QCAP SDK
1.1.0.123.7, 2013.08.12

1 SDK Introduction 16

1.1 Overview 16

1.2 Device Workflow in Software View 18

1.3 Programming Language Supported 19

1.4 Sample Code 19

1.5 Header Files and Link Library 19

1.6 SDK Package Organization 20

1.7 Platform and Thread Safety 20

2 System Function API 21

2.1 QCAP_SET_SYSTEM_CONFIGURATION 21

2.2 QCAP_QUERY_ENCODER_TYPE_CAP 23

3 Device Function API 25

3.1 QCAP_DEVICE_ENUMERATION 25

3.2 QCAP_CREATE 30

3.3 QCAP_DESTROY 33
3.4 QCAP_RUN 34
3.5 QCAP_STOP 35
3.6 QCAP_GET_VIDEO_INPUT 36
3.7 QCAP_SET_VIDEO_INPUT 37
3.8 QCAP_GET_AUDIO_INPUT 39
3.9 QCAP_SET_AUDIO_INPUT 40
3.10 QCAP_GET_VIDEO_STANDARD 42
3.11 QCAP_SET_VIDEO_STANDARD 44
3.12 QCAP_GET_VIDEO_CURRENT_INPUT_FORMAT 46
3.13 QCAP_GET_AUDIO_CURRENT_INPUT_FORMAT 48
3.14 QCAP_GET_VIDEO_DEFAULT_OUTPUT_FORMAT 50
3.15 QCAP_SET_VIDEO_DEFAULT_OUTPUT_FORMAT 52
3.16 QCAP_GET_AUDIO_DEFAULT_OUTPUT_FORMAT 54
3.17 QCAP_SET_AUDIO_DEFAULT_OUTPUT_FORMAT 56
3.18 QCAP_GET_VIDEO_DEINTERLACE_TYPE 58
3.19 QCAP_SET_VIDEO_DEINTERLACE_TYPE 59

3.20 QCAP_GET_VIDEO_DEINTERLACE 61
3.21 QCAP_SET_VIDEO_DEINTERLACE 62
3.22 QCAP_GET_VIDEO_BRIGHTNESS 64
3.23 QCAP_SET_VIDEO_BRIGHTNESS 65
3.24 QCAP_GET_VIDEO_CONTRAST 66
3.25 QCAP_SET_VIDEO_CONTRAST 67
3.26 QCAP_GET_VIDEO_HUE 68
3.27 QCAP_SET_VIDEO_HUE 69
3.28 QCAP_GET_VIDEO_SATURATION 70
3.29 QCAP_SET_VIDEO_SATURATION 71
3.30 QCAP_GET_VIDEO_SHARPNESS 72
3.31 QCAP_SET_VIDEO_SHARPNESS 73
3.32 QCAP_GET_AUDIO_VOLUME 74
3.33 QCAP_SET_AUDIO_VOLUME 75
3.34 QCAP_GET_DEVICE_CUSTOM_PROPERTY 76
3.35 QCAP_SET_DEVICE_CUSTOM_PROPERTY 78
3.36 QCAP_GET_DEVICE_CUSTOM_PROPERTY_EX 80

3.37 QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX 82

4 Snapshot Function API 84

4.1 QCAP_SNAPSHOT_BMP 84

4.2 QCAP_SNAPSHOT_JPG 86

4.3 QCAP_SNAPSHOT_BMP_EX 88

4.4 QCAP_SNAPSHOT_JPG_EX 91

5 Recording Function API 94

5.1 QCAP_GET_VIDEO_RECORD_PROPERTY 94

5.2 QCAP_SET_VIDEO_RECORD_PROPERTY 97

5.3 QCAP_GET_VIDEO_RECORD_PROPERTY_EX (CROPPING) 101

5.4 QCAP_SET_VIDEO_RECORD_PROPERTY_EX (CROPPING) 105

5.5 QCAP_GET_AUDIO_RECORD_PROPERTY 110

5.6 QCAP_SET_AUDIO_RECORD_PROPERTY 112

5.7 QCAP_START_RECORD 114

5.8 QCAP_STOP_RECORD 116

5.9 QCAP_RESUME_RECORD 118

5.10 QCAP_PAUSE_RECORD 119

6 Synchronized Recording Function API 120

6.1 QCAP_START_SYNCHRONIZED_RECORD 120

6.2 QCAP_STOP_SYNCHRONIZED_RECORD 122

6.3 QCAP_RESUME_SYNCHRONIZED_RECORD 125

6.4 QCAP_PAUSE_SYNCHRONIZED_RECORD 127

7 OSD Function API 129

7.1 QCAP_SET_OSD_TEXT 129

7.2 QCAP_SET_OSD_PICTURE 133

7.3 QCAP_SET_OSD_BUFFER 135

8 Clone Function API 139

8.1 QCAP_CREATE_CLONE 139

9 Share Recording Function API 143

9.1 Overview 143

9.2 QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY 144

9.3 QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY 147
9.4 QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX 151
9.5 QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX 155
9.6 QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY 161
9.7 QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY 163
9.8 QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER 166
9.9 QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER 169
9.10 QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER 171
9.11 QCAP_SET_AUDIO_SHARE_RECORD_COMPRESSION_BUFFER 173
9.12 QCAP_START_SHARE_RECORD 175
9.13 QCAP_STOP_SHARE_RECORD 178
9.14 QCAP_RESUME_SHARE_RECORD 180
9.15 QCAP_PAUSE_SHARE_RECORD 181
9.16 QCAP_SET_OSD_SHARE_RECORD_TEXT 182
9.17 QCAP_SET_OSD_SHARE_RECORD_PICTURE 185
9.18 QCAP_SET_OSD_SHARE_RECORD_BUFFER 188

10 File Playback/Editing Function API 192

10.1 QCAP_OPEN_FILE 192
10.2 QCAP_DESTROY_FILE 197
10.3 QCAP_PLAY_FILE 198
10.4 QCAP_PAUSE_FILE 199
10.5 QCAP_STOP_FILE 200
10.6 QCAP_GET_FILE_POSITION 201
10.7 QCAP_SET_FILE_POSITION 202
10.8 QCAP_GET_FILE_PLAYBACK_SPEED 203
10.9 QCAP_SET_FILE_PLAYBACK_SPEED 204
10.10 QCAP_GET_FILE_AUDIO_VOLUME 205
10.11 QCAP_SET_FILE_AUDIO_VOLUME 206
10.12 QCAP_SET_OSD_FILE_TEXT 207
10.13 QCAP_SET_OSD_FILE_PICTURE 210
10.14 QCAP_SET_OSD_FILE_BUFFER 213
10.15 QCAP_GET_FILE_VIDEO_STREAM_BUFFER 216
10.16 QCAP_GET_FILE_AUDIO_STREAM_BUFFER 218
10.17 QCAP_EXPORT_FILE 220

10.18 QCAP_MERGE_FILE 222

11 Broadcasting Server & Client Function API 224

11.1 Overview 224

11.2 QCAP_CREATE_BROADCAST_RTSP_SERVER 227

11.3 QCAP_CREATE_BROADCAST_RTMP_SERVER 229

11.4 QCAP_CREATE_BROADCAST_RTMP_WEB_PORTAL_SERVER 231

11.5 QCAP_CREATE_BROADCAST_HLS_SERVER 233

11.6 QCAP_DESTROY_BROADCAST_SERVER 236

11.7 QCAP_START_BROADCAST_SERVER 237

11.8 QCAP_STOP_BROADCAST_SERVER 238

11.9 QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY 239

11.10 QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY 242

11.11 QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX 246

11.12 QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX 250

11.13 QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY 257

11.14 QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY 259

11.15 QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER 261
11.16 QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER 266
11.17 QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER 268
11.18 QCAP_SET_AUDIO_BROADCAST_SERVER_COMPRESSION_BUFFER 271
11.19 QCAP_SET_OSD_BROADCAST_SERVER_TEXT 273
11.20 QCAP_SET_OSD_BROADCAST_SERVER_PICTURE 276
11.21 QCAP_SET_OSD_BROADCAST_SERVER_BUFFER 279
11.22 QCAP_CREATE_BROADCAST_CLIENT 283
11.23 QCAP_DESTROY_BROADCAST_CLIENT 285
11.24 QCAP_START_BROADCAST_CLIENT 286
11.25 QCAP_STOP_BROADCAST_CLIENT 288
11.26 QCAP_GET_BROADCAST_CLIENT_AUDIO_VOLUME 289
11.27 QCAP_SET_BROADCAST_CLIENT_AUDIO_VOLUME 290
11.28 QCAP_START_BROADCAST_CLIENT_RECORD 291
11.29 QCAP_STOP_BROADCAST_CLIENT_RECORD 293
11.30 QCAP_RESUME_BROADCAST_CLIENT_RECORD 295
11.31 QCAP_PAUSE_BROADCAST_CLIENT_RECORD 296

11.32 QCAP_SET_OSD_BROADCAST_CLIENT_TEXT 297
11.33 QCAP_SET_OSD_BROADCAST_CLIENT_PICTURE 300
11.34 QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER 303
11.35 An Broadcasting Software Programming Guides 307
12 Animation Function API 313
12.1 QCAP_CREATE_ANIMATION_CLIP 313
12.2 QCAP_DESTROY_ANIMATION_CLIP 315
12.3 QCAP_START_ANIMATION_CLIP 317
12.4 QCAP_STOP_ANIMATION_CLIP 318
12.5 QCAP_STEP_ANIMATION_CLIP 319
12.6 QCAP_SEEK_ANIMATION_CLIP 321
12.7 QCAP_SET_ANIMATION_CLIP_SPRITE_PICTURE 323
12.8 QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER 325
12.9 QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX 327
12.10 QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE 330
12.11 QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX 332

12.12 An Animation Clip Software Programming Guides 334

13 Virtual Camera Function API 336

13.1 QCAP_CREATE_VIRTUAL_CAMERA 336

13.2 QCAP_DESTROY_VIRTUAL_CAMERA 339

13.3 QCAP_GET_VIDEO_VIRTUAL_CAMERA_PROPERTY 340

13.4 QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY 342

13.5 QCAP_GET_AUDIO_VIRTUAL_CAMERA_PROPERTY 344

13.6 QCAP_SET_AUDIO_VIRTUAL_CAMERA_PROPERTY 346

13.7 QCAP_SET_VIDEO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER 348

13.8 QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER 351

13.9 QCAP_START_VIRTUAL_CAMERA 353

13.10 QCAP_STOP_VIRTUAL_CAMERA 354

14 Callback Function API 355

14.1 QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK 355

14.2 QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK 357

14.3 QCAP_REGISTER_FORMAT_CHANGED_CALLBACK 359

14.4 QCAP_REGISTER_SNAPSHOT_DONE_CALLBACK 361
14.5 QCAP_REGISTER_RECORD_DONE_CALLBACK 363
14.6 QCAP_REGISTER_SHARE_RECORD_DONE_CALLBACK 365
14.7 QCAP_REGISTER_BROADCAST_CLIENT_RECORD_DONE_CALLBACK 367
14.8 QCAP_REGISTER_BROADCAST_CLIENT_CONNECTED_CALLBACK 369
14.9 QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK 372
14.10 QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK 376
14.11 QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK 379
14.12 QCAP_REGISTER_VIDEO_RECORD_CALLBACK 381
14.13 QCAP_REGISTER_AUDIO_RECORD_CALLBACK 383
14.14 QCAP_REGISTER_VIDEO_SHARE_RECORD_CALLBACK 385
14.15 QCAP_REGISTER_AUDIO_SHARE_RECORD_CALLBACK 387
14.16 QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK 389
14.17 QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK 391
14.18 QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK 393
14.19 QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK 395
14.20 QCAP_REGISTER_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK 397

14.21 QCAP_REGISTER_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK 399
14.22 QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK 401
14.23 QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK 404
14.24 QCAP_REGISTER_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK 407
14.25 QCAP_REGISTER_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK 409
14.26 QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK 411
14.27 QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK 413

15 Lock/Unlock Functions API 415

15.1 QCAP_LOCK_VIDEO_PREVIEW_BUFFER 415
15.2 QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER 417
15.3 QCAP_LOCK_AUDIO_PREVIEW_BUFFER 418
15.4 QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER 420

16 Camera Function API 421

16.1 QCAP_CAMERA_ENUMERATION 421
16.2 QCAP_SET_DEFAULT_CAMERA 423

17 Sound Card Function API 425

17.1 QCAP_SOUNDCARD_ENUMERATION 425

17.2 QCAP_SET_DEFAULT_SOUNDCARD 427

1 SDK Introduction

1.1 Overview

The purpose of QCAP is different from original AMESDK library. It offers more high-level APIs for developer to easy access our capture card and to develop rich application. It is developed based on AMESDK's kernel and supports following features:

- Video and Audio Capture and Preview
- Video and Audio Clone Preview on Several Attached Windows
- Various Deinterlace Methods
- Alpha Blending OSD function for Text, Picture, and Buffer Objects
- Snapshot to BMP or JPG Format (Continuous, Scaling, and Cropping)
- Multiple Streams Channel Recording to AVI, MP4, ASF, FLV and TS (Scaling & Cropping)
- Multiple Streams Share Recording to AVI, MP4, ASF, FLV and TS (Scaling & Cropping)
- File Playback and Editing Library (Export, Merge, and Insert) *New*
- Multiple Streams Broadcasting RTSP Server for UDP/TCP/HTTP & RTSP Client

About AAC License

This software development kit contains software programming code and libraries related to AAC function that is subject to your development use only. All the intellectual properties are held by individual license groups and/or the property holders. License and royalty fee should be charged by the license groups and/or property holders and paid by you if any of them is implemented into your commercial product. We will not be responsible for any illegal usage and/or any infringement of the rights of the individual owners of these intellectual properties.
Details of AAC could be found in the website:
http://www.vialicensing.com/licensing/aac-overview.aspx

- Multiple Streams Broadcasting RTMP Server and RTMP Web Portal
- Multiple Streams Broadcasting HLS Server
- Transition Effect Animation Engine (Action Script File, Spline Transform, Alpha Blending, and Up/Down Scaling)
- Multiple Channels Synchronized Recording
- Virtual Camera *New*
- Callback Functions Registration
- Add-on Sound Card and USB Camera Capture Supported *New*
- Multiple Cards Supported

1.2 Device Workflow in Software View

Programming Step	Related API Functions
CoInitialize	
Create Device	QCAP_CREATE
Set Device Parameters	QCAP_SET_VIDEO_INPUT QCAP_SET_VIDEO_DEINTERLACE QCAP_SET_VIDEO_BRIGHTNESS, … QCAP_SET_AUDIO_VOLUME
Start Capturing	QCAP_RUN
Snapshot	QCAP_SNAPSHOT_BMP QCAP_SNAPSHOT_JPG
Start Recording	QCAP_SET_VIDEO_RECORD_PROPERTY QCAP_SET_AUDIO_RECORD_PROPERTY QCAP_START_RECORD
Stop Recording	QCAP_STOP_RECORD
Stop Capturing	QCAP_STOP
Delete Device	QCAP_DESTROY
CouninitializeCouninitialize	

1.3 Programming Language Supported

The API supports VC, VB, C#, Delphi and LabView development environments.

In this document, only how to use APIs in VC environment is illustrated.

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

For LabView developer, please add _QCAP_LABVIEW_IMPORT macro into the predefinition page during the library importing.

1.4 Sample Code

VC, VB, C#, Delphi and LabView sample code and sample program are available in SDK package.

1.5 Header Files and Link Library

VC, LabView	QCAP.H, QCAP.LIB, QCAP.DLL, AMESDK.DLL
C#, VB, Delphi	QCAP.H, QCAP.LIB, QCAP.DLL, QCAP.NET.DLL, AMESDK.DLL

Developer should find out 4 lib folders from SDK packet. If developer wants to access Intel hardware H.264 encoder (Intel Media SDK), you need use VC.GPU or NET.GPU library. Note!! Intel Media SDK is only support on Windows 7 system.

1.6 SDK Package Organization

DOC	SDK documents for capture devices
INC	Included files
LIB	Libraries
SAMPLES	Sample code and sample program

1.7 Platform and Thread Safety

Windows XP, Windows Vista, Windows 7 and Windows 8 are all supported by this SDK.

API calling can be used in the multiple process and the multiple threads.

2 System Function API

2.1 QCAP_SET_SYSTEM_CONFIGURATION

```
QRESULT QCAP_SET_SYSTEM_CONFIGURATION(  BOOL   bEnableMultipleUsersAccess = TRUE,
                                        BOOL   bEnableVideoPreviewDevice = TRUE,
                                        BOOL   bEnableAudioPreviewDevice = TRUE,
                                        BOOL   bEnableVideoHardwareMainEncoderDevice = TRUE,
                                        BOOL   bEnableVideoHardwareSubEncoderDevice = TRUE,
                                        ULONG  nAutoInputDetectionTimeout = 3000 )
```

Parameter

Name	Description
bEnableMultipleUsersAccess	**IN**, Enable/Disable the multiple user access
bEnableVideoPreviewDevice	**IN**, Enable/Disable the hardware video preview device
bEnableAudioPreviewDevice	**IN**, Enable/Disable the hardware audio preview device
bEnableHardwareMainEncoder	**IN**, Enable/Disable the hardware video main encoder device
bEnableHardwareSubEncoder	**IN**, Enable/Disable the hardware video sub encoder device
nAutoInputDetectionTimeout	**IN**, Set auto input detection timeout, unit is msec

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set some global parameters of the SDK, the function could be modified in future.

Example

```
QCAP_SET_SYSTEM_CONFIGURATION( TRUE, TRUE, TRUE, TRUE, TRUE, 3000 );
```

2.2 QCAP_QUERY_ENCODER_TYPE_CAP

```
QRESULT QCAP_QUERY_ENCODER_TYPE_CAP( ULONG nEncoderType )
```

Parameter

Name	Description

nEncoderType	**IN**, Specify query encoder type. Available values are
	`QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK`
	`QCAP_ENCODER_TYPE_NVIDIA_CUDA`

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_NON_SUPPORT	Fail due to encoder type non support
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set query encoder type, if exist in the system before you use GPU Encoder, we suggest you to use the function to check if GPU Encoder is enabled on the system.

Example

```
QCAP_QUERY_ENCODER_TYPE_CAP( QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK );
```

3 Device Function API

3.1 QCAP_DEVICE_ENUMERATION

```
QRESULT QCAP_DEVICE_ENUMERATION( ULONGLONG  * *    ppVideoDevicelist,
                                 ULONG *           pVideoDeviceSize,
                                 ULONGLONG * *     ppVideoEncoderDevicelist,
                                 ULONG *           pVideoEncoderDeviceSize,
                                 ULONGLONG * *     ppAudioDevicelist,
                                 ULONG *           pAudioDeviceSize,
                                 ULONGLONG * *     ppAudioEncoderDeviceList,
                                 ULONG *           pAudioEncoderDeviceSize,
                                 ULONG             nDeviceEnumType = QCAP_ENUM_TYPE_SERIAL_NUMBER )
```

Parameter

Name	Description
ppVideoDevicelist	**OUT**, Pointer to the address of all video capture devices
pVideoDeviceSize	**OUT**, Pointer to a variable that stores the quantity of video capture device
ppVideoEncoderDevicelist	**OUT**, Pointer to the address of all video encoder device
pVideoEncoderDeviceSize	**OUT**, Pointer to a variable that stores the quantity of video encoder device
ppAudioDeviceList	**OUT**, Pointer to the address of all audio capture devices
pAudioDeviceSize	**OUT**, Pointer to a variable that stores the quantity of audio capture device
ppAudioEncoderDeviceList	**OUT**, Pointer to the address of all audio encoder devices
pAudioEncoderDeviceSize	**OUT**, Pointer to the address of all audio encoder devices

nDeviceEnumType	**IN**, Specify input device enumerate type. Available values are
	`QCAP_ENUM_TYPE_DEVICE_NAME`
	`QCAP_ENUM_TYPE_SERIAL_NUMBER`

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

If user select QCAP_ENUM_TYPE_SERIAL_NUMBER, this function is used to enumerate all capture device on the platform.

