

This function can rescale volume of an audio buffer.

The *nVolume* parameter:

- set to 100 is double the audio amplitude.
- set 50 for the original volume,
- set 25 to half the audio amplitude.

## Parameters

type	parameter	I/O	descriptions
BYTE *	pFrameBuffer	IN	Specify a raw data of frame contained in a buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
ULONG	nChannels	IN	Specify the source total audio channels
ULONG	nBitsPerSample	IN	Specify the source audio bits per sample
ULONG	nSampleFrequency	IN	Specify the source audio sample frequency
ULONG	nVolume	IN	Specify the target audio volume, range from 0-100, 50 is original volume

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Double the audio amplitude*

```
QCAP_RESCALE_AUDIO_BUFFER( pFrameBuffer,
                           nFrameBufferLen,
                           2, 16, 48000,
                           100 );
```

# 18.17 QCAP\_HELPER\_OBJPTR

## Introduction

This is help function for .NET developer to get a handle of helper object.

## Parameters

type	parameter	I/O	descriptions
PVOID	pObj	IN	Handle of the helper object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To get the handle of helper object*

```
QCAP_HELPER_OBJPTR( pObj );
```

# 19 Media Timer Function API

---

## Introduction



This chapter helps user to create, set, and delete media timers. The user can schedule timer with a callback function after a specified time has elapsed. Each time the specified interval for a timer elapses, the system notifies the callback function associated with the timer. A media timer's accuracy depends on the system clock rate.

# 19.1 QCAP\_CREATE\_MEDIA\_TIMER

## Introduction



This function provides multimedia timer services allow a user to schedule periodic timer events, then a user can receive timer callback at user-defined intervals. The user uses this function to create a high-resolution media timer and set its elapsed time interval. for example, a practical media timer can be used in video playback or in share recording.

For example, if user wants to set time interval to  $3/4 = 0.75$  seconds:

- set *nElapseTimeNum* (the numerator) : 3
- set *nElapseTimeDeno* (the denominator) : 4

## Parameters

type	parameter	I/O	descriptions
ULONG	nElapseTimeNum	IN	Specify the elapse time numerator
ULONG	nElapseTimeDeno	IN	Specify the elapse time denominator
PVOID *	ppTimer	OUT	Handle of the timer object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Create a media timer and register its callback function*

```
QRETURN media_timer_function( PVOID pTimer,
                             double dSampleTime,
                             double dDelayTime,
                             PVOID pUserData )
{
    //timer callback when user-defined interval elapsed

    return QCAP_RT_OK;
```

```

}

VOID pTimer = NULL;

void test_timer()
{
    QCAP_CREATE_MEDIA_TIMER( 3, 4, &pTimer );

    //register media timer callback

    PF_MEDIA_TIMER_CALLBACK pCB = media_timer_function;

    QCAP_REGISTER_MEDIA_TIMER_CALLBACK( pTimer, pCB, pUserData );

    QCAP_START_MEDIA_TIMER( pTimer );

    //...

    //media timer is counting

    //...

    QCAP_STOP_MEDIA_TIMER( pTimer );

    QCAP_DESTROY_MEDIA_TIMER( pTimer );

}

```

## 19.2 QCAP\_START\_MEDIA\_TIMER

### Introduction

The user can use this function to start a media timer.

### Parameters

type	parameter	I/O	descriptions
PVOID	pTimer	IN	Handle of the Timer object

### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

### Examples

*Example : start a media timer.*

```

QCAP_START_MEDIA_TIMER( pTimer );

```

# 19.3 QCAP\_STOP\_MEDIA\_TIMER

## Introduction

The user can use this function to stop media timer.

## Parameters

type	parameter	I/O	descriptions
PVOID	pTimer	IN	Handle of the Timer object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Start a media timer*

```
QCAP_STOP_MEDIA_TIMER( pTimer );
```

# 19.4 QCAP\_DESTROY\_MEDIA\_TIMER

## Introduction

When a timer is no longer needed, this function can destroy media timer object and release its resource.

## Parameters

type	parameter	I/O	descriptions
PVOID	pTimer	IN	Handle of the Timer object

## Return value

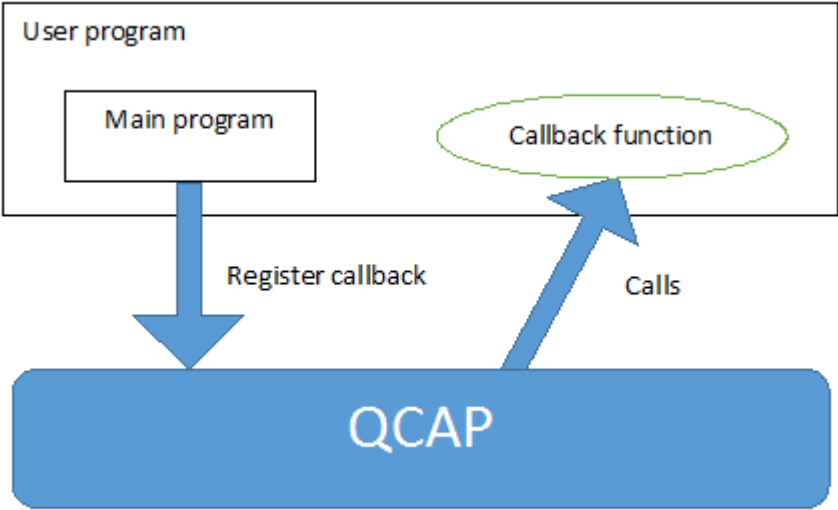
Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Close a media timer object and release resource*