Here, every item of the returned list uses 8 bytes to describe the devices information.

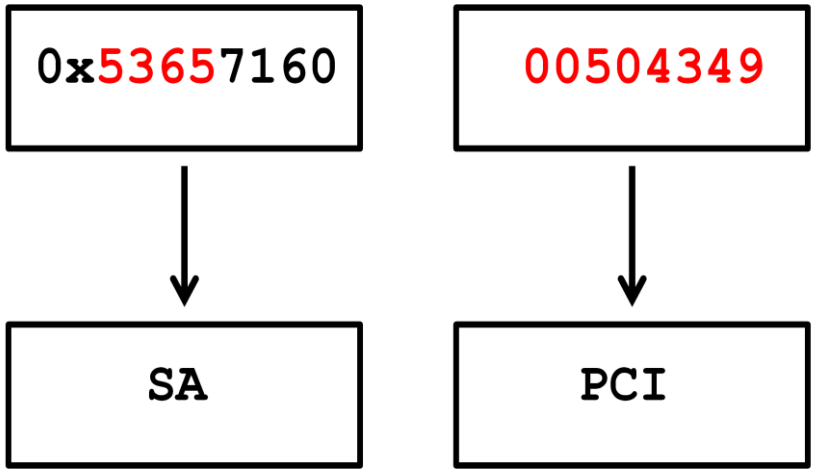
The lower 4 bytes are the devices serial number. The upper 4 bytes are device's number.

The upper 4 bytes are drivers product id. For instance, SC300N8 could return the 0x0000680212ABF308, from this function.

If user select QCAP_ENUM_TYPE_DEVICE_NAME, this function is used to list the names of all capture devices currently known to the user on the platform.

Although every item of the returned list of the latter is like the above mentioned brief explanation, nevertheless, the returned list it returns is to uses 8 bytes to describe the name of capture device currently known on the platform to the user.

For example, SC510N1 will return the value, 0x5365716000504349, from this function. And then we have to uses the ASCII character that corresponds to that ASCII value, so that the return value actually stands for or mean the device's name "SA7160 PCI".



Example

```
ULONGLONG * p_video_device_list = NULL;

ULONGLONG * p_video_encoder_device_list = NULL;

ULONGLONG * p_audio_device_list = NULL;

ULONGLONG * p_audio_encoder_device_list = NULL;

ULONG       n_video_device_list_size = 0;

ULONG       n_video_encoder_device_list_size = 0;

ULONG       n_audio_device_list_size = 0;

ULONG       n_audio_encoder_device_list_size = 0;
```

```
QCAP_DEVICE_ENUMERATION( &p_video_device_list, &n_video_device_list_size,
                         &p_video_encoder_device_list, &n_video_encoder_device_list_size,
                         &p_audio_device_list, &n_audio_device_list_size,
                         &p_audio_encoder_device_list, &n_audio_encoder_device_list_size,
                         QCAP_ENUM_TYPE_SERIAL_NUMBER );

for( ULONG i = 0 ; i < n_video_device_list_size ; i++ ) {

    sprintf( "Live#%02d = 0x%llX", i, p_video_device_list[ i ] );
}
for( ULONG i = 0 ; i < n_video_encoder_device_list_size ; i++ ) {

    sprintf( "Encoder#%02d = 0x%llX", i, p_video_encoder_device_list[ i ] );
}
```

3.2 QCAP_CREATE

```
QRESULT QCAP_CREATE( CHAR *    pszDevName,
                     UINT      iDevNum,
                     HWND      hAttachedWindow,
                     PVOID *   ppDevice,
                     BOOL      bThumbDraw = FALSE
                     BOOL      bMaintainAspectRatio = FALSE )
```

Parameter

Name	Description
pszDevName	**IN**, Device Name, Available values are "DC1150 USB" is for PD652 and PD652.3D "UB658G USB" is for UB658G "CY3014 USB" is for UB530 and UB530G "TW6802 PCI" is for SC200, SC300 and 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 "CAMERA" is for USB CAMERA *New* "SOUNDCARD" is for SOUND CARD *New*

iDevNum	**IN**, Device index specified according to pszDevName, start from 0
hAttachedWindow	**IN**, Handle of the window to show the preview video
ppDevice	**OUT**, Handle of the capture card object
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
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_GENERAL	Fail due to some errors

Introduction

Running multiple devices simultaneously is supported. Use this function to get a handle which represents each card.

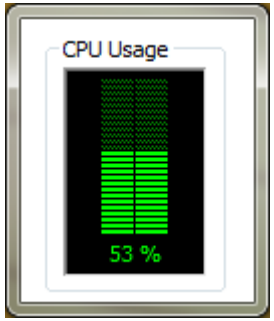
Control different devices in the same process or in different process are all supported.

If attached window is NULL, we will enable device under background mode.

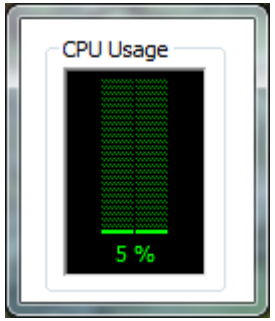
Developer can register both video and audio preview callbacks to obtain the raw frame buffers.

The thumb draw function can reduce the loading of video display, when display window size is small than the input video size.

For multiple channels developer, we suggest to turn on it in order to obtain the best rendering performance.



Normal Display



Thumb Draw Mode

Example

```
QCAP_CREATE( “SA7160 PCI”, 0, m_hWnd, &pDevice, FALSE, FALSE );
```

3.3 QCAP_DESTROY

```
QRESULT QCAP_DESTROY( PVOID pDevice )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Destroy capture card object to release its resource.

Example

```
QCAP_DESTROY( pDevice );
```

3.4 QCAP_RUN

```
QRESULT QCAP_RUN( PVOID pDevice )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Start video and audio capturing.

Example

```
QCAP_RUN( pDevice );
```

3.5 QCAP_STOP

```
QRESULT QCAP_STOP( PVOID pDevice )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Stop video and audio capturing.

Example

```
QCAP_STOP( pDevice );
```

3.6 QCAP_GET_VIDEO_INPUT

```
QRESULT QCAP_GET_VIDEO_INPUT( PVOID pDevice, ULONG * pInput )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pInput	**OUT**, Pointer of storage of input video source

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current video input source.

Example

```
QCAP_GET_VIDEO_INPUT( pDevice, &nInput );
```

3.7 QCAP_SET_VIDEO_INPUT

```
QRESULT QCAP_SET_VIDEO_INPUT( PVOID pDevice, ULONG nInput )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nInput	**IN**, Specify input video source. Available values are 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_AUTO

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set video source.

If nInput set to `QCAP_INPUT_TYPE_AUTO`, video source will be switched automatically if signal missed.

The duration of auto detection function between 2 inputs is 3 sec as default.

Developer can use `QCAP_SET_SYSTEM_CONFIGURATION` to modify it.

Note

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

Example

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

3.8 QCAP_GET_AUDIO_INPUT

```
QRESULT QCAP_GET_AUDIO_INPUT( PVOID pDevice, ULONG * pInput )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pInput	**OUT**, Pointer of storage of input audio source

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current audio input source.

Example

```
QCAP_GET_AUDIO_INPUT( pDevice, &nInput );
```

3.9 QCAP_SET_AUDIO_INPUT

```
QRESULT QCAP_SET_AUDIO_INPUT( PVOID pDevice, ULONG nInput )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nInput	**IN**, Specify input audio source. Available values are QCAP_INPUT_TYPE_EMBEDDED_AUDIO QCAP_INPUT_TYPE_LINE_IN QCAP_INPUT_TYPE_SOUNDCARD_MICROPHONE *New* QCAP_INPUT_TYPE_SOUNDCARD_LINE_IN *New*

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio input source.

Here, the embedded audio input means HDMI or SDI audio.

Note!! Custome can use this function to change to soundcard input;

Moreover, developer alse can use `QCAP_CREATE` function

to create independent input sound card device by `QCAP_CREATE( "SOUNDCARD", … )`.

Example

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

3.10 QCAP_GET_VIDEO_STANDARD

```
QRESULT QCAP_GET_VIDEO_STANDARD( PVOID      pDevice,
                                 ULONG  *   pStandard )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pStandard	**OUT**, Pointer to a variable that stores the standard. It cannot be NULL

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current video standard.

Example

```
QCAP_GET_VIDEO_STANDARD( pDevice, &nStandard );
```

3.11 QCAP_SET_VIDEO_STANDARD

```
QRESULT QCAP_SET_VIDEO_STANDARD( PVOID  pDevice,
                                 ULONG  nStandard )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nStandard	**IN**, Specify input video standard. Available values are 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)

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set current video standard.

If nStandard set to `QCAP_STANDARD_TYPE_AUTO`, video standard will be switched automatically.

Note

If your input signal is unstable, please disable atuo signal detection and set the standard directly.

Example

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

3.12 QCAP_GET_VIDEO_CURRENT_INPUT_FORMAT

```
QRESULT QCAP_GET_VIDEO_CURRENT_INPUT_FORMAT( PVOID   pDevice,
                                             ULONG * pWidth,
                                             ULONG * pHeight,
                                             BOOL  * pIsInterleaved,
                                             ULONG * pFrameRate )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pWidth	**OUT**, Pointer to the storage of current video width
pHeight	**OUT**, Pointer to the storage of current video height
pIsInterleaved	**OUT**, Pointer to the storage of current IsInterleaved
pFrameRate	**OUT**, Pointer to the storage of current frame rate

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current input video format information.

Example

```
QCAP_GET_VIDEO_CURRENT_INPUT_FORMAT( pDevice, &nWidth, &nHeight, &nIsInterleaved, &nFrameRate );
```

3.13 QCAP_GET_AUDIO_CURRENT_INPUT_FORMAT

```
QRESULT QCAP_GET_AUDIO_CURRENT_INPUT_FORMAT( PVOID   pDevice,
                                             ULONG * pChannels,
                                             ULONG * pBitsPerSample,
                                             BOOL  * pSampleFrequency )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pChannels	**OUT**, Pointer to the storage of current channels
pBitsPerSample	**OUT**, Pointer to the storage of current bit per sample
pSampleFrequency	**OUT**, Pointer to the storage of current sample frequency

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current default audio format parameters.

Example

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

3.14 QCAP_GET_VIDEO_DEFAULT_OUTPUT_FORMAT

```
QRESULT QCAP_GET_VIDEO_DEFAULT_OUTPUT_FORMAT(  PVOID      pDevice,
                                               ULONG  *   pWidth,
                                               ULONG  *   pHeight,
                                               BOOL   *   pIsInterleaved,
                                               ULONG  *   pFrameRate )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pWidth	**OUT**, Pointer to the storage of current video width
pHeight	**OUT**, Pointer to the storage of current video height
pIsInterleaved	**OUT**, Pointer to the storage of current IsInterleaved
pFrameRate	**OUT**, Pointer to the storage of current frame rate

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current default video format parameters.

Example

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

3.15 QCAP_SET_VIDEO_DEFAULT_OUTPUT_FORMAT

```
QRESULT QCAP_SET_VIDEO_DEFAULT_OUTPUT_FORMAT( PVOID   pDevice,
                                              ULONG   nWidth,
                                              ULONG   nHeight,
                                              BOOL    bIsInterleaved,
                                              ULONG   nFrameRate )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nWidth	**IN**, Specify video width
nHeight	**IN**, Specify video height
bIsInterleaved	**IN**, Specify video IsInterleaved
nFrameRate	**IN**, Specify video frame rate

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to change default video format output.

In original QCAP concept, the default format output is same as input signal format,

so customer doesn’t need setup output format from any function.

The function is used be advanced user who want to fixed special format output from capture card.

For example, the SC300 outputs 720x480i@60fps at NTSC as default.

You can use the function to change it to 704x480i@60fps.

The function is only opened by all SD capture cards currentlly.

SC5X0 products cannot support it.

Example

```
QCAP_SET_VIDEO_DEFAULT_OUTPUT_FORMAT( pDevice, 704, 480, 1, 60 );
```

3.16 QCAP_GET_AUDIO_DEFAULT_OUTPUT_FORMAT

```
QRESULT QCAP_GET_AUDIO_DEFAULT_OUTPUT_FORMAT(  PVOID   pDevice,
                                               ULONG  * pChannels,
                                               ULONG  * pBitsPerSample,
                                               BOOL   * pSampleFrequency )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pChannels	**OUT**, Pointer to the storage of current channels
pBitsPerSample	**OUT**, Pointer to the storage of current bit per sample
pSampleFrequency	**OUT**, Pointer to the storage of current sample frequency

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current default output audio format parameters.

Example

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

3.17 QCAP_SET_AUDIO_DEFAULT_OUTPUT_FORMAT

```
QRESULT QCAP_SET_AUDIO_DEFAULT_OUTPUT_FORMAT(  PVOID    pDevice,
                                               ULONG    nChannels,
                                               ULONG    nBitsPerSample,
                                               BOOL     nSampleFrequency )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nChannels	**IN**, Specify current channels
nBitsPerSample	**IN**, Specify current bit per sample
nSampleFrequency	**IN**, Specify current sample frequency

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set current default output audio format parameters.

Example

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

3.18 QCAP_GET_VIDEO_DEINTERLACE_TYPE

```
QRESULT QCAP_GET_VIDEO_DEINTERLACE_TYPE( PVOID pDevice, ULONG * pType )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pType	**OUT**, Pointer to a variable that stores the de-interlace

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current video de-interlace method.

Example

```
QCAP_GET_VIDEO_DEINTERLACE_TYPE( pDevice, &Type );
```

3.19 QCAP_SET_VIDEO_DEINTERLACE_TYPE

```
QRESULT QCAP_SET_VIDEO_DEINTERLACE_TYPE( PVOID  pDevice, ULONG  nType )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nType	**IN**, Specify input video de-interlace. Available values are QCAP_SOFTWARE_DEINTERLACE_TYPE_BLENDING QCAP_SOFTWARE_DEINTERLACE_TYPE_MOTIONAD_APTIVE QCAP_SOFTWARE_DEINTERLACE_TYPE_FILTER_TRIANGLE

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set current video de-interlace method.

Example

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

3.20 QCAP_GET_VIDEO_DEINTERLACE

QRESULT QCAP_GET_VIDEO_DEINTERLACE(PVOID pDevice, BOOL * pEnable)

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pEnable	**OUT**, Returned the current status of this function

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get the current status if de-interlace function is enabled.

Example

```
QCAP_GET_VIDEO_DEINTERLACE( pDevice, &bEnable );
```

3.21 QCAP_SET_VIDEO_DEINTERLACE

```
QRESULT QCAP_SET_VIDEO_DEINTERLACE( PVOID pDevice, BOOL bEnable )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
bEnable	**IN**, Enable/Disable this function

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to enable/disable de-interlace function.

The function is only enabled for interleaved video format, such as 1920x1080i@60fps or 720x480i@60fps.

The default algorithm of de-interlace function is the belnding method.

Example

```
QCAP_SET_VIDEO_DEINTERLACE( pDevice, TRUE );
```

3.22 QCAP_GET_VIDEO_BRIGHTNESS

```
QRESULT QCAP_GET_VIDEO_BRIGHTNESS( PVOID pDevice, ULONG * pValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pValue	**OUT**, Return the current brightness

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current brightness.

Example

```
QCAP_GET_VIDEO_BRIGHTNESS( pDevice, &nValue );
```

3.23 QCAP_SET_VIDEO_BRIGHTNESS

```
QRESULT QCAP_SET_VIDEO_BRIGHTNESS( PVOID pDevice, ULONG nValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nValue	**IN**, Specify the value for brightness, from 0 ~ 255

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set brightness.

Example

```
QCAP_SET_VIDEO_BRIGHTNESS( pDevice, 128 );
```

3.24 QCAP_GET_VIDEO_CONTRAST

```
QRESULT QCAP_GET_VIDEO_CONTRAST( PVOID pDevice, ULONG * pValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pValue	**OUT**, Return the current contrast

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current contrast.

Example

```
QCAP_GET_VIDEO_CONTRAST( pDevice, &nValue );
```

3.25 QCAP_SET_VIDEO_CONTRAST

```
QRESULT QCAP_SET_VIDEO_CONTRAST( PVOID pDevice, ULONG nValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nValue	**IN**, Specify the value for contrast, from 0 ~ 255

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set contrast.

Example

```
QCAP_SET_VIDEO_CONTRAST( pDevice, 128 );
```

3.26 QCAP_GET_VIDEO_HUE

```
QRESULT QCAP_GET_VIDEO_HUE( PVOID pDevice, ULONG * pValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pValue	**OUT**, Return the current hue

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get hue.

Example

```
QCAP_GET_VIDEO_HUE( pDevice, &nValue );
```

3.27 QCAP_SET_VIDEO_HUE

```
QRESULT QCAP_SET_VIDEO_HUE( PVOID pDevice, ULONG nValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nValue	**IN**, Specify the value for hue, from 0 ~ 255

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set hue.

Example

```
QCAP_SET_VIDEO_HUE( pDevice, 128 );
```

3.28 QCAP_GET_VIDEO_SATURATION

```
QRESULT QCAP_GET_VIDEO_SATURATION( PVOID pDevice, ULONG * pValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pValue	**OUT**, Return the current saturation

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get saturation.

Example

```
QCAP_GET_VIDEO_SATURATION( pDevice, &nValue );
```

3.29 QCAP_SET_VIDEO_SATURATION

```
QRESULT QCAP_SET_VIDEO_SATURATION( PVOID pDevice, ULONG nValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nValue	**IN**, Specify the value for saturation, from 0 ~ 255

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set saturation.

Example

```
QCAP_SET_VIDEO_SATURATION( pDevice, 128 );
```

3.30 QCAP_GET_VIDEO_SHARPNESS

```
QRESULT QCAP_GET_VIDEO_SHARPNESS( PVOID pDevice, ULONG * pValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pValue	**OUT**, Return the current sharpness

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get sharpness.

Example

```
QCAP_GET_VIDEO_SHARPNESS( pDevice, &nValue );
```

3.31 QCAP_SET_VIDEO_SHARPNESS

```
QRESULT QCAP_SET_VIDEO_SHARPNESS( PVOID pDevice, ULONG nValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nValue	**IN**, Specify the value for sharpness, from 0 ~ 255

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set sharpness.

Example

```
QCAP_SET_VIDEO_SHARPNESS( pDevice, 128 );
```

3.32 QCAP_GET_AUDIO_VOLUME

```
QRESULT QCAP_GET_AUDIO_VOLUME( PVOID pDevice, ULONG * pVolume )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pVolume	**OUT**, Pointer to the storage of audio volume

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get audio volume.

Example

```
QCAP_GET_AUDIO_VOLUME( pDevice, &nVolume );
```

3.33 QCAP_SET_AUDIO_VOLUME

```
QRESULT QCAP_SET_AUDIO_VOLUME( PVOID pDevice, ULONG nVolume )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nVolume	**IN**, Audio volume, from 0 ~ 100, 0 is mute

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio volume.

Example

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

3.34 QCAP_GET_DEVICE_CUSTOM_PROPERTY

```
QRESULT QCAP_GET_DEVICE_CUSTOM_PROPERTY( PVOID    pDevice,
                                         ULONG    nProperty,
                                         ULONG *  pValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nProperty	**IN**, Specifies the property that will be gotten from the device.
pValue	**OUT**, Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is dependent on its property

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is used to get some custom device properties.

Example

For more custom property programming, please reference Yuan's Windows Media SDK and

product’s Extra Prgoramming Guide in SDK packet.

For example, we can obtain device serial number from it.

```
QCAP_GET_DEVICE_CUSTOM_PROPERTY( pDevice,
                                 KSPROPERTY_CUSTOM_GET_DEVICE_SERIAL_NUMBER,
                                &nDeviceSerialNumber );
```

3.35 QCAP_SET_DEVICE_CUSTOM_PROPERTY

```
QRESULT QCAP_GET_DEVICE_CUSTOM_PROPERTY( PVOID    pDevice,
                                         ULONG    nProperty,
                                         ULONG    nValue )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nProperty	**IN**, Specifies the property that will be set to the device.
nValue	**IN**, Specifies the property value. The range of value is dependent on its property.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is used to set/change some custom device properties.

Example

For more custom property programming, please reference Yuan's Windows Media SDK and product’s Extra Prgoramming Guide in SDK packet.

```
QCAP_SET_DEVICE_CUSTOM_PROPERTY( pDevice,
                                 KSPROPERTY_CUSTOM_SET_OSD_COLOR,
                                 nOsdColor );
```

3.36 QCAP_GET_DEVICE_CUSTOM_PROPERTY_EX

```
QRESULT QCAP_GET_DEVICE_CUSTOM_PROPERTY_EX(PVOID     pDevice,
                                           ULONG     nProperty,
                                           BYTE   *  pValue,
                                           ULONG     nBytes )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nProperty	**IN**, Specifies the property that will be gotten from the device.
pValue	**OUT**, Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is dependent on its property.
nBytes	**IN**, Specifies the size of property value.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is used to get some custom device properties.

Example

For more custom property programming, please reference Yuan's Windows Media SDK and

product’s Extra Prgoramming Guide in SDK packet.

For example, we can obtain analog video switch table from it.

```
QCAP_GET_DEVICE_CUSTOM_PROPERTY_EX( pDevice,
                                    KSPROPERTY_CUSTOM_XET_ANALOG_VIDEO_SWITCH_TABLE,
                                    &nDeviceVideoSwitchTable
                                    12 );
```

3.37 QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX

```
QRESULT QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX(PVOID     pDevice,
                                           ULONG     nProperty,
                                           BYTE *    pValue,
                                           ULONG     nBytes )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
nProperty	**IN**, Specifies the property that will be gotten from the device.
pValue	**IN**, Pointer to a variable that stores the specify property value. It cannot be NULL. The range of value is dependent on its property.
nBytes	**IN**, Specifies the size of property value buffer.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is used to set/change some custom device properties.

Example

For more custom property programming, please reference Yuan's Windows Media SDK and

product’s Extra Prgoramming Guide in SDK packet.

```
QCAP_SET_DEVICE_CUSTOM_PROPERTY_EX( pDevice,
                                    KSPROPERTY_CUSTOM_SET_OSD_TEXT_STRING,
                                    NOsdText,
                                    64 );
```

4 Snapshot Function API

4.1 QCAP_SNAPSHOT_BMP

```
QRESULT QCAP_SNAPSHOT_BMP(  PVOID     pDevice,
                            CHAR *    pszFilePathName,
                            BOOL      bIsAsync = TRUE,
                            ULONG     nMilliseconds = 0 )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pszFilePathName	**IN**, Specify the file name to store BMP data "*.BMP24" -> 24bit BMP "*.BMP32" -> 32bit BMP "*.BMP" -> 32bit BMP
bIsAsync	**IN**, Synchronized snapshot flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to snapshot a picture with BMP format

At async mode, QCAP will use one degressive counter to snapshot image.

For example, if you call the function 100 times, you will generate 100 different and continuous images in sequence.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the snapshot is completed.

If the bIsAsync is FALSE, and the nMillisecond is 0, the function will return immediately and generate image file when next frame is coming.

Note

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

Example

```
QCAP_SNAPSHOT_BMP( pDevice, "C:\\PICTURE1.BMP24" ); // PICTURE1.BMP24 -> PICTURE1.BMP
```

4.2 QCAP_SNAPSHOT_JPG

```
QRESULT QCAP_SNAPSHOT_JPG(  PVOID     pDevice,
                            CHAR *    pszFilePathName,
                            ULONG     nQuality,
                            BOOL      bIsAsync = TRUE,
                            ULONG     nMilliseconds = 0 )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pszFilePathName	**IN**, Specify the file name to store JPG data
nQuality	**IN**, Specify the quality of JPG file, from 0 ~ 100
bIsAsync	**IN**, Synchronized snapshot flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to snapshot a picture with JPG format.

At async mode, QCAP will use one degressive counter to snapshot image.

For example, if you call the function 100 times, you will generate 100 different and continuous images in sequence.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the snapshot is completed.

If the bIsAsync is FALSE, and the nMillisecond is 0, the function will return immediately and generate image file when next frame is coming.

Note

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

Example

```
QCAP_SNAPSHOT_JPG( pDevice, "C:\\PICTURE1.JPG", 100 );
```

4.3 QCAP_SNAPSHOT_BMP_EX

```
QRESULT QCAP_SNAPSHOT_BMP_EX(  PVOID     pDevice,
                               CHAR *    pszFilePathName,
                               ULONG     nCropX,
                               ULONG     nCropY,
                               ULONG     nCropW,
                               ULONG     nCropH,
                               ULONG     nDstW,
                               ULONG     nDstH,
                               BOOL      bIsAsync = TRUE,
                               ULONG     nMilliseconds = 0 )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pszFilePathName	**IN**, Specify the file name to store BMP data "*.BMP24" -> 24bit BMP "*.BMP32" -> 32bit BMP "*.BMP" -> 32bit BMP
nCropX	**IN**, Specify the x-coordinate of the crop
nCropY	**IN**, Specify the y-coordinate of the crop
nCropW	**IN**, Specify the horizontal width of crop

nCropH	**IN**, Specify the vertical height of crop
nDstW	**IN**, Specify the horizontal width of current recording down scale
nDstH	**IN**, Specify the vertical height of crop
bIsAsync	**IN**, Synchronized snapshot flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set crop parameters of snapshot a picture with BMP format.

At async mode, QCAP will use one degressive counter to snapshot image.

For example, if you call the function 100 times, you will generate 100 different and continuous images in sequence.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the snapshot is completed.

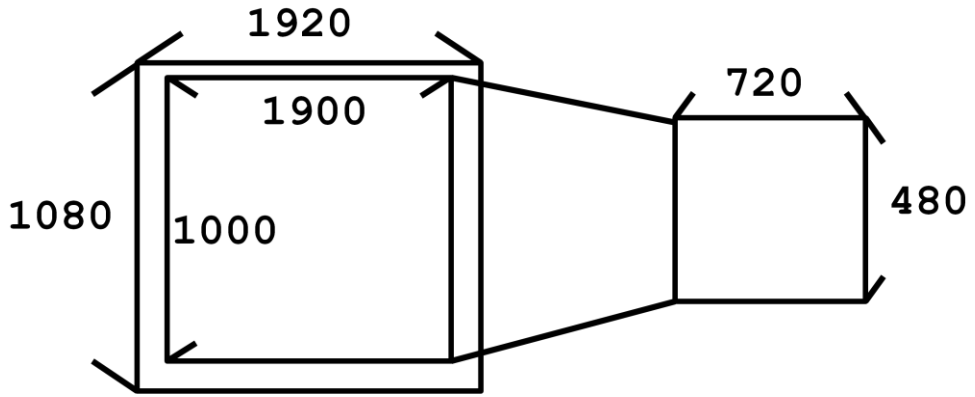
If the bIsAsync is FALSE, and the nMillisecond is 0, the function will return immediately and generate image file when next frame is coming.

Note

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

Example

```
QCAP_SNAPSHOT_BMP_EX( pDevice, “C:\\PICTURE1.BMP”, 10, 40, 1900, 1000, 720, 480 );
```



4.4 QCAP_SNAPSHOT_JPG_EX

```
QRESULT QCAP_SNAPSHOT_JPG_EX(  PVOID     pDevice,
                               CHAR *    pszFilePathName,
                               ULONG     nCropX,
                               ULONG     nCropY,
                               ULONG     nCropW,
                               ULONG     nCropH,
                               ULONG     nDstW,
                               ULONG     nDstH,
                               ULONG     nQuality,
                               BOOL      bIsAsync = TRUE,
                               ULONG     nMilliseconds = 0 )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pszFilePathName	**IN**, Specify the file name to store JPG data
nCropX	**IN**, Specify the x-coordinate of the crop
nCropY	**IN**, Specify the y-coordinate of the crop
nCropW	**IN**, Specify the horizontal width of crop
nCropH	**IN**, Specify the vertical height of crop

nDstW	**IN**, Specify the horizontal width of current recording down scale
nDstH	**IN**, Specify the vertical height of crop
nQuality	**IN**, Specify the quality of JPG file, from 0 ~ 100
bIsAsync	**IN**, Synchronized snapshot flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set crop parameters of snapshot a picture with JPG format.

At async mode, QCAP will use one degressive counter to snapshot image.

For example, if you call the function 100 times, you will generate 100 different and continuous images in sequence.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the snapshot is completed.

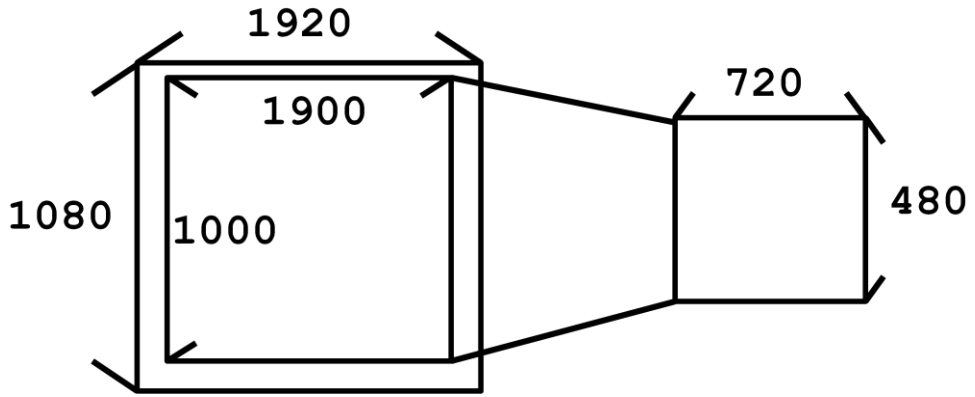
If the bIsAsync is FALSE, and the nMillisecond is 0, the function will return immediately and generate image file when next frame is coming.

Note

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

Example

```
QCAP_SNAPSHOT_JPG_EX( pDevice, “C:\\PICTURE1.JPG”, 10, 40, 1900, 1000, 720, 480 ,100 );
```



5 Recording Function API

5.1 QCAP_GET_VIDEO_RECORD_PROPERTY

```
QRESULT QCAP_GET_VIDEO_RECORD_PROPERTY( PVOID   pDevice,
                                        UINT    iRecNum,
                                        ULONG * pEncoderType,
                                        ULONG * pEncoderFormat,
                                        ULONG * pRecordMode,
                                        ULONG * pQuality,
                                        ULONG * pBitRate,
                                        ULONG * pGOP,
                                        ULONG * pAspectRatioX,
                                        ULONG * pAspectRatioY,
                                        ULONG * pDownscaleMode )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to get recording parameters, start from 0
pEncoderType	**OUT**, Pointer to the storage of current recording encoder type
pEncoderFormat	**OUT**, Pointer to the storage of current recording encoder format
pRecordMode	**OUT**, Pointer to the storage of current recording mode
pQuality	**OUT**, Pointer to the storage of current recording quality
pBitRate	**OUT**, Pointer to the storage of current recording bit rate
pGOP	**OUT**, Pointer to the storage of current recording GOP size
pAspectRatioX	**OUT**, Pointer to the storage of aspect ratio X axle
pAspectRatioY	**OUT**, Pointer to the storage of aspect ratio Y axle
pDownscaleMode	**OUT**, Pointer to the storage of current recording down scale mode

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get video recording parameters.

Example

```
QCAP_GET_VIDEO_RECORD_PROPERTY( pDevice, 0,
                                &nEncoderType, &nEncoderFormat, &nRecordMode, &nQuality, &nBitRate, &nGOP,
                                &nAspectRatioX, &nAspectRatioY, &nDownscaleMode );
```

5.2 QCAP_SET_VIDEO_RECORD_PROPERTY

```
QRESULT QCAP_SET_VIDEO_RECORD_PROPERTY( PVOID   pDevice,
                                        UINT    iRecNum,
                                        ULONG   nEncoderType,
                                        ULONG   nEncoderFormat,
                                        ULONG   nRecordMode,
                                        ULONG   nQuality,
                                        ULONG   nBitRate,
                                        ULONG   nGOP,
                                        ULONG   nAspectRatioX,
                                        ULONG   nAspectRatioY,
                                        ULONG   nDownscaleMode  )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to set recording parameters

nEncoderType	**IN**, Specify encoder type. Available value are
	`QCAP_ENCODER_TYPE_SOFTWARE = 0`
	`QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590`
	`QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2`
	`QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4`

nEncoderFormat	**IN**, Specify encoder format. Available value are `QCAP_ENCODER_FORMAT_H264 = 0`

nRecordMode	**IN**, Specify recording mode. Available values are
	`QCAP_RECORD_MODE_VBR = 0`
	`QCAP_RECORD_MODE_CBR = 1`
	`QCAP_RECORD_MODE_ABR = 2`

nQuality	**IN**, Specify recording quality, from 0 ~ 10000. It is used for VBR and ABR.
nBitRate	**IN**, Specify recording bit rate. It is used for CBR and ABR. (Ex. 12Mbps is 12 x 1024 x 1024 bps)
nGOP	**IN**, Specify recording GOP size, from 0 ~ 255
nAspectRatioX	**IN**, Specify the aspect ratio X axle, 0 is turned off
nAspectRatioY	**IN**, Specify the aspect ratio Y axle, 0 is turned off

nDownscaleMode	**IN**, Specify recording down scale mode. Available values are
	`QCAP_DOWNSCALE_MODE_OFF = 0`
	`QCAP_DOWNSCALE_MODE_2_3 = 1 // EX,1920X1080 -> 1280X720`
	`QCAP_DOWNSCALE_MODE_1_2 = 2 // EX,1920X1080 -> 960X540`
	`QCAP_DOWNSCALE_MODE_1_4 = 3 // EX,1920X1080 -> 480X270`

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set video recording parameters.

For the hardware compression card, SC290, SC390 and UB658G,

the nEncoderType is always `QCAP_ENCODER_TYPE_HARDWARE` at iRecNum 0 recording.

The parameter, nDownscaleMode, is `QCAP_DOWNSCALE_MODE_OFF`, too.

For the dual-stream hardware compression card, SC2B0, SC3B0, SC2A0, SC3A0, SC580 and SC590,

the nEncoderType is always `QCAP_ENCODER_TYPE_HARDWARE` at iRecNum 0 and iRecNum 1 recording.

Here, iRecNum 0 is used by main H.264 hardware encoder and iRecNum 1 is used by sub H.264 hardware encoder.

The parameter, nDownscaleMode, is `QCAP_DOWNSCALE_MODE_OFF` at iRecNum 0 and `QCAP_DOWNSCALE_MODE_1_2` at iRecNum 1.

<table>
<tr><th>Products</th><th>iRecNum 0</th><th>iRecNum 1</th></tr>
<tr><td>UB658G</td><td rowspan="6">QCAP_ENCODER_TYPE_HARDWARE
+
QCAP_DOWNSCALE_MODE_OFF</td><td rowspan="2">Software Video Encoder</td></tr>
<tr><td>SC290 / SC390</td></tr>
<tr><td>SC2B0 / SC3B0</td><td rowspan="4">QCAP_ENCODER_TYPE_HARDWARE
+
QCAP_DOWNSCALE_MODE_1_2</td></tr>
<tr><td>SC2A0 / SC3A0</td></tr>
<tr><td>SC580</td></tr>
<tr><td>SC590</td></tr>
</table>

Example

```
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 );
```

5.3 QCAP_GET_VIDEO_RECORD_PROPERTY_EX (CROPPING)

```
QRESULT QCAP_GET_VIDEO_RECORD_PROPERTY_EX ( PVOID   pDevice,
                                            UINT    iRecNum,
                                            ULONG * pEncoderType,
                                            ULONG * pEncoderFormat,
                                            ULONG * pRecordProfile,
                                            ULONG * pRecordLevel,
                                            ULONG * pRecordEntropy,
                                            ULONG * pRecordComplexity,
                                            ULONG * pRecordMode,
                                            ULONG * pQuality,
                                            ULONG * pBitRate,
                                            ULONG * pGOP,
                                            ULONG * pBFrames,
                                            ULONG * pAspectRatioX,
                                            ULONG * pAspectRatioY,
                                            ULONG * pCroX
                                            ULONG * pCroY
                                            ULONG * pCroW
                                            ULONG * pCropH
                                            ULONG * pDstW
                                            ULONG * pDstH )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to get recording parameters, start from 0
pEncoderType	**OUT**, Pointer to the storage of current recording encoder type
pEncoderFormat	**OUT**, Pointer to the storage of current recording encoder format
pRecordProfile	**OUT**, Pointer to the storage of current recording profile
pRecordLevel	**OUT**, Pointer to the storage of current recording level
pRecordEntropy	**OUT**, Pointer to the storage of current recording entropy
pRecordComplexity	**OUT**, Pointer to the storage of current recording complexity
pRecordMode	**OUT**, Pointer to the storage of current recording mode
pQuality	**OUT**, Pointer to the storage of current recording quality
pBitRate	**OUT**, Pointer to the storage of current recording bit rate
pGOP	**OUT**, Pointer to the storage of current recording GOP size
pBFrames	**OUT**, Pointer to the storage of current B frames
pAspectRatioX	**OUT**, Pointer to the storage of aspect ratio X axle
pAspectRatioY	**OUT**, Pointer to the storage of aspect ratio Y axle
pCroX	**OUT**, Pointer to the storage of x-coordinate of the crop
pCroY	**OUT**, Pointer to the storage of y-coordinate of the crop

pCroW	**OUT**, Pointer to the storage of horizontal width of crop
pCropH	**OUT**, Pointer to the storage of vertical height of crop
pDstW	**OUT**, Pointer to the storage of horizontal width of current recording down scale
pDstH	**OUT**, Pointer to the storage of vertical height of current recording down scale

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get video recording parameters and crop parameters or advance parameters from H.264 Encoder, such as Profile, Level, Entropy, Complexity and B Frames.

The extern function get supports the cropping function and the various up/down scaling function.

Example

```
QCAP_GET_VIDEO_RECORD_PROPERTY_EX( pDevice, 0, &nEncoderType, &nEncoderFormat, &nRecordProfile,
                                   &nRecordLevel, &nRecordEntropy, &nRecordComplexity, &nRecordMode,
                                   &nQuality, &nBitRate, &nGOP, &nBFrames, &nAspectRatioX, &nAspectRatioY,
                                   &nCropX, &nCropY, &nCropW, &nCropH, &nDstW, &nDstH );
```

5.4 QCAP_SET_VIDEO_RECORD_PROPERTY_EX (CROPPING)

```
QRESULT QCAP_SET_VIDEO_RECORD_PROPERTY_EX (PVOID   pDevice,
                                           UINT    iRecNum,
                                           ULONG   nEncoderType,
                                           ULONG   nEncoderFormat,
                                           ULONG   nRecordProfile,
                                           ULONG   nRecordLevel,
                                           ULONG   nRecordEntropy,
                                           ULONG   nRecordComplexity,
                                           ULONG   nRecordMode,
                                           ULONG   nQuality,
                                           ULONG   nBitRate,
                                           ULONG   nGOP,
                                           ULONG   nBFrames,
                                           ULONG   nAspectRatioX,
                                           ULONG   nAspectRatioY,
                                           ULONG   nCropX
                                           ULONG   nCropY
                                           ULONG   nCropW
                                           ULONG   nCropH
                                           ULONG   nDstW
                                           ULONG   nDstH )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to set recording parameters
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_H264 = 0

nRecordProfile	**IN**, Specify recording profile
	QCAP_RECORD_PROFILE_BASELINE = 0, // DEFAULT
	QCAP_RECORD_PROFILE_MAIN = 1,
	QCAP_RECORD_PROFILE_HIGH = 2

nRecordLevel	**IN**, Specify recording level QCAP_RECORD_LEVEL_1 = 10, QCAP_RECORD_LEVEL_1B = 9, QCAP_RECORD_LEVEL_11 = 11, QCAP_RECORD_LEVEL_12 = 12, QCAP_RECORD_LEVEL_13 = 13, QCAP_RECORD_LEVEL_2 = 20, QCAP_RECORD_LEVEL_21 = 21, QCAP_RECORD_LEVEL_22 = 22,

	QCAP_RECORD_LEVEL_3 = 30,
	QCAP_RECORD_LEVEL_31 = 31,
	QCAP_RECORD_LEVEL_32 = 32,
	QCAP_RECORD_LEVEL_4 = 40,
	QCAP_RECORD_LEVEL_41 = 41, // DEFAULT
	QCAP_RECORD_LEVEL_42 = 42,
	QCAP_RECORD_LEVEL_50 = 50,
	QCAP_RECORD_LEVEL_51 = 51,
	QCAP_RECORD_LEVEL_52 = 52,

nRecordEntropy	**IN**, Specify recording entropy
	QCAP_RECORD_ENTROPY_CAVLC = 0, // DEFAULT
	QCAP_RECORD_ENTROPY_CABAC = 1,

nRecordComplexity	**IN**, Specify recording complexity
	QCAP_RECORD_COMPLEXITY_0 = 0, // BEST SPEED, DEFAULT
	QCAP_RECORD_COMPLEXITY_1 = 1,
	QCAP_RECORD_COMPLEXITY_2 = 2,
	QCAP_RECORD_COMPLEXITY_3 = 3,
	QCAP_RECORD_COMPLEXITY_4 = 4,
	QCAP_RECORD_COMPLEXITY_5 = 5,
	QCAP_RECORD_COMPLEXITY_6 = 6, // BEST QUALITY

nRecordMode	**IN**, Specify recording mode. Available values are QCAP_RECORD_MODE_VBR = 0 QCAP_RECORD_MODE_CBR = 1 QCAP_RECORD_MODE_ABR = 2
nQuality	**IN**, Specify recording quality, from 0 ~ 10000. It is used for VBR and ABR.

nBitRate	**IN**, Specify recording bit rate. It is used for CBR and ABR. (Ex. 12Mbps is 12 x 1024 x 1024 bps)
nGOP	**IN**, Specify recording GOP size, from 0 ~ 255
nBFrames	**IN**, Specify recording B frames
nAspectRatioX	**IN**, Specify the aspect ratio X axle, 0 is turned off
nAspectRatioY	**IN**, Specify the aspect ratio Y axle, 0 is turned off
nCropX	**IN**, Specify the x-coordinate of the crop
nCropY	**IN**, Specify the y-coordinate of the crop
nCropW	**IN**, Specify the horizontal width of crop
nCropH	**IN**, Specify the vertical height of crop
nDstW	**IN**, Specify the horizontal width of current recording down scale
nDstH	**IN**, Specify the vertical height of current recording down scale

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set video recording parameters and crop parameters or advance parameters from H.264 encoder such as Profile, Level, Entropy, Complexity and B Frames.