# 19.5 Media Timer Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_MEDIA_TIMER_CALLBACK	PF_MEDIA_TIMER_CALLBACK

# 19.5.1 QCAP\_REGISTER\_MEDIA\_TIMER\_CALLBACK

## Introduction

The user can register a *PF\_MEDIA\_TIMER\_CALLBACK* function to get notification when the ideal time interval is reached. This callback function needs to be registered before share recording starts.

## Parameters

type	parameter	I/O	descriptions
PVOID	pTimer	IN	Handle of the Timer object
<i>PF_MEDIA_TIMER_CALLBACK</i>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# *PF\_MEDIA\_TIMER\_CALLBACK*

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pTimer	pointer to handle of media timer
double	dSampleTime	The sampling time in seconds
double	dDelayTime	Specify the delay time
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a media timer callback to receive media timer notification*

```
QRETURN media_timer_function( PVOID pTimer,
                             double dSampleTime,
                             double dDelayTime,
                             PVOID pUserData );

{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_MEDIA_TIMER_CALLBACK pCB = media_timer_function;

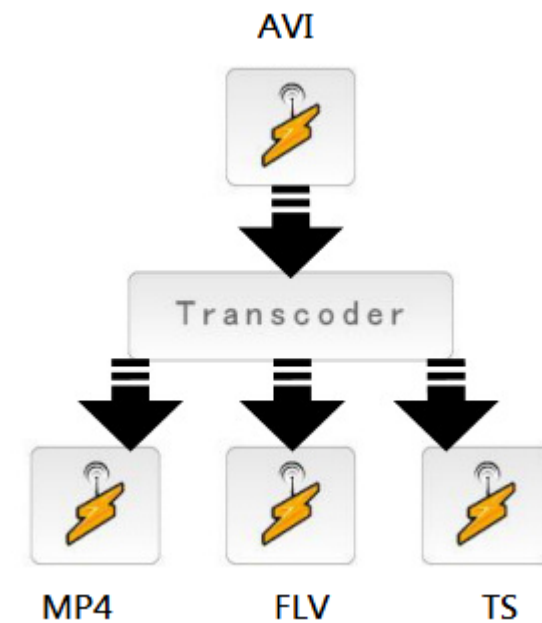
    QCAP_REGISTER_MEDIA_TIMER_CALLBACK( pTimer, pCB, pUserData );
}
```



# 20 File Transcoding Function API

---

## Introduction



QCAP provide the easy-to-use way to transcode the video and convert it to your favorite format in supported list. in this chapter video transcoding functions allows you to seamlessly integrate your application with extremely fast and scalable encoding API.

# 20.1 QCAP\_CREATE\_FILE\_TRANSCODER

## Introduction

This function can help a user to process that converts a video file format from one to another.

To convert a video file, first to create file transcoder object along with the location of a source video. It also provides the detail information about source video file. The user can set optional encoding properties. Then is ready to start an encoding job by **QCAP\_START\_FILE\_TRANSCODER()**.

## Parameters

type	parameter	I/O	descriptions
CHAR *	pszSrcFileName	IN	Specify the source video file name, supported file extensions: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF</b>
PVOID *	ppFileTranscoder	OUT	Handle of the file transcoder object
ULONG	nDecoderType	IN	Specify the decoder type: QCAP_DECODER_TYPE_SOFTWARE QCAP_DECODER_TYPE_HARDWARE QCAP_DECODER_TYPE_INTEL_MEDIA_SDK QCAP_DECODER_TYPE_AMD_STREAM QCAP_DECODER_TYPE_NVIDIA_CUDA QCAP_DECODER_TYPE_NVIDIA_NVENC
ULONG *	pVideoEncoderFormat	OUT	Pointer to the video format
ULONG *	pVideoWidth	OUT	Pointer to the video width
ULONG *	pVideoHeight	OUT	Pointer to the video height
double *	pVideoFrameRate	OUT	Pointer to the video frame rate
ULONG *	pAudioEncoderFormat	OUT	Pointer to the audio format
ULONG *	pAudioChannels	OUT	Pointer to the total audio channels
ULONG *	pAudioBitsPerSample	OUT	Pointer to the audio bits per sample
ULONG *	pAudioSampleFrequency	OUT	Pointer to the audio sample frequency
double *	pTotalDurationTimes	OUT	Pointer to the total duration times
ULONG *	pTotalVideoFrames	OUT	Pointer to the total video frames
ULONG *	pTotalAudioFrames	OUT	Pointer to the audio frames
ULONG *	pTotalMetadataFrames	OUT	pointer to total meta-data frames available