The extern function set supports the cropping function and the various up/down scaling function.

Example

```
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, 4, 3, 10, 40, 1900, 1000, 720, 480 );
```

5.5 QCAP_GET_AUDIO_RECORD_PROPERTY

```
QRESULT QCAP_GET_AUDIO_RECORD_PROPERTY( PVOID   pDevice,
                                        UINT    iRecNum,
                                        ULONG * pEncoderType,
                                        ULONG * pEncoderFormat )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to get recording parameters, start from 0
pEncoderType	**OUT**, Pointer to the storage of current recording encoder type
pEncoderFormat	**OUT**, Pointer to the storage of current recording encoder format

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get audio recording parameters.

Example

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

5.6 QCAP_SET_AUDIO_RECORD_PROPERTY

```
QRESULT QCAP_SET_AUDIO_RECORD_PROPERTY( PVOID   pDevice,
                                        UINT    iRecNum,
                                        ULONG   nEncoderType,
                                        ULONG   nEncoderFormat )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to set recording parameters
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_PCM = 0 QCAP_ENCODER_FORMAT_AAC = 1 QCAP_ENCODER_FORMAT_AAC_RAW = 1 QCAP_ENCODER_FORMAT_AAC_ADTS = 2

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio recording parameters.

For MP4 file recording, we suggest to use H.264 + AAC encoder. It is the most popular file format for Microsoft Windows 7 as default.

If customer doesn’t want to license AAC, the QCAP SDK also support H.264 + PCM file format for your software.

Note

Developer uses TS, FLV, RTMP and HLS must to set audio format as AAC_ADTS.

Example

```
QCAP_SET_AUDIO_RECORD_PROPERTY( pDevice, 0, QCAP_ENCODER_TYPE_SOFTWARE, QCAP_ENCODER_FORMAT_AAC );
```

About AAC License

This software development kit contains software programming code and libraries related to AAC function that is subject to your development use only. All the intellectual properties are held by individual license groups and/or the property holders. License and royalty fee should be charged by the license groups and/or property holders and paid by you if any of them is implemented into your commercial product. We will not be responsible for any illegal usage and/or any infringement of the rights of the individual owners of these intellectual properties.

Details of AAC could be found in the website: http://www.vialicensing.com/licensing/aac-overview.aspx

5.7 QCAP_START_RECORD

```
QRESULT QCAP_START_RECORD( PVOID  pDevice,
                           UINT   iRecNum,
                           CHAR * pszFilePathName,
                           DWORD  dwFlags = QCAP_RECORD_FLAG_FULL )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to start recording
pszFilePathName	**IN**, Specify the file name of storage for recording. Recording file format is AVI, if extension name is AVI. Recording file format is MP4, if extension name is MP4. Recording file format is ASF, if extension name is ASF. Recording file format is FLV, if extension name is FLV. Recording file format is TS, if extension name is TS.
dwFlags	**IN**, Specify the action flag of share recording. This parameter can also include any combination of the following flags. Available value are QCAP_RECORD_FLAG_FULL = 0x00000007 QCAP_RECORD_FLAG_FILE = 0x00000001 QCAP_RECORD_FLAG_ENCODE = 0x00000002 QCAP_RECORD_FLAG_DISPLAY = 0x00000004 QCAP_RECORD_FLAG_VIDEO_ONLY = 0x00000010 QCAP_RECORD_FLAG_AUDIO_ONLY = 0x00000020

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start recording.

QCAP support the multiple-streams recording at the same time.

It means you can generate many recording files with different resolution or bit rate for one capture device.

The dwFlags parameter allows you to turn on/off internal components of recording.

Example

```
QCAP_START_RECORD( pDevice, 0, "D:\\CH01.AVI" ); // @ 1920X1080 ( QCAP_DOWNSCALE_MODE_OFF )

QCAP_START_RECORD( pDevice, 1, "D:\\CH02.MP4" ); // @ 1280X720 ( QCAP_DOWNSCALE_MODE_2_3 )
```

5.8 QCAP_STOP_RECORD

```
QRESULT QCAP_STOP_RECORD(  PVOID  pDevice,
                           UINT   iRecNum,
                           BOOL   bIsAsync = TRUE,
                           ULONG  nMilliseconds = 0 )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to stop recording
bIsAsync	**IN**, Synchronized stop recording flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop recording.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the recording file is closed.

Note

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

Example

```
QCAP_STOP_RECORD( pDevice, 0 );
```

5.9 QCAP_RESUME_RECORD

```
QRESULT QCAP_RESUME_RECORD(   PVOID  pDevice,
                              UINT   iRecNum )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to resume recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to resume recording.

Example

```
QCAP_RESUME_RECORD( pDevice, 0 );
```

5.10 QCAP_PAUSE_RECORD

```
QRESULT QCAP_PAUSE_RECORD( PVOID  pDevice,
                           UINT   iRecNum )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to pause recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to pause recording.

Example

```
QCAP_PAUSE_RECORD( pDevice, 0 );
```

6 Synchronized Recording Function API

6.1 QCAP_START_SYNCHRONIZED_RECORD

```
QRESULT QCAP_START_SYNCHRONIZED_RECORD( ULONG     nFileArgs,
                                        CHAR *    pszFilePathName1,
                                        PVOID     pDevice1,
                                        UINT      iRecNum1,
                                        CHAR *    pszFilePathName2,
                                        PVOID     pDevice2,
                                        UINT      iRecNum2, … )
```

Parameter

Name	Description
nFileArgs	**IN**, Set the devices number
pszFilePathName1	**IN**, Specify the file name of storage for recording. Recording file format is MP4, if extension name is MP4. Recording file format is ASF, if extension name is ASF. Recording file format is FLV, if extension name is FLV. Recording file format is TS, if extension name is TS.
pDevice1, pDevice2, …	**IN**, Handle of the capture card object
iRecNum1, iRecNum2, …	**IN**, Specify recorder index to start synchronized recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start synchronized recording. At this function, QCAP will give unified sample time to start multiple recording.

In other words, QCAP will provide same start sample time for many recording. User can use nFileArgs parameter to set number of devices.

Note

AVI format cannot support synchronized recording.

Example

```
QCAP_START_SYNCHRONIZED_RECORD( 4,
                                "SR_CH01.MP4", pDevices[ 0 ],
                                 "SR_CH02.MP4", pDevices[ 1 ], 0,
                                 "SR_CH03.MP4", pDevices[ 2 ], 0,
                                 "SR_CH04.MP4", pDevices[ 3 ], 0, NULL, NULL, 0, NULL, NULL, 0 );
```

6.2 QCAP_STOP_SYNCHRONIZED_RECORD

```
QRESULT QCAP_STOP_SYNCHRONIZED_RECORD(  BOOL   bIsAsync,
                                        ULONG  nMilliseconds,
                                        ULONG  nFileArgs,
                                        PVOID  pDevice1,
                                        UINT   iRecNum1,
                                        PVOID  pDevice2,
                                        UINT   iRecNum2, … )
```

Parameter

Name	Description
bIsAsync	**IN**, Synchronized stop recording flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.
nFileArgs	**IN**, Set the devices number
pDevice1, pDevice2, …	**IN**, Handle of the capture card object
iRecNum1, iRecNum2, …	**IN**, Specify recorder index to stop synchronized recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop synchronized recording.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the recording file is closed.

User can use nFileArgs parameter to set number of devices.

Note

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

Example

```
QCAP_STOP_SYNCHRONIZED_RECORD( TRUE, 0, 4,
                               pDevices[ 0 ], 0,
                               pDevices[ 1 ], 0,
                               pDevices[ 2 ], 0,
                               pDevices[ 3 ], 0, NULL, 0 );
```

6.3 QCAP_RESUME_SYNCHRONIZED_RECORD

```
QRESULT QCAP_RESUME_SYNCHRONIZED_RECORD(    ULONG  nFileArgs,
                                            PVOID  pDevice1,
                                            UINT   iRecNum1,
                                            PVOID  pDevice2,
                                            UINT   iRecNum2, … )
```

Parameter

Name	Description
nFileArgs	**IN**, Set the devices number
pDevice1, pDevice2, …	**IN**, Handle of the capture card object
iRecNum1, iRecNum2, …	**IN**, Specify recorder index to resume synchronized recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to resume synchronized recording.

User can use nFileArgs parameter to set number of devices.

Example

```
QCAP_RESUME_SYNCHRONIZED_RECORD( 4, pDevice, 0 );
```

6.4 QCAP_PAUSE_SYNCHRONIZED_RECORD

```
QRESULT QCAP_PAUSE_SYNCHRONIZED_RECORD( ULONG  nFileArgs,
                                        PVOID  pDevice1,
                                        UINT   iRecNum1,
                                        PVOID  pDevice2,
                                        UINT   iRecNum2, … )
```

Parameter

Name	Description
nFileArgs	**IN**, Set the devices number
pDevice1, pDevice2, …	**IN**, Handle of the capture card object
iRecNum1, iRecNum2, …	**IN**, Specify recorder index to pause synchronized recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to pause synchronized recording.

User can use nFileArgs parameter to set number of devices.

Example

```
QCAP_PAUSE_SYNCHRONIZED_RECORD( 4, pDevice, 0 );
```

7 OSD Function API

7.1 QCAP_SET_OSD_TEXT

```
QRESULT QCAP_SET_OSD_TEXT(    PVOID     pDevice,
                              UINT      iOsdNum,
                              INT       x,
                              INT       y,
                              INT       width,
                              INT       height,
                              CHAR *    pszString,
                              CHAR *    pszFontFamilyName,
                              ULONG     nFontStyle,
                              ULONG     nFontSize,
                              DWORD     dwFontColor,
                              DWORD     dwBackgroundColor,
                              ULONG     nTransparent,
                              ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object

iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output, if width is 0, QCAP will auto calculate width.
height	**IN**, Specify the vertical height of OSD output, if height is 0, QCAP will auto calculate height.
pszString	**IN**, Specify to display the text of OSD output
pszFontFamilyName	**IN**, Specify the font name used to display the text of OSD output
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
nFontSize	**IN**, Specify the font size used to display the text of OSD output
dwFontColor	**IN**, Specify the font color used to display the text of OSD output
dwBackgroundColor	**IN**, Specify the background color used to display the text of OSD output
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a text field objects used for on screen display of information.

If you want to get a text field object auto calculated actual width and height value; you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Note

Color space of FontColor is ABGR, don’t forgot to set its alpha value.

Color space of BackgroundColor is ABGR, don’t forgot to set its alpha value.

Transparent is one global parameter for all colors.

Example

```
QCAP_SET_OSD_TEXT( hHandle, 0, 0, 0, 0, 0, “CH01”, “Arial”,

                   QCAP_FONT_STYLE_BOLD, 12, 0xFF000000,

                   0xFFFFFFFF, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

7.2 QCAP_SET_OSD_PICTURE

```
QRESULT QCAP_SET_OSD_PICTURE( PVOID      pDevice,
                              UINT       iOsdNum,
                              INT        x,
                              INT        y,
                              INT        width,
                              INT        height,
                              CHAR *     pszFilePathName,
                              ULONG      nTransparent
                              ULONG      nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output If width is 0, QCAP will use original picture width as default.
height	**IN**, Specify the vertical height of OSD output. If height is 0, QCAP will use original picture height as default.

pszFilePathName	**IN**, Specify the file name to display image OSD output, support a file extension of "BMP" as 24 or 32-bit, “JPG” and “PNG”
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a static image objects used for on screen display of information.

If you want to get a static image object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_PICTURE( hHandle, 0, 0, 0, 0, 0, “C:\\SAMPLE.PNG”, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

7.3 QCAP_SET_OSD_BUFFER

```
QRESULT QCAP_SET_OSD_BUFFER(  PVOID     pDevice,
                              UINT      iOsdNum,
                              INT       x,
                              INT       y,
                              INT       width,
                              INT       height,
                              ULONG     nColorSpaceType,
                              BYTE *    pFrameBuffer,
                              ULONG     nFrameWidth,
                              ULONG     nFrameHeight,
                              ULONG     nFramePitch,
                              ULONG     nTransparent
                              ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output
height	**IN**, Specify the vertical height of OSD output
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer
nFrameWidth	**IN**, Specify the horizontal width of frame contained in a buffer
nFrameHeight	**IN**, Specify the vertical height of frame contained in a buffer
nFramePitch	**IN**, Specify the distance in bytes between the start of one scan line to the next. If it is 0, we will auto calculate it by width and color space format.
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255

nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

* The performance of BGR type is slower than RGB in our alpha blending algorithm.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a frame buffer objects used for on screen display of information.

The function’s purpose is used to point directly to a valid frame buffer whose entire pixel data,

which is stored in available color space, and display it.

Beside, you also can adjust the width and height parameters to scale the OSD output

and OSD sequence style to video-streams video-recording the OSD output.

Example

```
QCAP_SET_OSD_BUFFER( hHandle, 0, 0, 0, 1920, 1080,

                     QCAP_COLORSPACE_TYEP_YUY2,

                     pFrameBuffer, 800, 600, 800 * 2, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

8 Clone Function API

8.1 QCAP_CREATE_CLONE

```
QRESULT QCAP_CREATE_CLONE( PVOID     pDevice,
                           HWND      hAttachedWindow,
                           PVOID *   ppCloneDevice,
                           BOOL      bThumbDraw = FALSE,
                           BOOL      bMaintainAspectRatio = FALSE )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
hAttachedWindow	**IN**, Handle of the window to show the preview clone video
ppCloneDevice	**OUT**, Handle of the clone capture device
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the Maintain aspect ratio

Returned Value

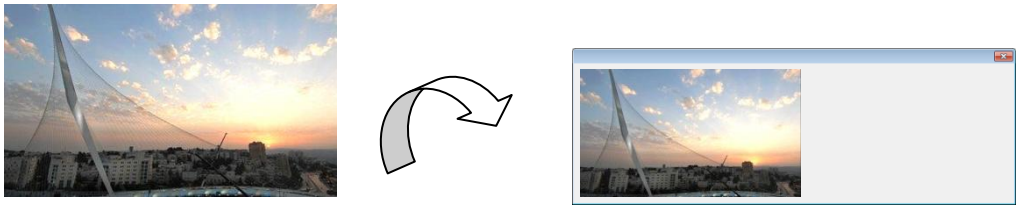
Name	Description
QCAP_RS_SUCCESSFUL	Success

QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a clone device from current hardware device object.

The purpose of clone function is to help user to create multiple preview windows from one hardware capture device.



Physical Device Display Clone Display on Other Window

Developer also can use the clone device to access device's property function, such as QCAP_SET_VIDEO_INPUT.

QCAP will be auto mapping to its physical device object.

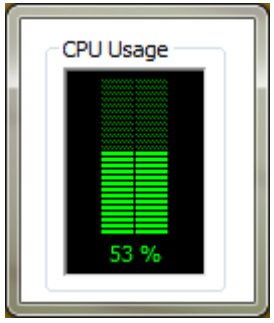
However, every clone device also can access own independently functions.

These independent functions are listed as below:

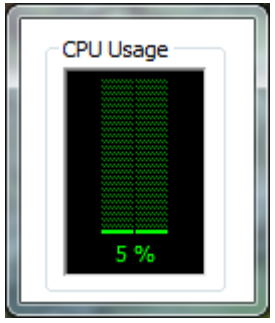
Related Clone API Functions
QCAP_CREATE_CLONE
QCAP_RUN
QCAP_STOP
QCAP_DESTROY
QCAP_GET_VOLUME / QCAP_SET_VOLUME

The thumb draw function can reduce the loading of video display, when display window size is small than the input video size.

For multiple channels developer, we suggest to turn on it in order to obtain the best rendering performance.



Normal Display



Thumb Draw Mode

Programming Workflow of Clone Device

Programming Step	Related API Functions
Create Device	`QCAP_CREATE( …, &pDevice )`
Start Capturing	`QCAP_RUN( pDevice )`
Create Clone Device	`QCAP_CREATE_CLONE( pDevice, …, &pClone, … )`
Start Clone Preview	`QCAP_RUN( pClone )`
Stop Clone Preview	`QCAP_STOP( pClone )`
Delete Clone Device	`QCAP_DESTROY( pClone )`
Stop Capturing	`QCAP_STOP( pDevice )`
Delete Device	`QCAP_DESTROY( pDevice )`

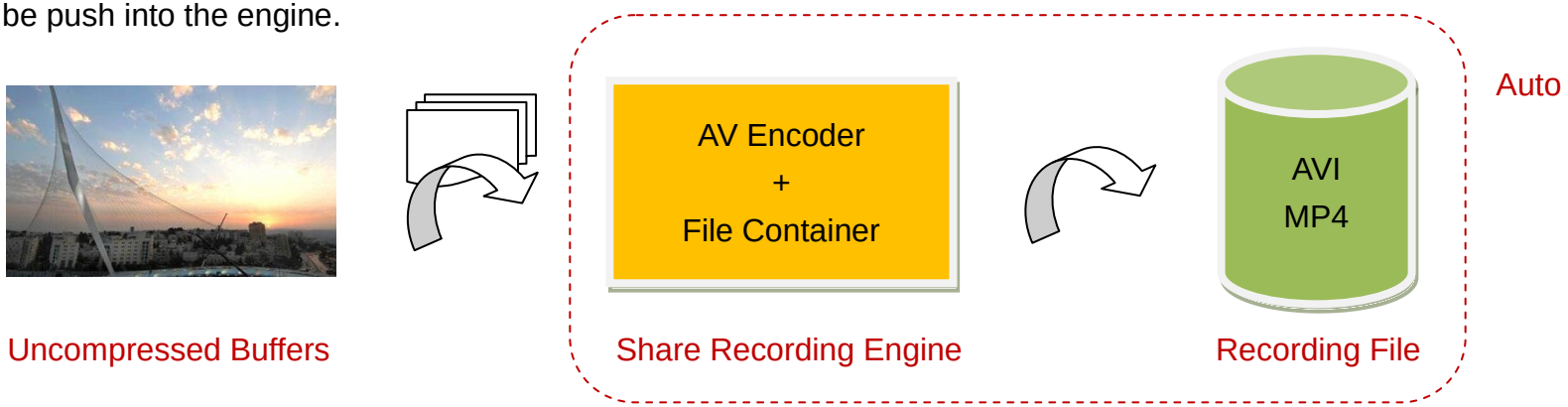
Example

```
QCAP_CREATE_CLONE( pDevice, m_hWnd, &pClone, FALSE, FALSE );
```

9 Share Recording Function API

9.1 Overview

The function of share recording can help you to create one customized recording file easily. It is different from channel recording at Chapter 5, the software developer can freedom combine any audio and video frame buffers into one recording engine. The engine will hide all complex actions for stream encoding and file saving. For one software developer, you just need to decide which frame buffer will be push into the engine.