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

*Example : Create a file transcoder object before start converting video file*

```
QCAP_CREATE_FILE_TRANSCODER( pSrcFileName,
                              &pFileTranscoder,
                              QCAP_DECODER_TYPE_SOFTWARE,
                              &nVideoFormat,
                              &nVideoWidth,
                              &nVideoHeight,
                              &nVideoFrameRate,
                              &nAudioFormat,
                              &nAudioChannels,
                              &nAudioBitsPerSample,
                              &nAudioSampleFrequency,
                              &nTotalDurationTimes,
                              &nTotalVideoFrames,
                              &nTotalAudioFrames );
```

## 20.2 QCAP\_START\_FILE\_TRANSCODER

Introduction

After a user has created a file transcoder with source video file assigned. The user can use this function to start file transcoder to another video format. It might take a long time to get transcoder job done and it depends on your video file size.

Parameters

type	parameter	I/O	descriptions
PVOID	pFileTranscoder	IN	Handle of the file transcoder object
CHAR *	pszDstFileName	IN	Specify the target video file name to output, supported file extensions: <b>AVI, MP4, ASF, WMV, FLV, TS, M3U8, SCF</b>

Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

*Example : Start to convert file transcoder object to MP4 format*

```
QCAP_START_FILE_TRANSCODER( pFileTranscoder, "CONVERT_RESULT.MP4" );
```

# 20.3 QCAP\_STOP\_FILE\_TRANSCODER

## Introduction

When a user starts a video file transcoder, it might take a while. If a user wants to stop the transcoding, a user can call this function to stop file converting process.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFileTranscoder	IN	Handle of the file transcoder object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To stop video file converting process*

```
QCAP_STOP_FILE_TRANSCODER( pFileTranscoder );
```

# 20.4 QCAP\_DESTROY\_FILE\_TRANSCODER

## Introduction

Call this function to destroy file transcoder object and release its system resource.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFileTranscoder	IN	Handle of the file transcoder object

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : To release the file transcoder resources*

```
QCAP_DESTROY_FILE_TRANSCODER( pFileTranscoder );
```

# 20.5 File Transcoding Property Function

## Introduction

Like the recording properties, there are many properties to set in video transcoding. Before you send a video to transcoding, you will want to apply properties/setting so that the transcoding functions will output the result you want. This section provides functions are for software encoder only.

### 20.5.1 QCAP\_SET\_VIDEO\_FILE\_TRANSCODER\_PROPERTY

### 20.5.2 QCAP\_SET\_VIDEO\_FILE\_TRANSCODER\_PROPERTY\_EX

## Introduction

The user can use this function to set file transcoder encoder parameters or even parameters from compression system such as Profile, Level, Entropy, Complexity, B-Frames and SceneCut...etc.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_VIDEO\_RECORD\_PROPERTY\_EX()**.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFileTranscoder	IN	Handle of the file transcoder object
ULONG	nEncoderType	IN	Specify transcoder encoder type: QCAP_ENCODER_TYPE_SOFTWARE QCAP_ENCODER_TYPE_HARDWARE QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK QCAP_ENCODER_TYPE_AMD_STREAM QCAP_ENCODER_TYPE_NVIDIA_CUDA QCAP_ENCODER_TYPE_NVIDIA_NVENC <a href="#">Note: QCAP_ENCODER_TYPE_HARDWARE transcoding is not supported</a>
ULONG	nEncoderFormat	IN	Specify transcoder encoder format: QCAP_ENCODER_FORMAT_MPEG2 QCAP_ENCODER_FORMAT_H264 QCAP_ENCODER_FORMAT_H264_3D QCAP_ENCODER_FORMAT_H264_VC QCAP_ENCODER_FORMAT_RAW QCAP_ENCODER_FORMAT_RAW_NATIVE QCAP_ENCODER_FORMAT_H265 QCAP_ENCODER_FORMAT_RAW_YUY2 QCAP_ENCODER_FORMAT_RAW_UYVY QCAP_ENCODER_FORMAT_RAW_YV12 QCAP_ENCODER_FORMAT_RAW_I420 QCAP_ENCODER_FORMAT_RAW_Y800