For instance, we can combine CH1 and CH2 video into one H.264's MP4 file. The first 5min video is come from CH1 and remains are CH2.

9.2 QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY

```
QRESULT QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY(  UINT    iRecNum,
                                               ULONG * pEncoderType,
                                               ULONG * pEncoderFormat,
                                               ULONG * pColorSpaceType,
                                               ULONG * pWidth,
                                               ULONG * pHeight,
                                               ULONG * pFrameRate,
                                               ULONG * pRecordMode,
                                               ULONG * pQuality,
                                               ULONG * pBitRate,
                                               ULONG * pGOP,
                                               ULONG * pAspectRatioX,
                                               ULONG * pAspectRatioY,
                                               HWND *  pAttachedWindow = NULL,
                                               BOOL *  pThumbDraw = NULL,
                                               BOOL *  pMaintainAspectRatio = NULL )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to get recording parameters, start from 0
pEncoderType	**OUT**, Pointer to the storage of current recording encoder type
pEncoderFormat	**OUT**, Pointer to the storage of current recording encoder format
pColorSpaceType	**OUT**, Pointer to the storage of current recording encoder color space type
pWidth	**OUT**, Pointer to the storage of current recording encoder width
pHeight	**OUT**, Pointer to the storage of current recording encoder height
pFrameRate	**OUT**, Pointer to the storage of current recording encoder frame rate
pRecordMode	**OUT**, Pointer to the storage of current recording mode
pQuality	**OUT**, Pointer to the storage of current recording quality
pBitRate	**OUT**, Pointer to the storage of current recording bit rate
pGOP	**OUT**, Pointer to the storage of current recording GOP size
pAspectRatioX	**OUT**, Pointer to the storage of aspect ratio X axle
pAspectRatioY	**OUT**, Pointer to the storage of aspect ratio Y axle
pAttachedWindow	**OUT**, Pointer to the storage of window to show the recording video
pThumbDraw	**OUT**, Pointer to the enable flag of thumb draw renderer
pMaintainAspectRatio	**OUT**, Pointer to the enable flag of maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get video share recording parameters.

Example

```
QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY( 0, &nEncoderType, &nEncoderFormat,
                                      &nColorSpaceType, &nWidth, &nHeight, &nFrameRate,
                                      &nRecordMode, &nQuality, &nBitRate, &nGOP,
                                      &pAspectRatioX, &npAspectRatioY, &hAttachedWindow,
                                      &bThumbDraw, &MaintainAspectRatio );
```

9.3 QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY

```
QRESULT QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY(  UINT    iRecNum,
                                               ULONG   nEncoderType,
                                               ULONG   nEncoderFormat,
                                               ULONG   nColorSpaceType,
                                               ULONG   nWidth,
                                               ULONG   nHeight,
                                               ULONG   nFrameRate,
                                               ULONG   nRecordMode,
                                               ULONG   nQuality,
                                               ULONG   nBitRate,
                                               ULONG   nGOP,
                                               ULONG   nAspectRatioX,
                                               ULONG   nAspectRatioY,
                                               HWND    hAttchedWindow = NULL,
                                               BOOL    bThumbDraw = FALSE,
                                               BOOL    bMaintainAspectRatio = FALSE )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to set recording parameters
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_H264 = 0
nColorSpaceType	**IN**, Specify encoder color space type. Available values are 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
nWidth	**IN**, Specify encoder width.
nHeight	**IN**, Specify encoder height.
nFrameRate	**IN**, Specify encoder frame rate.

nRecordMode	**IN**, Specify recording mode. Available values are
	`QCAP_RECORD_MODE_VBR = 0`
	`QCAP_RECORD_MODE_CBR = 1`
	`QCAP_RECORD_MODE_ABR = 2`

nQuality	**IN**, Specify recording quality, from 0 ~ 10000. It is used for VBR and ABR.
nBitRate	**IN**, Specify recording bit rate. It is used for CBR and ABR.(Ex. 12Mbps is 12 x 1024 x 1024 bps)
nGOP	**IN**, Specify recording GOP size, from 0 ~ 255
nAspectRatioX	**IN**, Specify the aspect ratio X axle, 0 is turned off
nAspectRatioY	**IN**, Specify the aspect ratio Y axle, 0 is turned off
hAttachedWindow	**IN**, Handle of the window to show the preview video
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

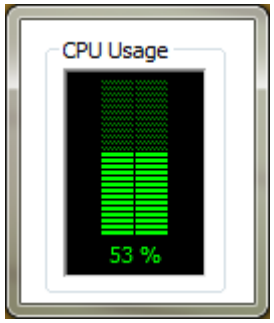
Introduction

User can use this function to set video recording parameters.

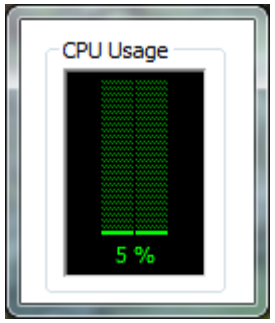
By using the attached window, QCAP can help you to real-time display the share recording's result.

The thumb draw function can reduce the loading of video display, when display window size is small than the input video size.

For multiple channels developer, we suggest to turn on it in order to obtain the best rendering performance.



Normal Display



Thumb Draw Mode

Example

```
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 );
```

9.4 QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX

```
QRESULT QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX(  UINT    iRecNum,
                                                  ULONG * pEncoderType,
                                                  ULONG * pEncoderFormat,
                                                  ULONG * pColorSpaceType,
                                                  ULONG * pWidth,
                                                  ULONG * pHeight,
                                                  ULONG * pFrameRate,
                                                  ULONG * pRecordProfile,
                                                  ULONG * pRecordLevel,
                                                  ULONG * pRecordEntropy,
                                                  ULONG * pRecordComplexity,
                                                  ULONG * pRecordMode,
                                                  ULONG * pQuality,
                                                  ULONG * pBitRate,
                                                  ULONG * pGOP,
                                                  ULONG * pBFrames,
                                                  ULONG * pAspectRatioX,
                                                  ULONG * pAspectRatioY,
                                                  HWND *  pAttachedWindow = NULL,
                                                  BOOL *  pThumbDraw = NULL,
                                                  BOOL *  pMaintainAspectRatio = NULL )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to get recording parameters, start from 0
pEncoderType	**OUT**, Pointer to the storage of current recording encoder type
pEncoderFormat	**OUT**, Pointer to the storage of current recording encoder format
pColorSpaceType	**OUT**, Pointer to the storage of current recording encoder color space type
pWidth	**OUT**, Pointer to the storage of current recording encoder width
pHeight	**OUT**, Pointer to the storage of current recording encoder height
pFrameRate	**OUT**, Pointer to the storage of current recording encoder frame rate
pRecordProfile	**OUT**, Pointer to the storage of current recording profile
pRecordLevel	**OUT**, Pointer to the storage of current recording level
pRecordEntropy	**OUT**, Pointer to the storage of current recording entropy
pRecordComplexity	**OUT**, Pointer to the storage of current recording complexity
pRecordMode	**OUT**, Pointer to the storage of current recording mode
pQuality	**OUT**, Pointer to the storage of current recording quality
pBitRate	**OUT**, Pointer to the storage of current recording bit rate
pGOP	**OUT**, Pointer to the storage of current recording GOP size
pBFrames	**OUT**, Pointer to the storage of current recording B frames
pAspectRatioX	**OUT**, Pointer to the storage of aspect ratio X axle

pAspectRatioY	**OUT**, Pointer to the storage of aspect ratio Y axle
pAttachedWindow	**OUT**, Pointer to the storage of window to show the recording video
pThumbDraw	**OUT**, Pointer to the enable flag of thumb draw renderer
pMaintainAspectRatio	**OUT**, Pointer to the enable flag of maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get video share recording parameters or even parameters from H.264 system such as Profile, Level, Entropy Complexity and B Frames.

Example

```
QCAP_GET_VIDEO_SHARE_RECORD_PROPERTY_EX( 0, &nEncoderType, &nEncoderFormat,
                                         &nColorSpaceType, &nWidth, &nHeight, &nFrameRate,
                                         & nRecordProfile,  &nRecordLevel, &nRecordEntropy,
                                         & nRecordComplexity, &nRecordMode, &nQuality, &nBitRate,
                                         &nGOP, &nBframe, &nAspectRatioX, &npAspectRatioY,
                                         &hAttachedWindow, &nbThumbDraw, &nMaintainAspectRatio );
```

9.5 QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX

```
QRESULT QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY_EX(  UINT    iRecNum,
                                                  ULONG   nEncoderType,
                                                  ULONG   nEncoderFormat,
                                                  ULONG   nColorSpaceType,
                                                  ULONG   nWidth,
                                                  ULONG   nHeight,
                                                  ULONG   nFrameRate,
                                                  ULONG   nRecordProfile,
                                                  ULONG   nRecordLevel,
                                                  ULONG   nRecordEntropy,
                                                  ULONG   nRecordComplexity,
                                                  ULONG   nRecordMode,
                                                  ULONG   nQuality,
                                                  ULONG   nBitRate,
                                                  ULONG   nGOP,
                                                  ULONG   nBFrame,
                                                  ULONG   nAspectRatioX,
                                                  ULONG   nAspectRatioY,
                                                  HWND    hAttchedWindow = NULL,
                                                  BOOL    bThumbDraw = FALSE,
                                                  BOOL    bMaintainAspectRatio = FALSE )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to set recording parameters
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK= 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_H264 = 0
nColorSpaceType	**IN**, Specify encoder color space type. Available values are 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_YV1 QCAP_COLORSPACE_TYEP_I420
nWidth	**IN**, Specify encoder width.
nHeight	**IN**, Specify encoder height.
nFrameRate	**IN**, Specify encoder frame rate.
nRecordProfile	**IN**, Specify recording profile QCAP_RECORD_PROFILE_BASELINE = 0, // DEFAULT

	QCAP_RECORD_PROFILE_MAIN = 1, QCAP_RECORD_PROFILE_HIGH = 2

nRecordLevel	**IN**, Specify recording level
	QCAP_RECORD_LEVEL_1 = 10,
	QCAP_RECORD_LEVEL_1B = 9,
	QCAP_RECORD_LEVEL_11 = 11,
	QCAP_RECORD_LEVEL_12 = 12,
	QCAP_RECORD_LEVEL_13 = 13,
	QCAP_RECORD_LEVEL_2 = 20,
	QCAP_RECORD_LEVEL_21 = 21,
	QCAP_RECORD_LEVEL_22 = 22,
	QCAP_RECORD_LEVEL_3 = 30,
	QCAP_RECORD_LEVEL_31 = 31,
	QCAP_RECORD_LEVEL_32 = 32,
	QCAP_RECORD_LEVEL_4 = 40,
	QCAP_RECORD_LEVEL_41 = 41, // DEFAULT
	QCAP_RECORD_LEVEL_42 = 42,
	QCAP_RECORD_LEVEL_50 = 50,
	QCAP_RECORD_LEVEL_51 = 51,
	QCAP_RECORD_LEVEL_52 = 52,

nRecordEntropy	**IN**, Specify recording entropy
	QCAP_RECORD_ENTROPY_CAVLC = 0, // DEFAULT
	QCAP_RECORD_ENTROPY_CABAC = 1,

nRecordComplexity	**IN**, Specify recording complexity QCAP_RECORD_COMPLEXITY_0 = 0, // BEST SPEED, DEFAULT

	QCAP_RECORD_COMPLEXITY_1 = 1, QCAP_RECORD_COMPLEXITY_2 = 2, QCAP_RECORD_COMPLEXITY_3 = 3, QCAP_RECORD_COMPLEXITY_4 = 4, QCAP_RECORD_COMPLEXITY_5 = 5, QCAP_RECORD_COMPLEXITY_6 = 6, // BEST QUALITY
nRecordMode	**IN**, Specify recording mode. Available values are QCAP_RECORD_MODE_VBR = 0 QCAP_RECORD_MODE_CBR = 1 QCAP_RECORD_MODE_ABR = 2
nQuality	**IN**, Specify recording quality, from 0 ~ 10000. It is used for VBR and ABR.
nBitRate	**IN**, Specify recording bit rate. It is used for CBR and ABR. (Ex. 12Mbps is 12 x 1024 x 1024 bps)
nGOP	**IN**, Specify recording GOP size, from 0 ~ 255
nBFrames	**IN**, Specify recording Bframe
nAspectRatioX	**IN**, Specify the aspect ratio X axle, 0 is turned off
nAspectRatioY	**IN**, Specify the aspect ratio Y axle, 0 is turned off
hAttachedWindow	**IN**, Handle of the window to show the preview video
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

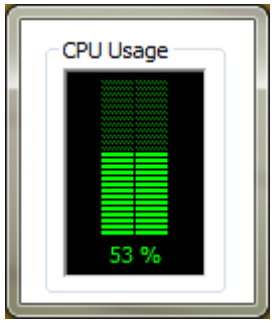
Introduction

User can use this function to set video share recording parameters or even parameters from H.264 system such as Profile, Level, Entropy Complexity and B Frames.

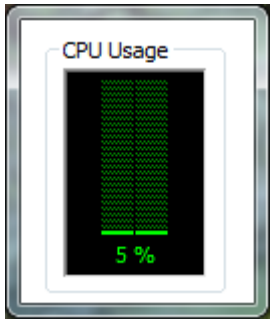
By using the attached window, QCAP can help you to real-time display the share recording’s result.

The thumb draw function can reduce the loading of video display, when display window size is small than the input video size.

For multiple channels developer, we suggest to turn on it in order to obtain the best rendering performance.



Normal Display



Thumb Draw Mode

Example

```
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, 4, 3,
                                         hAttachedWindow, FALSE, FALSE );
```

9.6 QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY

```
QRESULT QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY(  UINT    iRecNum,
                                               ULONG * pEncoderType,
                                               ULONG * pEncoderFormat,
                                               ULONG * pChannels,
                                               ULONG * pBitsPerSample,
                                               ULONG * pSampleFrequency,
                                               ULONG * pVolume )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to get recording parameters, start from 0
pEncoderType	**OUT**, Pointer to the storage of current recording encoder type
pEncoderFormat	**OUT**, Pointer to the storage of current recording encoder format
pChannels	**OUT**, Pointer to the storage of current recording encoder channels
pBitsPerSample	**OUT**, Pointer to the storage of current recording encoder bits per sample
pSampleFrequency	**OUT**, Pointer to the storage of current recording encoder sample frequency
pVolume	**OUT**, Pointer to the storage of current audio renderer’s volume

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get audio recording parameters.

Example

```
QCAP_GET_AUDIO_SHARE_RECORD_PROPERTY( 0, &nEncoderType, &nEncoderFormat,
                                      &nChannels, &nBitsPerSample, &nSampleFrequency, &nVolume );
```

9.7 QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY

```
QRESULT QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY(  UINT    iRecNum,
                                               ULONG   nEncoderType,
                                               ULONG   nEncoderFormat,
                                               ULONG   nChannels,
                                               ULONG   nBitsPerSample,
                                               ULONG   nSampleFrequency,
                                               ULONG   nVolume )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index to set recording parameters
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_PCM = 0 QCAP_ENCODER_FORMAT_AAC = 1 QCAP_ENCODER_FORMAT_AAC_ADTS = 2
nChannels	**IN**, Specify encoder channels.
nBitsPerSample	**IN**, Specify encoder bits per sample.

nSampleFrequency	**IN**, Specify encoder sample frequency.
nVolume	**IN**, Specify the volume of audio renderer, from 0 ~ 100

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio recording parameters.

For MP4 file recording, we suggest to use H.264 + AAC encoder. It is the most popular file format for Microsoft Windows 7 as default.

If customer doesn't want to license AAC, the QCAP SDK also support H.264 + PCM file format for your software.

Example

```
QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( 0, QCAP_ENCODER_TYPE_SOFTWARE, QCAP_ENCODER_FORMAT_AAC,
                                      2, 16, 48000, 50 );
```

About AAC License

This software development kit contains software programming code and libraries related to AAC function that is subject to your development use only. All the intellectual properties are held by individual license groups and/or the property holders. License and royalty fee should be charged by the license groups and/or property holders and paid by you if any of them is implemented into your commercial product. We will not be responsible for any illegal usage and/or any infringement of the rights of the individual owners of these intellectual properties.

Details of AAC could be found in the website:

http://www.vialicensing.com/licensing/aac-overview.aspx

9.8 QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER

```
QRESULT QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( UINT     iRecNum ,
                                                          ULONG    nColorSpaceType,
                                                          ULONG    nWidth,
                                                          ULONG    nHeight,
                                                          BYTE *   pFrameBuffer,
                                                          ULONG    nFrameBufferLen,
                                                          double   dSampleTime = 0.0 )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to start recording
nColorSpaceType	**IN**, Specify the input source buffer's color space type, Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
nWidth	**IN**, Specify the input source buffer's width
nHeight	**IN**, Specify the input source buffer's height
pFrameBuffer	**IN**, Specify the input source buffer

nFrameBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

* The performance of RGB type is slower than BGR in our color converter algorithm.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push un-compression video buffer into share recorder.

Recorder will send the buffer into the encoder and generate its bit-stream.

Finally, the bit-stream can be saved to disk by the file writer.

If the width or the height of source frame are different from the target frame size.

QCAP will enable one auto video scaler to resize it before encoding.

Example

```
QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, QCAP_COLORSPACE_TYEP_YUY2, 1280, 720,
                                                  pFrameBuffer, 1280 * 720 * 2 );
```

9.9 QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER

```
QRESULT QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( UINT     iRecNum ,
                                                          BYTE *   pFrameBuffer,
                                                          ULONG    nFrameBufferLen,
                                                          double   dSampleTime = 0.0 )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to start recording
pFrameBuffer	**IN**, Specify the input source buffer
nFrameBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push un-compression audio buffer into share recorder.

Recorder will send the buffer into the encoder and generate its bit-stream.

Finally, the bit-stream can be saved to disk by the file writer.

Example

```
QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( 0, pFrameBuffer, 4096 );
```

9.10 QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER

```
QRESULT QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER(  UINT      iRecNum ,
                                                         BYTE *    pStreamBuffer,
                                                         ULONG     nStreamBufferLen,
                                                         ULONG     bIsKeyFrame,
                                                         double    dSampleTime = 0.0 )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to start recording
pStreamBuffer	**IN**, Specify the input source buffer
nStreamBufferLen	**IN**, Specify the input source buffer's size
bIsKeyFrame	**IN**, Specify the input source's frame whether it is key-frame or not
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push compression stream buffer directly into share recorder.

The function will not enable encoder.

Example

```
QCAP_SET_VIDEO_SHARE_RECORD_COMPRESSION_BUFFER( 0, pStreamBuffer, 4096, TRUE );
```

9.11 QCAP_SET_AUDIO_SHARE_RECORD_COMPRESSION_BUFFER

```
QRESULT QCAP_SET_AUDIO_SHARE_RECORD_COMPRESSION_BUFFER(  UINT      iRecNum ,
                                                         BYTE *    pStreamBuffer,
                                                         ULONG     nStreamBufferLen,
                                                         double    dSampleTime = 0.0 )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to start recording
pStreamBuffer	**IN**, Specify the input source buffer
nStreamBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push compression stream buffer directly into share recorder.

The function will not enable encoder.