ULONG	nWidth	IN	Specify transcoder encoder width.
ULONG	nHeight	IN	Specify transcoder encoder height.
double	dFrameRate	IN	Specify transcoder encoder frame rate.
ULONG	nRecordProfile	IN	<b>default QCAP_RECORD_PROFILE_BASELINE</b> Specify recording profile: QCAP_RECORD_PROFILE_BASELINE QCAP_RECORD_PROFILE_MAIN QCAP_RECORD_PROFILE_HIGH QCAP_RECORD_PROFILE_CONSTRAINED_BASELINE QCAP_RECORD_PROFILE_CONSTRAINED_HIGH <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nRecordLevel	IN	<b>default QCAP_RECORD_LEVEL_41</b> Specify recording levels: QCAP_RECORD_LEVEL_1 QCAP_RECORD_LEVEL_1B QCAP_RECORD_LEVEL_11 QCAP_RECORD_LEVEL_12 QCAP_RECORD_LEVEL_13 QCAP_RECORD_LEVEL_2 QCAP_RECORD_LEVEL_21 QCAP_RECORD_LEVEL_22 QCAP_RECORD_LEVEL_3 QCAP_RECORD_LEVEL_31 QCAP_RECORD_LEVEL_32 QCAP_RECORD_LEVEL_4 QCAP_RECORD_LEVEL_41 QCAP_RECORD_LEVEL_42 QCAP_RECORD_LEVEL_50 QCAP_RECORD_LEVEL_51 QCAP_RECORD_LEVEL_52 QCAP_RECORD_LEVEL_60 QCAP_RECORD_LEVEL_61 QCAP_RECORD_LEVEL_62 <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nRecordEntropy	IN	<b>default QCAP_RECORD_ENTROPY_CAVLC</b> Specify recording entropy: QCAP_RECORD_ENTROPY_CAVLC QCAP_RECORD_ENTROPY_CABAC <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nRecordComplexity	IN	<b>default 0</b> Specify recording complexity: QCAP_RECORD_COMPLEXITY_0 (Best Speed) QCAP_RECORD_COMPLEXITY_1 QCAP_RECORD_COMPLEXITY_2 QCAP_RECORD_COMPLEXITY_3 QCAP_RECORD_COMPLEXITY_4 QCAP_RECORD_COMPLEXITY_5 QCAP_RECORD_COMPLEXITY_6 (Best Quality) <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>

ULONG	nRecordMode	IN	Specify transcoder encoder mode: QCAP_RECORD_MODE_VBR QCAP_RECORD_MODE_CBR QCAP_RECORD_MODE_ABR QCAP_RECORD_MODE_CQP
ULONG	nQuality	IN	Specify transcoder encoder quality, from 0-10000. It is used for <b>VBR</b> and <b>ABR</b> .
ULONG	nBitRate	IN	Specify transcoder encoder bit rate. It is used for <b>CBR</b> and <b>ABR</b> e.g. 12Mbps = 12 x 1024 x 1024 bps
ULONG	nGOP	IN	Specify transcoder encoder GOP size, from 0-255
ULONG	nBFrames	IN	<b>default 0</b> Specify transcoder encoder B-Frame <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL	bIsInterleaved	IN	<b>default FALSE</b> Enable/Disable the Interleaved <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nSlices	IN	<b>default 0</b> Specify transcoder encoder slices, default 0 <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nLayers	IN	<b>default 0</b> Specify transcoder encoder layers, default 0 <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nSceneCut	IN	<b>default 0</b> Specify transcoder encoder Scene Cut, recommended value is 40 set 0 to turned off this function <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL	bMultiThread	IN	<b>default TRUE</b> Enable/Disable the multi-threaded CPU loading balance support <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL	bMBBRC	IN	<b>default FALSE</b> Enable/Disable the mbbrc <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL	bExtBRC	IN	<b>default FALSE</b> Enable/Disable the extbrc <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nMinQP	IN	<b>default 0</b> Specify the value of x264 Minimum quantizer settings <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nMaxQP	IN	

			<b>default 0</b> Specify the value of x264 Maximum quantizer settings <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nVBVMaxRate	IN	<b>default 0</b> Specify the value that x264 fills the buffer at (up to) the max rate <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nVBVBufSize	IN	<b>default 0</b> Specify the size that x264 fills the buffer <b>Only in</b> <b>QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG	nAspectRatioX	IN	Specify the aspect ratio X axis set 0 to turned off
ULONG	nAspectRatioY	IN	Specify the aspect ratio Y axis set 0 to turned off

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.



## Examples

*Example : Set the properties before transcoding a video file*

```
QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY( pFileTranscoder,  
                                           QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,  
                                           QCAP_ENCODER_FORMAT_H264,  
                                           1280, 720,  
                                           60,  
                                           QCAP_RECORD_MODE_CBR,  
                                           8000,  
                                           12*1024*1024,  
                                           30,  
                                           4, 3 );  
  
QCAP_SET_VIDEO_FILE_TRANSCODER_PROPERTY_EX( pFileTranscoder,  
                                              QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK,  
                                              QCAP_ENCODER_FORMAT_H264,  
                                              1280, 720,  
                                              60,  
                                              nRecordProfile,  
                                              nRecordLevel,  
                                              nRecordEntropy,  
                                              QCAP_RECORD_COMPLEXITY_1,  
                                              QCAP_RECORD_MODE_CBR,  
                                              8000,  
                                              12*1024*1024,  
                                              30,  
                                              0, 0, 0, 0, 0,  
                                              FALSE,  
                                              0, 0,  
                                              0, 0, 0, 0,  
                                              4, 3 );
```

## 20.5.3 QCAP\_GET\_VIDEO\_FILE\_TRANSCODER\_PROPERTY

## 20.5.4 QCAP\_GET\_VIDEO\_FILE\_TRANSCODER\_PROPERTY\_EX

### Introduction

The user can use this function to get file transcoder encoder parameters or even parameters from compression system such as Profile, Level, Entropy Complexity, B-Frames, and SceneCut...etc.