Example

```
QCAP_SET_AUDIO_SHARE_RECORD_COMPRESSION_BUFFER( 0, pStreamBuffer, 4096 );
```

9.12 QCAP_START_SHARE_RECORD

```
QRESULT QCAP_START_SHARE_RECORD( UINT   iRecNum,
                                 CHAR * pszFilePathName,
                                 DWORD  dwFlags = QCAP_RECORD_FLAG_FULL )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to start recording
pszFilePathName	**IN**, Specify the file name of storage for recording. Recording file format is AVI, if extension name is AVI. Recording file format is MP4, if extension name is MP4. Recording file format is ASF, if extension name is ASF. Recording file format is FLV, if extension name is FLV. Recording file format is TS, if extension name is TS.
dwFlags	**IN**, Specify the action flag of share recording. This parameter can also include any combination of the following flags. Available value are QCAP_RECORD_FLAG_FULL = 0x00000007 QCAP_RECORD_FLAG_FILE = 0x00000001 QCAP_RECORD_FLAG_ENCODE = 0x00000002 QCAP_RECORD_FLAG_DISPLAY = 0x00000004 QCAP_RECORD_FLAG_DECODE = 0x00000008 QCAP_RECORD_FLAG_VIDEO_ONLY = 0x00000010 QCAP_RECORD_FLAG_AUDIO_ONLY = 0x00000020

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start share recording.

The dwFlags parameter allows you to turn on/off internal components of recording.

For example, if you don’t want to display video/audio stream, the dwFlags is `QCAP_RECORD_FLAG_FILE | QCAP_RECORD_FLAG_ENCODE`.

Note

`QCAP_RECORD_FLAG_DECODE` is new parameter. If user wants to show video date from compression buffer, the new parameter of dwFlags can be used to preview video date from compression buffer on the window.

For example, if you want to display video stream from compression buffer,

the dwFlags is `QCAP_RECORD_FLAG_FILE | QCAP_RECORD_FLAG_DECODE`.

Programming Workflow of Share Recording

Programming Step	Related API Functions
Set Video Record Property	`QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( idx, … )`
Set Audio Record Property	`QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( idx, … )`
Start Share Recording	`QCAP_START_SHARE_RECORD( idx, … )`
Push Video Frame Buffer into Recorder *	`QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( idx, … )`
Push Audio Frame Buffer into Recorder *	`QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( idx, … )`
Stop Share Recording	`QCAP_STOP_SHARE_RECORD( idx, … )`

* The function is called for several times.

Example

```
QCAP_START_SHARE_RECORD( 0, "D:\\CH01.AVI" );

QCAP_START_SHARE_RECORD( 1, "D:\\CH02.MP4" );
```

9.13 QCAP_STOP_SHARE_RECORD

```
QRESULT QCAP_STOP_SHARE_RECORD(   UINT   iRecNum,
                                  BOOL   bIsAsync = TRUE,
                                  ULONG  nMilliseconds = 0 )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to stop recording
bIsAsync	**IN**, Synchronized stop recording flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop share recording.

If the bIsAsync is FALSE, and the nMillisecond is INFINITE, the function will be blocked and returned until the recording file is closed.

Note

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

Example

```
QCAP_STOP_SHARE_RECORD( 0 );
```

9.14 QCAP_RESUME_SHARE_RECORD

```
QRESULT QCAP_RESUME_SHARE_RECORD( UINT iRecNum )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to resume recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to resume share recording.

Example

```
QCAP_RESUME_SHARE_RECORD( 0 );
```

9.15 QCAP_PAUSE_SHARE_RECORD

QRESULT QCAP_PAUSE_SHARE_RECORD(UINT iRecNum)

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to pause recording

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to pause share recording.

Example

```
QCAP_PAUSE_SHARE_RECORD( 0 );
```

9.16 QCAP_SET_OSD_SHARE_RECORD_TEXT

```
QRESULT QCAP_SET_OSD_SHARE_RECORD_TEXT( UINT      iRecNum,
                                        UINT      iOsdNum,
                                        INT       x,
                                        INT       y,
                                        INT       width,
                                        INT       height,
                                        CHAR *    pszString,
                                        CHAR *    pszFontFamilyName,
                                        ULONG     nFontStyle,
                                        ULONG     nFontSize,
                                        DWORD     dwFontColor,
                                        DWORD     dwBackgroundColor,
                                        ULONG     nTransparent,
                                        ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to set OSD output
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output

y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output, if width is 0, QCAP will auto calculate width.
height	**IN**, Specify the vertical height of OSD output, if height is 0, QCAP will auto calculate height.
pszString	**IN**, Specify to display the text of OSD output
pszFontFamilyName	**IN**, Specify the font name used to display the text of OSD output
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
nFontSize	**IN**, Specify the font size used to display the text of OSD output
dwFontColor	**IN**, Specify the font color used to display the text of OSD output
dwBackgroundColor	**IN**, Specify the background color used to display the text of OSD output
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

We also support to create a text field objects used for on screen display of information

when the users use the share record function to save record to a file.

If you want to get a text field object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_SHARE_RECORD_TEXT(   0, 0, 0, 0, 0, 0, “CH01”, “Arial”,
                                  QCAP_FONT_STYLE_BOLD, 12, 0xFF000000,
                                  0xFFFFFFFF, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

9.17 QCAP_SET_OSD_SHARE_RECORD_PICTURE

```
QRESULT QCAP_SET_OSD_SHARE_RECORD_PICTURE( UINT       iRecNum,
                                           UINT       iOsdNum,
                                           INT        x,
                                           INT        y,
                                           INT        width,
                                           INT        height,
                                           CHAR *     pszFilePathName,
                                           ULONG      nTransparent,
                                           ULONG      nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to set OSD output
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output If width is 0, QCAP will use original picture width as default.
height	**IN**, Specify the vertical height of OSD output. If height is 0, QCAP will use original picture height as default.

pszFilePathName	**IN**, Specify the file name to display image OSD output, support a file extension of "BMP" as 24 or 32-bit, “JPG” and “PNG”
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

We also support to create a static image objects used for on screen display of information

when the users use the share record function to save record to a file.

If you want to get a text field object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_SHARE_RECORD_PICTURE( 0, 0, 0, 0, 0, 0, “C:\\SAMPLE.PNG”, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

9.18 QCAP_SET_OSD_SHARE_RECORD_BUFFER

```
QRESULT QCAP_SET_OSD_SHARE_RECORD_BUFFER ( UINT      iRecNum,
                                           UINT      iOsdNum,
                                           INT       x,
                                           INT       y,
                                           INT       width,
                                           INT       height,
                                           ULONG     nColorSpaceType,
                                           BYTE *    pFrameBuffer,
                                           ULONG     nFrameWidth,
                                           ULONG     nFrameHeight,
                                           ULONG     nFramePitch,
                                           ULONG     nTransparent,
                                           ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index to set OSD output
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output

width	**IN**, Specify the horizontal width of OSD output
height	**IN**, Specify the vertical height of OSD output
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer
nFrameWidth	**IN**, Specify the horizontal width of frame contained in a buffer
nFrameHeight	**IN**, Specify the vertical height of frame contained in a buffer
nFramePitch	**IN**, Specify the distance in bytes between the start of one scan line to the next. If it is 0, we will auto calculate it by width and color space format.
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

We also support to create a frame buffer objects used for on screen display of information

when the users use the share record function to save record to a file.

The function’s purpose is used to point directly to a valid frame buffer whose entire pixel data,

which is stored in available color space, and display it.

Beside, you also can adjust the width and height parameters to scale the OSD output

and OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_SHARE_RECORD_BUFFER( 0, 0, 0, 0, 1920, 1080,

                                  QCAP_COLORSPACE_TYEP_YUY2, pFrameBuffer,

                                  800, 600, 800 * 2, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

10 File Playback/Editing Function API

10.1 QCAP_OPEN_FILE

```
QRESULT QCAP_OPEN_FILE(    CHAR  *    pszFileName,
                           PVOID *    ppFile,
                           ULONG      nDecoderType,
                           ULONG *    pVideoFormat,
                           ULONG *    pVideoWidth,
                           ULONG *    pVideoHeight,
                           double *   pVideoFrameRate,
                           ULONG *    pAudioFormat,
                           ULONG *    pAudioChannels,
                           ULONG *    pAudioBitsPerSample,
                           ULONG *    pAudioSampleFrequency,
                           double *   pTotalDurationTimes,
                           ULONG *    pTotalVideoFrames,
                           ULONG *    pTotalAudioFrames,
                           HWND       hAttachedWindow,
                           BOOL       bThumbDraw = FALSE,
                           BOOL       bMaintainAspectRatio = FALSE )
```

Parameter

Name	Description
pszFileName	**IN**, Specify the video file name to display video output, support a file extension of "MP4" and "AVI"
ppFile	**OUT**, Handle of the file object

nDecoderType	**IN**, Specify decoder type. Available value are
	`QCAP_DECODER_TYPE_SOFTWARE = 0`
	`QCAP_DECODER_TYPE_INTEL_MEDIA_SDK = 2`

pVideoFormat	**OUT**, Pointer to the storage of the video format
pVideoWidth	**OUT**, Pointer to the storage of the video width
pVideoHeight	**OUT**, Pointer to the storage of the video height
pVideoFrameRate	**OUT**, Pointer to the storage of the video frame rate
pAudioFormat	**OUT**, Pointer to the storage of the audio format
pAudioChannels	**OUT**, Pointer to the storage of the audio channel
pAudioBitsPerSample	**OUT**, Pointer to the storage of the audio bit per sample
pAudioSampleFrequency	**OUT**, Pointer to the storage of the audio sample frequency
pTotalDurationTimes	**OUT**, Pointer to the storage of the total duration times
pTotalVideoFrames	**OUT**, Pointer to the storage of the total video frames
pTotalAudioFrames	**OUT**, Pointer to the storage of the audio frames
hAttachedWindow	**IN**, Handle of the window to show the preview video. If it is NULL, the file will be used at Editing mode only.

bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

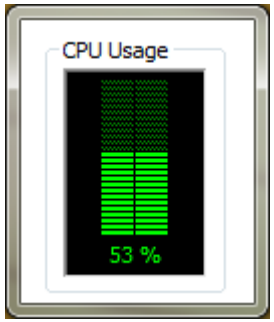
Introduction

User can use this function to create one video file used for on screen playback.

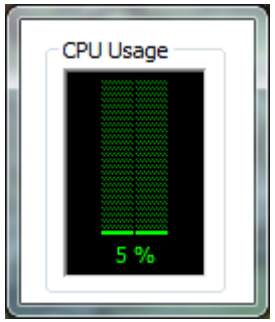
If you want to use editing mode to access this file, you can set the hAttachedWindow as NULL.

The thumb draw function can reduce the loading of video display, when display window size is small than the input video size.

For multiple channels developer, we suggest to turn on it in order to obtain the best rendering performance.



Normal Display



Thumb Draw Mode

Example

```
QCAP_OPEN_FILE( "D:\\CH01.AVI", &pFile, QCAP_DECODER_TYPE_SOFTWARE, &nVideoFormat, &nVideoWidth,
                &nVideoHeight, &nVideoFrameRate, &nAudioFormat, &nAudioChannels, &nAudioBitsPerSample,
                &nAudioSampleFrequency, &nTotalDurationTimes, &nTotalVideoFrames, &nTotalAudioFrames,
                hAttachedWindow );
```

10.2 QCAP_DESTROY_FILE

QRESULT QCAP_DESTROY_FILE(PVOID pFile)

Parameter

Name	Description
pFile	**IN**, Handle of the filer object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Destroy file object to release its resource.

Example

```
QCAP_DESTROY_FILE( pFile );
```

10.3 QCAP_PLAY_FILE

QRESULT QCAP_PLAY_FILE(PVOID pFile)

Parameter

Name	Description
pFile	**IN**, Handle of the filer object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to play file.

Example

```
QCAP_PLAY_FILE( pFile );
```

10.4 QCAP_PAUSE_FILE

```
QRESULT QCAP_PAUSE_FILE( PVOID pFile )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to pause file.

Example

```
QCAP_PAUSE_FILE( pFile );
```

10.5 QCAP_STOP_FILE

```
QRESULT QCAP_STOP_FILE( PVOID pFile )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop file.

Example

```
QCAP_STOP_FILE( pFile );
```

10.6 QCAP_GET_FILE_POSITION

```
QRESULT QCAP_GET_FILE_POSITION( PVOID pFile, double * pSampleTime )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
pSampleTime	**OUT**, Pointer to the storage of sample time

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get video frame position such as video seek function.

Example

```
QCAP_GET_FILE_POSITION( pFile, &nSampleTime );
```

10.7 QCAP_SET_FILE_POSITION

```
QRESULT QCAP_SET_FILE_POSITION( PVOID pFile, double dSampleTime )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
dSampleTime	**IN**, Specify the timestamp of this frame.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current file position such as video seek function.

Example

```
QCAP_SET_FILE_POSITION( pFile, SetSampleTime );
```

10.8 QCAP_GET_FILE_PLAYBACK_SPEED

```
QRESULT QCAP_GET_FILE_PLAYBACK_SPEED( PVOID pFile, double * pSpeed )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
pSpeed	**OUT**, Pointer to the storage of speed

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get current file playback speed.

Example

```
QCAP_GET_FILE_PLAYBACK_SPEED( pFile, &nSpeed );
```

10.9 QCAP_SET_FILE_PLAYBACK_SPEED

QRESULT QCAP_SET_FILE_PLAYBACK_SPEED(PVOID pFile, double dSpeed)

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
dSpeed	**IN**, Specify the speed

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set and control file playback speed.

Example

```
QCAP_SET_FILE_PLAYBACK_SPEED( pFile, nSpeed  );
```

10.10 QCAP_GET_FILE_AUDIO_VOLUME

```
QRESULT QCAP_GET_FILE_AUDIO_VOLUME( PVOID pFile, ULONG * pVolume )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
pVolume	**OUT**, Pointer to the storage of audio volume

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get audio volume of file.

Example

```
QCAP_GET_FILE_AUDIO_VOLUME( pFile, nVolume );
```

10.11 QCAP_SET_FILE_AUDIO_VOLUME

```
QRESULT QCAP_SET_FILE_AUDIO_VOLUME( PVOID pFile, ULONG  nVolume )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
nVolume	**IN**, Specify the audio volume

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio volume of file.

Example

```
QCAP_SET_FILE_AUDIO_VOLUME( pFile, 50 );
```

10.12 QCAP_SET_OSD_FILE_TEXT

```
QRESULT QCAP_SET_OSD_FILE_TEXT(   PVOID      pFile,
                                  UINT       iOsdNum,
                                  INT        x,
                                  INT        y,
                                  INT        width,
                                  INT        height,
                                  CHAR *     pszString,
                                  CHAR *     pszFontFamilyName,
                                  ULONG      nFontStyle,
                                  ULONG      nFontSize,
                                  DWORD      dwFontColor,
                                  DWORD      dwBackgroundColor,
                                  ULONG      nTransparent,
                                  ULONG      nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output

y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output, if width is 0, QCAP will auto calculate width.
height	**IN**, Specify the vertical height of OSD output, if height is 0, QCAP will auto calculate height.
pszString	**IN**, Specify to display the text of OSD output
pszFontFamilyName	**IN**, Specify the font name used to display the text of OSD output
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
nFontSize	**IN**, Specify the font size used to display the text of OSD output
dwFontColor	**IN**, Specify the font color used to display the text of OSD output
dwBackgroundColor	**IN**, Specify the background color used to display the text of OSD output
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

When the users create a file object,

If you want to get a text field object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_FILE_TEXT( pFile, 0, 0, 0, 0, 0, "CH01", "Arial",
                        QCAP_FONT_STYLE_BOLD, 12, 0xFF000000,
                        0xFFFFFFFF, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

10.13 QCAP_SET_OSD_FILE_PICTURE

```
QRESULT QCAP_SET_OSD_FILE_PICTURE(   PVOID      pFile,
                                     UINT       iOsdNum,
                                     INT        x,
                                     INT        y,
                                     INT        width,
                                     INT        height,
                                     CHAR *     pszFilePathName,
                                     ULONG      nTransparent,
                                     ULONG      nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output If width is 0, QCAP will use original picture width as default.
height	**IN**, Specify the vertical height of OSD output. If height is 0, QCAP will use original picture height as default.

pszFilePathName	**IN**, Specify the file name to display image OSD output, support a file extension of "BMP" as 24 or 32-bit, “JPG” and “PNG”
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

When the users create a file object,

If you want to get a text field object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_FILE_PICTURE( pFile, 0, 0, 0, 0, 0,
                           “C:\\SAMPLE.PNG”,
                           128,
                           QCAP_SEQUENCE_STYLE_FOREMOST  );
```

10.14 QCAP_SET_OSD_FILE_BUFFER

```
QRESULT QCAP_SET_OSD_FILE_BUFFER( PVOID    pFile,
                                  UINT     iOsdNum,
                                  INT      x,
                                  INT      y,
                                  INT      width,
                                  INT      height,
                                  ULONG    nColorSpaceType,
                                  BYTE *   pFrameBuffer,
                                  ULONG    nFrameWidth,
                                  ULONG    nFrameHeight,
                                  ULONG    nFramePitch,
                                  ULONG    nTransparent
                                  ULONG    nSequenceStyle =  QCAP_SEQUENCE_STYLE_FOREMOST );
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output
height	**IN**, Specify the vertical height of OSD output
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer
nFrameWidth	**IN**, Specify the horizontal width of frame contained in a buffer
nFrameHeight	**IN**, Specify the vertical height of frame contained in a buffer
nFramePitch	**IN**, Specify the distance in bytes between the start of one scan line to the next. If it is 0, we will auto calculate it by width and color space format.
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255

nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

When the users create a file object,

The function’s purpose is used to point directly to a valid frame buffer whose entire pixel data,

which is stored in available color space, and display it.

Beside, you also can adjust the width and height parameters to scale the OSD output.

Example

```
QCAP_SET_OSD_FILE_BUFFER( pFile, 0, 0, 0, 1920, 1080, QCAP_COLORSPACE_TYEP_YUY2, pFrameBuffer,
                          800, 600, 800 * 2, 128 );
```

10.15 QCAP_GET_FILE_VIDEO_STREAM_BUFFER

```
QRESULT QCAP_GET_FILE_VIDEO_STREAM_BUFFER( PVOID      pFile,
                                           UINT       iFrameNum,
                                           BYTE   *   pStreamBuffer,
                                           ULONG  *   pStreamBufferLen,
                                           BOOL   *   pIsKeyFrame,
                                           double *   pSampleTime )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
iFrameNum	**IN**, Specify the number of frame
pStreamBuffer	**OUT**, Pointer to the storage of the input source buffer
pStreamBufferLen	**IN**, Specify the max input source buffer's size **OUT**, Pointer to the storage of actual input source buffer's size
pIsKeyFrame	**OUT**, Pointer to the storage of the key frame
pSampleTime	**OUT**, Pointer to the storage of sample time

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get information from video stream buffer.