For more detailed parameters descriptions, please refer to **QCAP\_SET\_VIDEO\_FILE\_TRANSCODER\_PROPERTY\_EX()**.

### Parameters

type	parameter	I/O	descriptions
PVOID	pFileTranscoder	IN	Handle of the file transcoder object
ULONG *	pEncoderType	OUT	Pointer to the current transcoder encoder type
ULONG *	pEncoderFormat	OUT	Pointer to the current transcoder encoder format
ULONG *	pWidth	OUT	Pointer to the current transcoder encoder width
ULONG *	pHeight	OUT	Pointer to the current transcoder encoder height
double *	pFrameRate	OUT	Pointer to the current transcoder encoder frame rate
ULONG *	pRecordProfile	OUT	Pointer to the current transcoder profile <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pRecordLevel	OUT	Pointer to the current transcoder level <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pRecordEntropy	OUT	Pointer to the current transcoder entropy <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pRecordComplexity	OUT	Pointer to the current transcoder complexity <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pRecordMode	OUT	Pointer to the current transcoder mode
ULONG *	pQuality	OUT	Pointer to the current transcoder quality
ULONG *	pBitRate	OUT	Pointer to the current transcoder bit rate
ULONG *	pGOP	OUT	Pointer to the current transcoder GOP size
ULONG *	pBFrames	OUT	Pointer to the current transcoder B-Frames <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL *	pIsInterleaved	OUT	Pointer to the current transcoder Interleaved <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pSlices	OUT	Pointer to the current transcoder slices <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pLayers	OUT	

			Pointer to the current transcoder layers <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pSceneCut	OUT	Pointer to the current transcoder screen cut <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL *	pMultiThread	OUT	Pointer to the current multi-threaded CPU loading balance status <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL *	pMBBRC	OUT	Pointer to the current mbbrc status <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
BOOL *	pExtBRC	OUT	Pointer to the current extbrc status <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pMinQP	OUT	Pointer to the value of x264 Minimum quantizer settings <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pMaxQP	OUT	Pointer to the value of x264 Maximum quantizer settings <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pVBVMaxRate	OUT	Pointer to the value that x264 fills the buffer at (up to) the max rate <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pVBVBufSize	OUT	Pointer to the size that x264 fills the buffer <b>Only in</b> <b>QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX()</b>
ULONG *	pAspectRatioX	OUT	Pointer to the aspect ratio X axis
ULONG *	pAspectRatioY	OUT	Pointer to the aspect ratio Y axis

#### Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

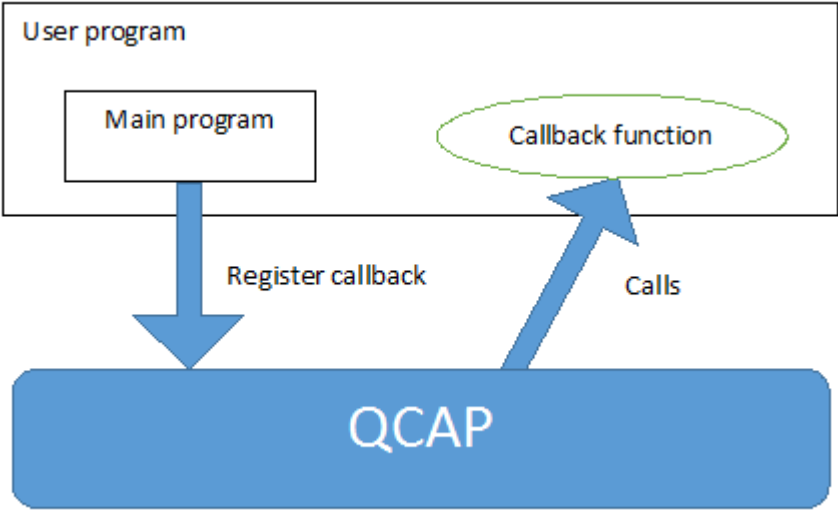
*Example : Retrieve the video transcoding properties*

```
QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY( pFileTranscoder,
                                           &nEncoderType,
                                           &nEncoderFormat,
                                           &nWidth, &nHeight,
                                           &nFrameRate,
                                           &nRecordMode,
                                           &nQuality,
                                           &nBitRate,
                                           &nGOP,
                                           &pAspectRatioX,
                                           &npAspectRatioY );

QCAP_GET_VIDEO_FILE_TRANSCODER_PROPERTY_EX( pFileTranscoder,
                                              &nEncoderType,
                                              &nEncoderFormat,
                                              &nWidth, &nHeight,
                                              &nFrameRate,
                                              &nRecordProfile,
                                              &nRecordLevel,
                                              &nRecordEntropy,
                                              &nRecordComplexity,
                                              &nRecordMode,
                                              &nQuality,
                                              &nBitRate,
                                              &nGOP,
                                              &nBframe,
                                              &nIsInterleaved,
                                              &nSlices,
                                              &nLayers,
                                              &nSceneCut,
                                              &nMultiThread,
                                              &nMBBRC, &nExtBRC,
                                              &nMinQP, &nMaxQP,
                                              &nVBVMaxRate, &nVBVBufSize,
                                              &nAspectRatioX, &nAspectRatioY );
```

# 20.6 File Transcoding Callback Function

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_FILE_TRANSCODER_CALLBACK	PF_FILE_TRANSCODER_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

# 20.6.1 QCAP\_REGISTER\_FILE\_TRANSCODER\_CALLBACK

## Introduction

When a video is converting to another format, this callback function provides the file transcoder progress in percentage.

## Parameters

type	parameter	I/O	descriptions
PVOID	pFileTranscoder	IN	Handle of the file transcoder object
<b>PF_FILE_TRANSCODER_CALLBACK</b>	pCB	IN	Callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## PF\_FILE\_TRANSCODER\_CALLBACK

### Parameters of Callback

type	parameter	callback descriptions
PVOID	pFileTranscoder	Handle of the file transcoder object
double	dPercentageCompleted	Specify the percentage of transcoding progress
PVOID	pUserData	Pointer to custom user data

### Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.

## Examples

*Example : Register a callback function to get video transcoding progress*

```
QRETURN file_transcode_callback( PVOID pFileTranscoder,
                                double dPercentageCompleted,
                                PVOID pUserData );

{
    return QCAP_RT_OK;
}

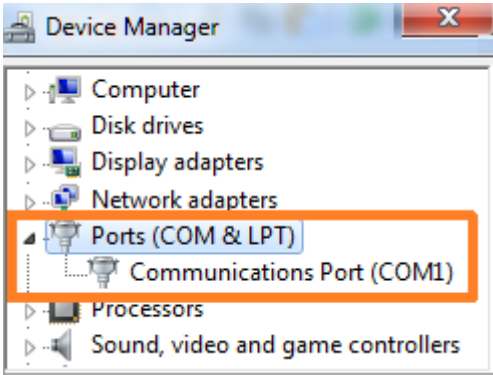
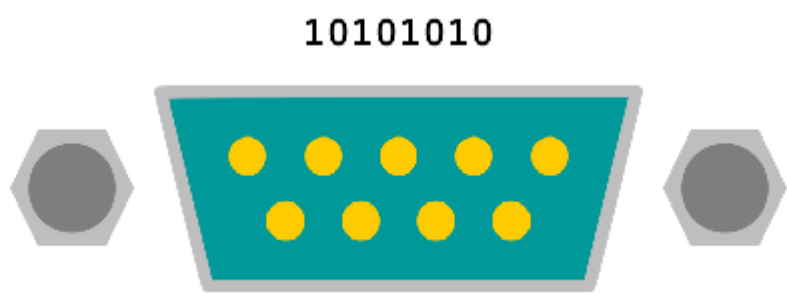
void test_callback()
{
    PF_FILE_TRANSCODER_CALLBACK pCB = file_transcode_callback;

    QCAP_REGISTER_FILE_TRANSCODER_CALLBACK( pFileTranscoder, pCB, pUserData );
}
```

# 21 Serial Port Function API

## Introduction

When user needs to communicate with an external device connected on a serial port device, this chapter provides a set of serial port functions that help a user to control a communications port with RS232/RS422/RS485.



# 21.1 QCAP\_CREATE\_SERIAL\_PORT

## Introduction

This function can let user opens a communications port. There are two I/O mode to create the serial port function to open the communications port: Asynchronous and Synchronous. The port number specify the port assigned from the system device, for example, the port number of "COM9" is 9.