Example

```
QCAP_GET_FILE_VIDEO_STREAM_BUFFER( pFile, 100, &nStreamBuffer, &nStreamBufferLen, &nIsKeyFrame,
                                   &nSampleTime );
```

10.16 QCAP_GET_FILE_AUDIO_STREAM_BUFFER

```
QRESULT QCAP_GET_FILE_AUDIO_STREAM_BUFFER( PVOID      pFile,
                                           UINT       iFrameNum,
                                           BYTE   *   pStreamBuffer,
                                           ULONG  *   pStreamBufferLen,
                                           double *   pSampleTime )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
iFrameNum	**IN**, Specify the number of frame
pStreamBuffer	**OUT**, Pointer to the storage of the input source buffer
pStreamBufferLen	**IN**, Specify the max input source buffer's size **OUT**, Pointer to the storage of actual input source buffer's size
pSampleTime	**OUT**, Pointer to the storage of sample time

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get information from audio stream buffer.

Example

```
QCAP_GET_FILE_AUDIO_STREAM_BUFFER( pFile, 100, &nStreamBuffer, &nStreamBufferLen, &nSampleTime );
```

10.17 QCAP_EXPORT_FILE

```
QRESULT QCAP_EXPORT_FILE(  PVOID     pFile,
                           double    dStartSampleTime,
                           double    dStopSampleTime,
                           CHAR *    pszExportedFileName )
```

Parameter

Name	Description
pFile	**IN**, Handle of the filer object
dStartSampleTime	**IN**, Specify the start sample time, in sec unit
dStopSampleTime	**IN**, Specify the stop sample time, in sec unit
pszExportedFileName	**IN**, Specify the file name to store data, support a file extension of "MP4" and "AVI"

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

It is one helper function.

User can use this function to export one file segment from start sample time to stop start sample time.

Example

```
QCAP_EXPORT_FILE( pFile, 5.000, 10.000, “C:\\EXPORT.MP4” );
```

10.18 QCAP_MERGE_FILE

```
QRESULT QCAP_MERGE_FILE(   PVOID     pFrontEndFile,
                           PVOID     pBackEndFile,
                           CHAR *    pszMergedFileName )
```

Parameter

Name	Description
pFrontEndFile	**IN**, Specify the front end file
pBackEndFile	**IN**, Specify the back end file
pszMergedFileName	**IN**, Specify the merged file name, support a file extension of "MP4" and "AVI"

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

It is one helper function.

User can use this function to merge two files into one file.

Note

The recording format of two files must to be same.

You cannot combine one 1920x1080 file and one 1280x720 file together.

Example

```
QCAP_MERGE_FILE( pFrontEndFile, pBackEndFile, “C:\\MERGE.MP4” );
```

11 Broadcasting Server & Client Function API

11.1 Overview

In this section of the document, you will understand how to design a streaming server for RTSP, RTMP and HLS.

They all can be designed into your software and are enabled at the same time.

For different streaming protocols, only their created function is different in QCAP.

For example, if you had already used `QCAP_CREATE_BROADCAST_RTSP_SERVER` to create one RTSP server,

you can change it to `QCAP_CREATE_BROADCAST_HLS_SERVER` to enable HLS server.

Other programming steps are same as RTSP.

In document as below, we will use RTSP as one basic sample to describe how to design one streaming server.

Beside of streaming server, in our SDK, we also support the RTSP client function to allow you to obtain the video and audio streams from the server side application, such as VLC Player and QuickTime Player and so on.

For RTMP, we support RTMP Web Portal too. RTMP portal allows you to send stream to RTMP server, such as Adobe Flash Media Server.

All protocols, you can find its sample from our SDK samples folder.

Programming Workflow of Broadcast Server Object

Programming Step	Related API Functions
Create Broadcast Server Object	`QCAP_CREATE_BROADCAST_RTSP_SERVER( …, &pServer )`
Register Callback Function	`QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK( pServer, … );` `QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK( pServer, … );`
Set Video / Audio Property	`QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer, … )` `QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY( pServer, … )`
Start Video / Audio Broadcast Server	`QCAP_START_BROADCAST_SERVER( pServer )`

Push Uncompression or Compression 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, … )`

Stop Video / Audio Broadcast Server	`QCAP_STOP_BROADCAST_SERVER( pServer )`
Delete Broadcast Server Object	`QCAP_DESTROY_BROADCAST_SERVER( pServer )`

Programming Workflow of Broadcast Client Object

Programming Step	Related API Functions
Create Broadcast Client Object	`QCAP_CREATE_BROADCAST_CLIENT ( …, &pClient, … )`
Register Callback Function	`QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK( pClient, … );` `QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK( pClient, … );`
Start Video / Audio Broadcast Client	`QCAP_START_BROADCAST_CLIENT( pClient )`
Stop Video / Audio Broadcast Client	`QCAP_STOP_BROADCAST_CLIENT( pClient )`
Delete Broadcast Client Object	`QCAP_DESTROY_BROADCAST_CLIENT( pClient )`

About RTMP of Adobe License

This software development kit contains software programming code and libraries related to RTMP of Adobe connections that is subject to your development use only. All the intellectual properties are held by individual license groups and/or the property holders. License and royalty fee should be charged by the license groups and/or property holders and paid by you if any of them is implemented into your commercial product. We will not be responsible for any illegal usage and/or any infringement of the rights of the individual owners of these intellectual properties.

Details of RTMP of Adobe could be found in the website:

http://www.adobe.com/products/eulas/

11.2 QCAP_CREATE_BROADCAST_RTSP_SERVER

```
QRESULT QCAP_CREATE_BROADCAST_RTSP_SERVER( UINT       iSvrNum,
                                           ULONG      nTotalSessions,
                                           PVOID  *   ppServer,
                                           CHAR   *   pszAccount = NULL,
                                           CHAR   *   pszPassword = NULL,
                                           ULONG      nNetworkPort_RTSP = 554,
                                           ULONG      nNetworkPort_RTSPOverHTTP = 0 )
```

Parameter

Name	Description
iSvrNum	**IN**, Specify broadcast server number to get server parameters, start from 0
nTotalSessions	**IN**, Specify the total number of session connections
ppServer	**OUT**, Handle of the broadcast server object
pszAccount	**IN**, Specify the server side account information to login, the default string is NULL
pszPassword	**IN**, Specify the server side password information to login, the default string is NULL
NetworkPort_RTSP	**IN**, Specify the port number that the server listens on for RTSP, the default port is 554
NetworkPort_RTSPOverHTTP	**IN**, Specify the port number that the server listens on for RTSP over HTTP requests, the default port is 8080. 0 is disable this function.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a broadcast server object.

iSvrNum can allow you to create multiple broadcast servers with different port.

nTotalSessions can create multiple sessions from one server.

Every session includes one video stream and one audio stream.

In the client side, user can use the URL, rtsp://account:password@ip:port/session#num.mpg to access it.

For example, rtsp://root:root@10.10.10.123:554/session0.mpg.

Example

```
QCAP_CREATE_BROADCAST_RTSP_SERVER( 0, 4, &pServer, “root”, “root”, 554, 0 )
```

11.3 QCAP_CREATE_BROADCAST_RTMP_SERVER

```
QRESULT QCAP_CREATE_BROADCAST_RTMP_SERVER( UINT       iSvrNum,
                                           ULONG      nTotalSessions,
                                           PVOID  *   ppServer,
                                           CHAR   *   pszAccount = NULL,
                                           CHAR   *   pszPassword = NULL,
                                           ULONG      nNetworkPort_RTMP = 1935,
                                           ULONG      nNetworkPort_RTMPOverHTTP = 0 )
```

Parameter

Name	Description
iSvrNum	**IN**, Specify broadcast RTMP server number to get server parameters, start from 0
nTotalSessions	**IN**, Specify the total number of session connections
ppServer	**OUT**, Handle of the broadcast server object
pszAccount	**IN**, Specify the server side account information to login, the default string is NULL
pszPassword	**IN**, Specify the server side password information to login, the default string is NULL
NetworkPort_RTMP	**IN**, Specify the port number that the server listens on for RTMP, the default port is 1935
NetworkPort_RTMPOverHTTP	**IN**, Specify the port number that the server listens on for RTMP over HTTP requests, the default port is 8080. 0 is disable this function.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a broadcast RTMP server object.

iSvrNum can allow you to create multiple broadcast RTMP servers with different port.

nTotalSessions can create multiple sessions from one server.

Every session includes one video stream and one audio stream.

In the client side, user can use the URL, http://localhost/NetReceiver.Example/ to access it.

Note

If users want to use RTMP Server function, please read Yuan's QCAP SDK RTMP Server Installation Guide ENG firstly.

The encoder audio format should set QCAP_ENCODER_FORMAT_AAC_ADTS.

Example

```
QCAP_CREATE_BROADCAST_RTMP_SERVER( 0, 4, &pServer, "root", "root", 1935, 0 );
```

11.4 QCAP_CREATE_BROADCAST_RTMP_WEB_PORTAL_SERVER

```
QRESULT QCAP_CREATE_BROADCAST_RTMP_WEB_PORTAL_SERVER( UINT     iSvrNum,
                                                      CHAR  *  pszURL,
                                                      PVOID *  ppServer )
```

Parameter

Name	Description
iSvrNum	**IN**, Specify broadcast server number to get server parameters, start from 0
pszURL	**IN**, In the client side, user can use the URL to access it. For example, "`rtmp://10.10.0.80/live/session0.mpg`"
ppServer	**OUT**, Handle of the broadcast server object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a broadcast RTMP web portal servers object.

iSvrNum can allow you to create multiple broadcast RTMP web portals.

One portal can send one video stream and one audio stream to one RTMP server’s URL mapping.

User can use the same URL, "rtmp://10.10.0.80/live/session0.mpg", to access live streams from RTMP server.

Example

```
QCAP_CREATE_BROADCAST_RTSP_RTMP_WEB_PORTAL_SERVER( 0, "rtmp:// 10.10.0.80/live/session0.mpg", &pServer )
```

11.5 QCAP_CREATE_BROADCAST_HLS_SERVER

```
QRESULT QCAP_CREATE_BROADCAST_HLS_SERVER(   UINT      iSvrNum,
                                            ULONG     nTotalSessions,
                                            PVOID  *  ppServer,
                                            CHAR   *  pszWebServerRootFolderPath,
                                            CHAR   *  pszSubFolderPath,
                                            ULONG     nSegmentDuration )
```

Parameter

Name	Description
iSvrNum	**IN**, Specify broadcast server number to get server parameters, start from 0
nTotalSessions	**IN**, Specify the total number of session connections
ppServer	**OUT**, Handle of the broadcast server object
pszWebServerRootFolderPath	**IN**, Specify the web server's root folder path
pszSubFolderPath	**IN**, Specify the sub folder path to save HLS files.
nSegmentDuration	**IN**, Set the segment duration, the default number is 3000 msec.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a broadcast HLS servers object.

iSvrNum can allow you to create multiple broadcast HLS servers.

Every session includes one video stream and one audio stream and user can use nSegmentDuration to control the create session time.

If nSegmentDuration set large, the player video will more stable.

User can use pszWebServerRootFolderPath and pszSubFolderPath parameters to set Web server root folder path with sub folder path.

Note

When user use below API, we suggest user must to set dSampleTime. Due to HLS need accurate Timestamp.

```
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
```

Example

```
QCAP_CREATE_BROADCAST_HLS_SERVER( 0, 4 /*4 CHANNELS*/, &pServer, "C:\\AppServ\\www\\" "hls\\" );

User can use these URL to access different HLS session as below:

CH01 URL -> http://xxx.xxx.xxx.xxx/hls/session0.m3u8 (HERE, xxx.xxx.xxx.xxx IS YOUR WEB SERVER ADDRESS)

CH02 URL -> http://xxx.xxx.xxx.xxx/hls/session1.m3u8

CH03 URL -> http://xxx.xxx.xxx.xxx/hls/session2.m3u8

CH04 URL -> http://xxx.xxx.xxx.xxx/hls/session3.m3u8
```

11.6 QCAP_DESTROY_BROADCAST_SERVER

QRESULT QCAP_DESTROY_BROADCAST_SERVER(PVOID pServer)

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Destroy broadcast server object to release its resource.

Example

```
QCAP_DESTROY_BROADCAST_SERVER( pServer );
```

11.7 QCAP_START_BROADCAST_SERVER

QRESULT QCAP_START_BROADCAST_SERVER(PVOID pServer)

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start broadcast server.

Example

```
QCAP_START_BROADCAST_SERVER( pServer );
```

11.8 QCAP_STOP_BROADCAST_SERVER

QRESULT QCAP_STOP_BROADCAST_SERVER(PVOID pServer)

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop broadcast server.

Example

```
QCAP_STOP_BROADCAST_SERVER( pServer );
```

11.9 QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY

```
QRESULT QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY( PVOID     pServer,
                                                  UINT      iSessionNum,
                                                  ULONG  *  pEncoderType,
                                                  ULONG  *  pEncoderFormat,
                                                  ULONG  *  pColorSpaceType,
                                                  ULONG  *  pWidth,
                                                  ULONG  *  pHeight,
                                                  ULONG  *  pFrameRate,
                                                  ULONG  *  pRecordMode,
                                                  ULONG  *  pQuality,
                                                  ULONG  *  pBitRate,
                                                  ULONG  *  pGOP,
                                                  ULONG  *  pAspectRatioX,
                                                  ULONG  *  pAspectRatioY,
                                                  HWND   *  pAttachedWindow = NULL,
                                                  BOOL   *  pThumbDraw = NULL,
                                                  BOOL   *  pMaintainAspectRatio = NULL,
                                                  DWORD  *  pFlags = NULL )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pEncoderType	**OUT**, Pointer to the storage of the encoder type for the i-th video stream session
pEncoderFormat	**OUT**, Pointer to the storage of the encoder format for the i-th video stream session
pColorSpaceType	**OUT**, Pointer to the storage of the color space type for the i-th video stream session
pWidth	**OUT**, Pointer to the storage of the horizontal width for the i-th video stream session
pHeight	**OUT**, Pointer to the storage of the vertical height for the i-th video stream session
pFrameRate	**OUT**, Pointer to the storage of the frame rate for the i-th video stream session
pRecordMode	**OUT**, Pointer to the storage of the record mode for the i-th video stream session
pQuality	**OUT**, Pointer to the storage of the quality for the i-th video stream session
pBitRate	**OUT**, Pointer to the storage of the bit rate for the i-th video stream session
pGOP	**OUT**, Pointer to the storage of the GOP size for the i-th video stream session
pAspectRatioX	**OUT**, Pointer to the storage of the aspect ratio X axle for the i-th video stream session
pAspectRatioY	**OUT**, Pointer to the storage of the aspect ratio Y axle for the i-th video stream session
pAttachedWindow	**OUT**, Pointer to the storage of window to show the i-th video stream session
pThumbDraw	**OUT**, Pointer to the enable flag of thumb draw renderer
pMaintainAspectRatio	**OUT**, Pointer to the enable flag of maintain aspect ratio

pFlags	**OUT**, Pointer to the enable flag of the i-th video stream session

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get parameters of the i-th video stream session in broadcast server.

Example

```
QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY(  pServer, 0, &nEncoderType, &nEncoderFormat, &nColorSpaceType,
                                           &nWidth, &nHeight, &nFrameRate, &nRecordMode, &nQuality,
                                           &nBitRate, &nGOP, &nAspectRatioX, &nAspectRatioY,
                                           &nAttachedWindow, &nThumbDraw, &nMaintainAspectRatio,
                                           &nFlags );
```

11.10 QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY

```
QRESULT QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( PVOID     pServer,
                                                  UINT      iSessionNum,
                                                  ULONG     nEncoderType,
                                                  ULONG     nEncoderFormat,
                                                  ULONG     nColorSpaceType,
                                                  ULONG     nWidth,
                                                  ULONG     nHeight,
                                                  ULONG     nFrameRate,
                                                  ULONG     nRecordMode,
                                                  ULONG     nQuality,
                                                  ULONG     nBitRate,
                                                  ULONG     nGOP,
                                                  ULONG     nAspectRatioX,
                                                  ULONG     nAspectRatioY,
                                                  HWND      hAttachedWindow = NULL,
                                                  BOOL      bThumbDraw = FALSE,
                                                  BOOL      bMaintainAspectRatio = FLASE,
                                                  DWORD     dwFlags = QCAP_BROADCAST_FLAG_FULL );
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_H264 = 0
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_RGB24 CAP_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
nWidth	**IN**, Specify the horizontal width for the i-th video stream session
nHeight	**IN**, Specify the vertical height for the i-th video stream session
nFrameRate	**IN**, Specify the frame rate for the i-th video stream session

nRecordMode	**IN**, Specify recording mode. Available values are QCAP_RECORD_MODE_VBR = 0 QCAP_RECORD_MODE_CBR = 1 QCAP_RECORD_MODE_ABR = 2
nQuality	**IN**, Specify video quality. It is used by VBR and ABR.
nBitRate	**IN**, Specify video bit rate. It is used by CBR and ABR. (Ex. 6Mbps is 6 x 1024 x 1024 bps)
nGOP	**IN**, Specify recording GOP size, from 0 ~ 255
nAspectRatioX	**IN**, Specify the aspect ratio X axle, 0 is turned off
nAspectRatioY	**IN**, Specify the aspect ratio Y axle, 0 is turned off
hAttachedWindow	**IN**, Specify the handle of the window to show the i-th video stream session
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio
nFlags	**IN**, Specify broadcasting mode. Available values are QCAP_BROADCAST_FLAG_FULL = 0x00000007 QCAP_BROADCAST_FLAG_NETWORK = 0x00000001 QCAP_BROADCAST_FLAG_ENCODE = 0x00000002 QCAP_BROADCAST_FLAG_DISPLAY = 0x00000004 QCAP_BROADCAST_FLAG_DECODE = 0x00000008

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set parameters of the i-th video stream session in broadcast server.

Example

```
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, FALSE, QCAP_BROADCAST_FLAG_FULL );
```

11.11 QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX

```
QRESULT QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY_EX( PVOID     pServer,
                                                     UINT      iSessionNum,
                                                     ULONG  *  pEncoderType,
                                                     ULONG  *  pEncoderFormat,
                                                     ULONG  *  pColorSpaceType,
                                                     ULONG  *  pWidth,
                                                     ULONG  *  pHeight,
                                                     ULONG  *  pFrameRate,
                                                     ULONG *   pRecordProfile,
                                                     ULONG *   pRecordLevel,
                                                     ULONG *   pRecordEntropy,
                                                     ULONG *   pRecordComplexity,
                                                     ULONG  *  pRecordMode,
                                                     ULONG  *  pQuality,
                                                     ULONG  *  pBitRate,
                                                     ULONG  *  pGOP,
                                                     ULONG  *  pBFrames,
                                                     ULONG  *  pAspectRatioX,
                                                     ULONG  *  pAspectRatioY,
                                                     HWND   *  pAttachedWindow = NULL,
                                                     BOOL   *  pThumbDraw = NULL,
                                                     BOOL   *  pMaintainAspectRatio = NULL,
```

```
DWORD  *  pFlags = NULL )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pEncoderType	**OUT**, Pointer to the storage of the encoder type for the i-th video stream session
pEncoderFormat	**OUT**, Pointer to the storage of the encoder format for the i-th video stream session
pColorSpaceType	**OUT**, Pointer to the storage of the color space type for the i-th video stream session
pWidth	**OUT**, Pointer to the storage of the horizontal width for the i-th video stream session
pHeight	**OUT**, Pointer to the storage of the vertical height for the i-th video stream session
pFrameRate	**OUT**, Pointer to the storage of the frame rate for the i-th video stream session
pRecordProfile	**OUT**, Pointer to the storage of the record profile for the i-th video stream session
pRecordLevel	**OUT**, Pointer to the storage of the record level for the i-th video stream session
pRecordEntropy	**OUT**, Pointer to the storage of the record entropy for the i-th video stream session
pRecordComplexity	**OUT**, Pointer to the storage of the record complexity for the i-th video stream session
pRecordMode	**OUT**, Pointer to the storage of the record mode for the i-th video stream session
pQuality	**OUT**, Pointer to the storage of the quality for the i-th video stream session
pBitRate	**OUT**, Pointer to the storage of the bit rate for the i-th video stream session
pGOP	**OUT**, Pointer to the storage of the GOP size for the i-th video stream session
pBFrames	**OUT**, Pointer to the storage of the B Frames

pAspectRatioX	**OUT**, Pointer to the storage of the aspect ratio X axle for the i-th video stream session
pAspectRatioY	**OUT**, Pointer to the storage of the aspect ratio Y axle for the i-th video stream session
pAttachedWindow	**OUT**, Pointer to the storage of window to show the i-th video stream session
pThumbDraw	**OUT**, Pointer to the enable flag of thumb draw renderer
pMaintainAspectRatio	**OUT**, Pointer to the enable flag of maintain aspect ratio
pFlags	**OUT**, Pointer to the enable flag of the i-th video stream session

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get additional parameters of the i-th video stream session in broadcast server.