## Parameters

type	parameter	I/O	descriptions
UINT	iPortNum	IN	Specify the serial port number
PVOID *	ppPort	OUT	Pointer to the Handle to the serial port
ULONG	nBaudRate	IN	<b>default 9600</b> Specify the serial baud-rate: 9600, 19200, 38400, 57600 and 115200 bits per second.
ULONG	nDataBits	IN	<b>default 8</b> Specify the serial data bits per character
ULONG	nParityCheck	IN	<b>default QCAP_SERIAL_PORT_PARITY_CHECK_NONE</b> Specify the serial parity error detection: QCAP_SERIAL_PORT_PARITY_CHECK_NONE QCAP_SERIAL_PORT_PARITY_CHECK_ODD QCAP_SERIAL_PORT_PARITY_CHECK_EVEN QCAP_SERIAL_PORT_PARITY_CHECK_MARK QCAP_SERIAL_PORT_PARITY_CHECK_SPACE
ULONG	nStopBits	IN	<b>default QCAP_SERIAL_PORT_STOP_BITS_ONE</b> Specify the serial stop bits: QCAP_SERIAL_PORT_STOP_BITS_ONE QCAP_SERIAL_PORT_STOP_BITS_ONE_POINT_FIVE QCAP_SERIAL_PORT_STOP_BITS_TWO
ULONG	nFlowControl	IN	<b>default QCAP_SERIAL_PORT_FLOW_CONTROL_NONE</b> Specify the serial flow control: QCAP_SERIAL_PORT_FLOW_CONTROL_NONE QCAP_SERIAL_PORT_FLOW_CONTROL_CTS_RTS QCAP_SERIAL_PORT_FLOW_CONTROL_CTS_DTR QCAP_SERIAL_PORT_FLOW_CONTROL_DSR_RTS QCAP_SERIAL_PORT_FLOW_CONTROL_DSR_DTR QCAP_SERIAL_PORT_FLOW_CONTROL_XON_XOFF
BOOL	bAsynchronousIO	IN	<b>default FALSE</b> Set the asynchronous operation flag

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.



## Examples

*Example : Open a serial port to transfer data*

```
VOID *pPort = NULL;

QRETURN serial_port_received_data( PVOID pPort,
                                   UINT iPortNum,
                                   BYTE * pDataBuffer,
                                   ULONG nDataBufferLen,
                                   PVOID pDataUser );

{
    //receive data from serial port
    return QCAP_RT_OK;
}

QCAP_CREATE_SERIAL_PORT( 9,    //port number
                        &pPort, //get serial port handle
                        115200, //baud rate
                        8,      //data bits
                        QCAP_SERIAL_PORT_PARITY_CHECK_NONE, //parity
                        QCAP_SERIAL_PORT_STOP_BITS_ONE,    //stop bits
                        QCAP_SERIAL_PORT_FLOW_CONTROL_NONE, //flow control
                        FALSE ); //asynchronous I/O

//register callback

PF_SERIAL_PORT_RECEIVED_DATA_CALLBACK pCB = serial_port_received_data;

QCAP_REGISTER_SERIAL_PORT_RECEIVED_DATA_CALLBACK( pPort, pCB, pDataUser );

QCAP_START_SERIAL_PORT( pPort );

//transmit data
QCAP_SEND_SERIAL_PORT_SIGNAL( pPort, QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_XOFF );

BYTE *pDataBuffer="This is a TEST data";

//release the serial port
QCAP_STOP_SERIAL_PORT( pPort );

QCAP_DESTROY_SERIAL_PORT( pPort );
```

# 21.2 QCAP\_START\_SERIAL\_PORT

## Introduction

This function will start the serial port device, and then a user can transfer/receive data.

## Parameters

type	parameter	I/O	descriptions
PVOID	pPort	IN	Handle to the serial port

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Start the serial port before transfer data*

```
QCAP_START_SERIAL_PORT( pPort );
```

# 21.3 QCAP\_STOP\_SERIAL\_PORT

## Introduction

This function will stop the serial port device, then a user can no longer transfer/receive data.

## Parameters

type	parameter	I/O	descriptions
PVOID	pPort	IN	Handle to the serial port

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Stop the serial port*

```
QCAP_STOP_SERIAL_PORT( pPort );
```

# 21.4 QCAP\_DESTROY\_SERIAL\_PORT

## Introduction

When the serial port is no longer needed, use can call this function to release the system resource.

## Parameters

type	parameter	I/O	descriptions
PVOID	pPort	IN	Handle to the serial port

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Destroy a serial port*

```
QCAP_DESTROY_SERIAL_PORT( pPort );
```

# 21.5 QCAP\_SEND\_SERIAL\_PORT\_SIGNAL

## Introduction

When the serial port is ready to transmit/receive data, a user can use this function to send flow control signal to a serial port.

## Parameters

type	parameter	I/O	descriptions
PVOID	pPort	IN	Handle to the serial port
ULONG	nSignalType	IN	Specify the serial port signal type: QCAP_SERIAL_PORT_SIGNAL_TYPE_CLEAR_DTR QCAP_SERIAL_PORT_SIGNAL_TYPE_CLEAR_RTS QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_DTR QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_RTS QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_XOFF QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_XON QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_BREAK QCAP_SERIAL_PORT_SIGNAL_TYPE_CLEAR_BREAK

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Send the flow control signal XOFF to serial port*

```
QCAP_SEND_SERIAL_PORT_SIGNAL( pPort, QCAP_SERIAL_PORT_SIGNAL_TYPE_SET_XOFF );
```

# 21.6 QCAP\_SEND\_SERIAL\_PORT\_DATA

## Introduction

This function can transmit a block of data to the serial port.

## Parameters

type	parameter	I/O	descriptions
PVOID	pPort	IN	Handle to the serial port
BYTE *	pDataBuffer	IN	Pointer to the data input buffer
ULONG	nDataBufferLen	IN	Indicates the data input buffer size

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

*Example : Send a test string to serial port*

```
BYTE *pDataBuffer="This is a TEST data";

QCAP_SEND_SERIAL_PORT_DATA( pPort, pDataBuffer, strlen(pDataBuffer) );
```

# 21.7 QCAP\_SERIAL\_PORT\_ENUMERATION

## Introduction

This function can enumerate the serial ports available on the system.

## Parameters

type	parameter	I/O	descriptions
UINT *	pAvailablePortNum	OUT	The number of total serial ports
BOOL	bNext	IN	<b>default FALSE</b> The flag to enumerate next port or not

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

## Examples

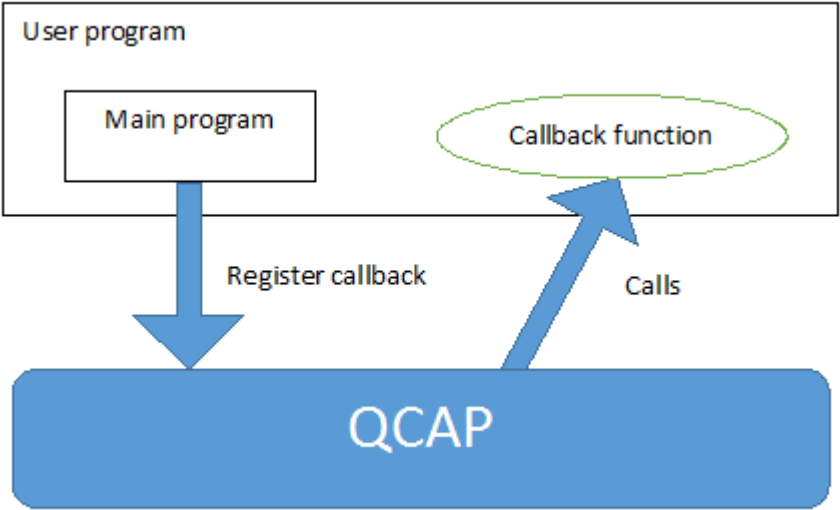
*Example : To enumerate the serial ports*

```
QCAP_SERIAL_PORT_ENUMERATION( &pAvailablePortNum, false );
```

C

# 21.8 Serial Port Callback Functions

## Introduction



The callback function is a pointer to a user defined function and will be called by the QCAP library. There are many callback functions (and its register functions) for different purposes:

This section contains how to register channel record callback functions for:

Register functions	Callback functions
QCAP_REGISTER_SERIAL_PORT_RECEIVED_DATA_CALLBACK	PF_SERIAL_PORT_RECEIVED_DATA_CALLBACK



If the callback function passes the buffer pointer in NULL, and a buffer length is 0, indicates there is no source anymore.

# 21.8.1 QCAP\_REGISTER\_SERIAL\_PORT\_RECEIVED\_DATA\_CALLBACK

## Introduction

The user can register a *PF\_SERIAL\_PORT\_RECEIVED\_DATA\_CALLBACK* function to get notification when the there is data coming in from the serial port. This callback function needs to be registered before serial port starts to operate.

## Parameters

type	parameter	I/O	descriptions
PVOID	pPort	IN	Handle to the serial port
<i>PF_SERIAL_PORT_RECEIVED_DATA_CALLBACK</i>	pCB	IN	callback function
PVOID	pUserData	IN	Pointer to custom user data

## Return value

Returns **QCAP\_RS\_SUCCESSFUL** if OK, otherwise an error occurred.

# *PF\_SERIAL\_PORT\_RECEIVED\_DATA\_CALLBACK*

## Parameters of Callback

type	parameter	callback descriptions
PVOID	pPort	Handle to the serial port
UINT	iPortNum	Specify the serial port number
BYTE *	pDataBuffer	Pointer to the input source buffer
ULONG	nDataBufferLen	Specify the input source buffer size
PVOID	pUserData	Pointer to custom user data

## Return value of Callback

Returns **QCAP\_RT\_OK** or **QCAP\_RT\_FAIL** after callback processed.



## Examples

*Example : Register callback to receive data from serial port*

```
QRETURN serial_port_received_data( PVOID pPort,
                                   UINT iPortNum,
                                   BYTE * pDataBuffer,
                                   ULONG nDataBufferLen,
                                   PVOID pUserData );

{
    return QCAP_RT_OK;
}

void test_callback()
{
    PF_SERIAL_PORT_RECEIVED_DATA_CALLBACK pCB = serial_port_received_data;

    QCAP_REGISTER_SERIAL_PORT_RECEIVED_DATA_CALLBACK( pPort, pCB, pUserData );
}
```