Example

```
QCAP_GET_VIDEO_BROADCAST_SERVER_PROPERTY(  pServer, 0, &nEncoderType, &nEncoderFormat, &nColorSpaceType,
                                           &nWidth, &nHeight, &nFrameRate, &nRecordMode, &nQuality,
                                           &nBitRate, &nGOP, &nBFrames, &nAspectRatioX, &nAspectRatioY,
                                           &nAttachedWindow, &nThumbDraw, &nMaintainAspectRatio, nFlags );
```

11.12 QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX

```
QRESULT QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY_EX( PVOID     pServer,
                                                     UINT      iSessionNum,
                                                     ULONG     nEncoderType,
                                                     ULONG     nEncoderFormat,
                                                     ULONG     nColorSpaceType,
                                                     ULONG     nWidth,
                                                     ULONG     nHeight,
                                                     ULONG     nFrameRate,
                                                     ULONG     nRecordProfile,
                                                     ULONG     nRecordLevel,
                                                     ULONG     nRecordEntropy,
                                                     ULONG     nRecordComplexity,
                                                     ULONG     nRecordMode,
                                                     ULONG     nQuality,
                                                     ULONG     nBitRate,
                                                     ULONG     nGOP,
                                                     ULONG     nBFrames,
                                                     ULONG     nAspectRatioX,
                                                     ULONG     nAspectRatioY,
                                                     HWND      hAttachedWindow = NULL,
                                                     BOOL      bThumbDraw = FALSE,
                                                     BOOL      bMaintainAspectRatio = FLASE,
```

```
                                        DWORD      dwFlags = QCAP_BROADCAST_FLAG_FULL );
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_H264 = 0
nColorSpaceType	**IN**, Specify encoder color space type. Available values are 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
nWidth	**IN**, Specify the horizontal width for the i-th video stream session
nHeight	**IN**, Specify the vertical height for the i-th video stream session
nFrameRate	**IN**, Specify the frame rate for the i-th video stream session

nRecordProfile	**IN**, Specify recording profile
	QCAP_RECORD_PROFILE_BASELINE = 0, // DEFAULT
	QCAP_RECORD_PROFILE_MAIN = 1,
	QCAP_RECORD_PROFILE_HIGH = 2

nRecordLevel	**IN**, Specify recording level
	QCAP_RECORD_LEVEL_1 = 10,
	QCAP_RECORD_LEVEL_1B = 9,
	QCAP_RECORD_LEVEL_11 = 11,
	QCAP_RECORD_LEVEL_12 = 12,
	QCAP_RECORD_LEVEL_13 = 13,
	QCAP_RECORD_LEVEL_2 = 20,
	QCAP_RECORD_LEVEL_21 = 21,
	QCAP_RECORD_LEVEL_22 = 22,
	QCAP_RECORD_LEVEL_3 = 30,
	QCAP_RECORD_LEVEL_31 = 31,
	QCAP_RECORD_LEVEL_32 = 32,
	QCAP_RECORD_LEVEL_4 = 40,
	QCAP_RECORD_LEVEL_41 = 41, // DEFAULT
	QCAP_RECORD_LEVEL_42 = 42,
	QCAP_RECORD_LEVEL_50 = 50,
	QCAP_RECORD_LEVEL_51 = 51,
	QCAP_RECORD_LEVEL_52 = 52,

nRecordEntropy	**IN**, Specify recording entropy
	QCAP_RECORD_ENTROPY_CAVLC = 0, // DEFAULT
	QCAP_RECORD_ENTROPY_CABAC = 1,

nRecordComplexity	**IN**, Specify recording complexity
	QCAP_RECORD_COMPLEXITY_0 = 0, // BEST SPEED, DEFAULT
	QCAP_RECORD_COMPLEXITY_1 = 1,
	QCAP_RECORD_COMPLEXITY_2 = 2,
	QCAP_RECORD_COMPLEXITY_3 = 3,
	QCAP_RECORD_COMPLEXITY_4 = 4,
	QCAP_RECORD_COMPLEXITY_5 = 5,
	QCAP_RECORD_COMPLEXITY_6 = 6, // BEST QUALITY

nRecordMode	**IN**, Specify recording mode. Available values are QCAP_RECORD_MODE_VBR = 0 QCAP_RECORD_MODE_CBR = 1 QCAP_RECORD_MODE_ABR = 2
nQuality	**IN**, Specify video quality. It is used by VBR and ABR.
nBitRate	**IN**, Specify video bit rate. It is used by CBR and ABR. (Ex. 6Mbps is 6 x 1024 x 1024 bps)
nGOP	**IN**, Specify recording GOP size, from 0 ~ 255
nBFrames	**IN**, Specify recording B Frames
nAspectRatioX	**IN**, Specify the aspect ratio X axle, 0 is turned off
nAspectRatioY	**IN**, Specify the aspect ratio Y axle, 0 is turned off
hAttachedWindow	**IN**, Specify the handle of the window to show the i-th video stream session
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio

nFlags	**IN**, Specify broadcasting mode. Available values are QCAP_BROADCAST_FLAG_FULL = 0x00000007 QCAP_BROADCAST_FLAG_NETWORK = 0x00000001 QCAP_BROADCAST_FLAG_ENCODE = 0x00000002 QCAP_BROADCAST_FLAG_DISPLAY = 0x00000004 QCAP_BROADCAST_FLAG_DECODE = 0x00000008

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set additional parameters of the i-th video stream session in broadcast server.

Example

```
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, nBFrames,

                                           30, 4, 3, hAttachedWindow, FALSE, QCAP_BROADCAST_FLAG_FULL );
```

11.13 QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY

```
QRESULT QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY( PVOID     pServer,
                                                  UINT      iSessionNum,
                                                  ULONG  *  pEncoderType,
                                                  ULONG  *  pEncoderFormat,
                                                  ULONG  *  pChannels,
                                                  ULONG  *  pBitsPerSample,
                                                  ULONG  *  pSampleFrequency,
                                                  ULONG  *  pVolume )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pEncoderType	**OUT**, Pointer to the storage of the encoder type for the i-th audio stream session
pEncoderFormat	**OUT**, Pointer to the storage of the encoder format for the i-th video stream session
pChannels	**OUT**, Pointer to the storage of encoder channels for the i-th video stream session
pBitsPerSample	**OUT**, Pointer to the storage of encoder bits per sample for the i-th video stream session
pSampleFrequency	**OUT**, Pointer to the storage of encoder sample frequency for the i-th video stream session
pVolume	**OUT**, Pointer to the storage of audio renderer's volume for the i-th video stream session

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get parameters of the i-th audio stream session in broadcast server.

Example

```
QCAP_GET_AUDIO_BROADCAST_SERVER_PROPERTY( pServer, 0, &nEncoderType, &nEncoderFormat,
                                          &nChannels, &nBitsPerSample, &nSampleFrequency, &nVolume );
```

11.14 QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY

```
QRESULT QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY( PVOID     pServer,
                                                  UINT      iSessionNum,
                                                  ULONG     nEncoderType,
                                                  ULONG     nEncoderFormat,
                                                  ULONG     nChannels,
                                                  ULONG     nBitsPerSample,
                                                  ULONG     nSampleFrequency,
                                                  ULONG     nVolume )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
nEncoderType	**IN**, Specify encoder type. Available value are QCAP_ENCODER_TYPE_SOFTWARE = 0 QCAP_ENCODER_TYPE_HARDWARE = 1 for SC580 & SC590 QCAP_ENCODER_TYPE_INTEL_MEDIA_SDK = 2 QCAP_ENCODER_TYPE_NVIDIA_CUDA = 4
nEncoderFormat	**IN**, Specify encoder format. Available value are QCAP_ENCODER_FORMAT_PCM = 0 QCAP_ENCODER_FORMAT_AAC = 1 QCAP_ENCODER_FORMAT_AAC_ADTS = 2
nChannels	**IN**, Specify encoder channels for the i-th audio stream session

nBitsPerSample	**IN**, Specify encoder bits per sample for the i-th audio stream session
nSampleFrequency	**IN**, Specify encoder sample frequency for the i-th audio stream session
nVolume	**IN**, Specify the volume of audio renderer for the i-th audio stream session, from 0 ~ 100

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set parameters of the i-th audio stream session in broadcast server.

Example

```
QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY(  pServer, 0, QCAP_ENCODER_TYPE_SOFTWARE,

                                           QCAP_ENCODER_FORMAT_PCM, 2, 16, 48000, 50 );
```

11.15 QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER

```
QRESULT QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(   PVOID     pServer,
                                                                UINT      iSessionNum,
                                                                ULONG     nColorSpaceType,
                                                                ULONG     nWidth,
                                                                ULONG     nHeight,
                                                                BYTE   *  pFrameBuffer,
                                                                ULONG     nFrameBufferLen,
                                                                double    dSampleTime = 0.0 )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
nColorSpaceType	**IN**, Specify the input source buffer's color space type, Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
nWidth	**IN**, Specify the input source buffer's width
nHeight	**IN**, Specify the input source buffer's height
pFrameBuffer	**IN**, Specify the input source buffer
nFrameBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

* The performance of RGB type is slower than BGR in our color converter algorithm.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push un-compression video buffer into broadcasting recorder.

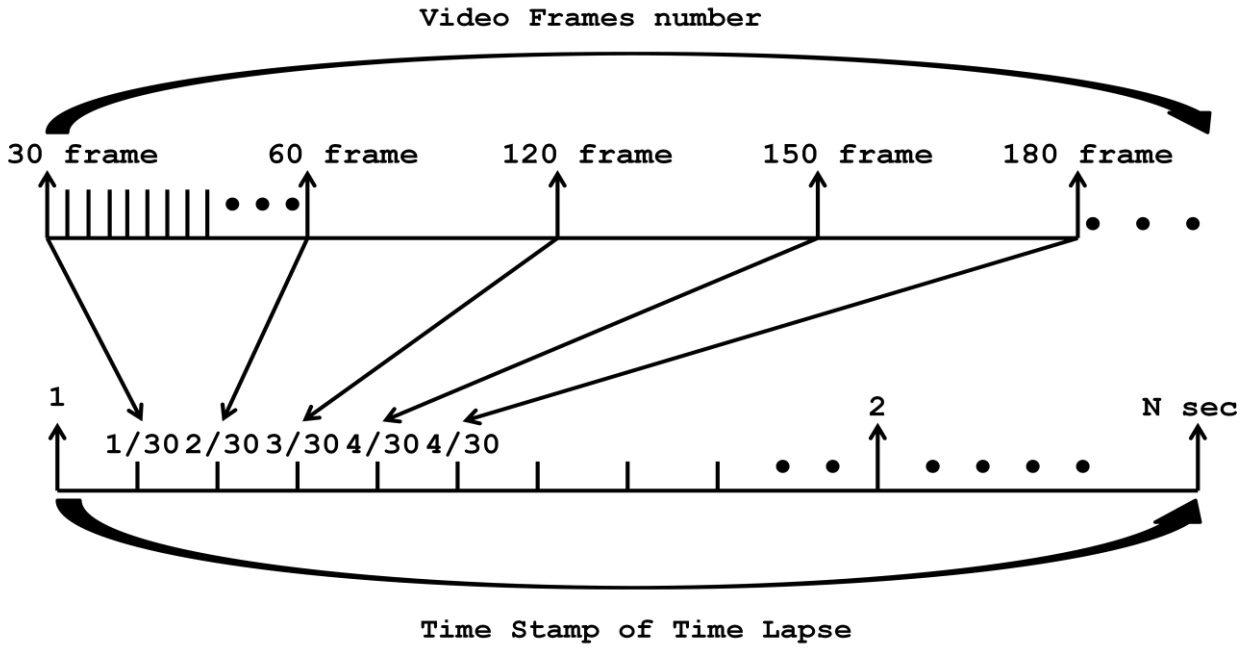
Server will send the buffer into the encoder and generate its bit-stream.

Finally, the bit-stream can be send out by internet.

If the width or the height of source frame are different from the target frame size.

QCAP will enable one auto video scaler to resize it before encoding.

If users want to get Time Lapse record result, user can use SampleTime to control time stamp of video. For instance the way as show in below.



Example

```
QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(pServer, 0, QCAP_COLORSPACE_TYEP_YUY2,

                                                    1280, 720, pFrameBuffer, 1280 * 720 * 2, 0 );
```

11.16 QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER

```
QRESULT QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(   PVOID     pServer,
                                                                UINT      iSessionNum,
                                                                BYTE   *  pFrameBuffer,
                                                                ULONG     nFrameBufferLen,
                                                                double    dSampleTime = 0.0 )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pFrameBuffer	**IN**, Specify the input source buffer
nFrameBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push un-compression audio buffer into broadcasting recorder.

Recorder will send the buffer into the encoder and generate its bit-stream.

Finally, the bit-stream can be send out by internet.

Note

The SampleTime of audio must same with SampleTime of video.

Example

```
QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 0, pFrameBuffer, 4096, 0 );
```

11.17 QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER

```
QRESULT QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER( PVOID     pServer,
                                                            UINT      iSessionNum,
                                                            BYTE   *  pStreamBuffer,
                                                            ULONG     nStreamBufferLen,
                                                            ULONG     bIsKeyFrame,
                                                            double    dSampleTime = 0.0 )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pStreamBuffer	**IN**, Specify the input source buffer
nStreamBufferLen	**IN**, Specify the input source buffer's size
bIsKeyFrame	**IN**, Specify the input source's frame whether it is key-frame or not
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

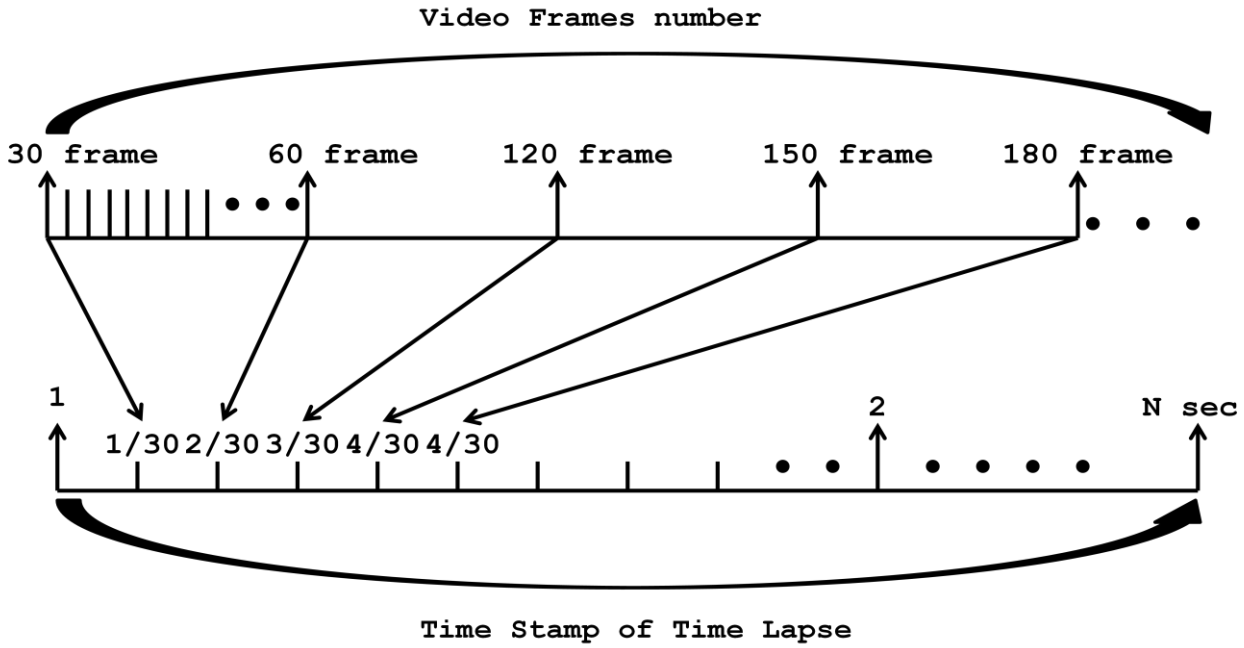
Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_DEVICE	Cannot open video or audio capture device
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to push compression stream buffer directly into broadcasting recorder.

The function will not enable encoder.

If users want to get Time Lapse result, user can use SampleTime to control time stamp of video. For instance the way as show in below.



Example

```
QCAP_SET_VIDEO_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer, 0, pStreamBuffer, 3072, TRUE, 0 );
```

11.18 QCAP_SET_AUDIO_BROADCAST_SERVER_COMPRESSION_BUFFER

```
QRESULT QCAP_SET_AUDIO_BROADCAST_SERVER_COMPRESSION_BUFFER( PVOID     pServer,
                                                            UINT      iSessionNum,
                                                            BYTE   *  pStreamBuffer,
                                                            ULONG     nStreamBufferLen,
                                                            double    dSampleTime = 0.0 )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pStreamBuffer	**IN**, Specify the input source buffer
nStreamBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to push compression stream buffer directly into broadcasting recorder.

The function will not enable encoder.

Note

The SampleTime of audio must same with SampleTime of video.

Example

```
QCAP_SET_AUDIO_BROADCAST_SERVER_COMPRESSION_BUFFER( pServer, 0, pStreamBuffer, 4096, 0 );
```

11.19 QCAP_SET_OSD_BROADCAST_SERVER_TEXT

```
QRESULT QCAP_SET_OSD_BROADCAST_SERVER_TEXT(    PVOID      pServer,
                                               UINT       iSessionNum,
                                               UINT       iOsdNum,
                                               INT        x,
                                               INT        y,
                                               INT        width,
                                               INT        height,
                                               CHAR *     pszString,
                                               CHAR *     pszFontFamilyName,
                                               ULONG      nFontStyle,
                                               ULONG      nFontSize,
                                               DWORD      dwFontColor,
                                               DWORD      dwBackgroundColor,
                                               ULONG      nTransparent,
                                               ULONG      nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0

iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output, if width is 0, QCAP will auto calculate width.
height	**IN**, Specify the vertical height of OSD output, if height is 0, QCAP will auto calculate height.
pszString	**IN**, Specify to display the text of OSD output
pszFontFamilyName	**IN**, Specify the font name used to display the text of OSD output
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
nFontSize	**IN**, Specify the font size used to display the text of OSD output
dwFontColor	**IN**, Specify the font color used to display the text of OSD output
dwBackgroundColor	**IN**, Specify the background color used to display the text of OSD output
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

When the users create a broadcast server object,

In the i-th video stream session we also support to show a text field objects used for on screen display of information.

If you want to get a text field object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_BROADCAST_SERVER_TEXT(  pServer, 0, 0, 0, 0, 0, 0, “CH01”, “Arial”,
                                     QCAP_FONT_STYLE_BOLD, 12, 0xFF000000,
                                     0xFFFFFFFF, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

11.20 QCAP_SET_OSD_BROADCAST_SERVER_PICTURE

```
QRESULT QCAP_SET_OSD_BROADCAST_SERVER_PICTURE(    PVOID      pServer,
                                                  UINT       iSessionNum,
                                                  UINT       iOsdNum,
                                                  INT        x,
                                                  INT        y,
                                                  INT        width,
                                                  INT        height,
                                                  CHAR *     pszFilePathName,
                                                  ULONG      nTransparent,
                                                  ULONG      nSequenceStyle =
                                                  QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output

width	**IN**, Specify the horizontal width of OSD output If width is 0, QCAP will use original picture width as default.
height	**IN**, Specify the vertical height of OSD output. If height is 0, QCAP will use original picture height as default.
pszFilePathName	**IN**, Specify the file name to display image OSD output, support a file extension of "BMP" as 24 or 32-bit, “JPG” and “PNG”
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

When the users create a broadcast server object,

In the i-th video stream session we also support to show a static image objects used for on screen display of information.

If you want to get a text field object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_BROADCAST_SERVER_PICTURE( pServer, 0, 0, 0, 0, 0, 0,
                                       “C:\\SAMPLE.PNG”,
                                       128,
                                       QCAP_SEQUENCE_STYLE_FOREMOST  );
```

11.21 QCAP_SET_OSD_BROADCAST_SERVER_BUFFER

```
QRESULT QCAP_SET_OSD_BROADCAST_SERVER_BUFFER(  PVOID      pServer,
                                               UINT       iSessionNum,
                                               UINT       iOsdNum,
                                               INT        x,
                                               INT        y,
                                               INT        width,
                                               INT        height,
                                               ULONG      nColorSpaceType,
                                               BYTE *     pFrameBuffer,
                                               ULONG      nFrameWidth,
                                               ULONG      nFrameHeight,
                                               ULONG      nFramePitch,
                                               ULONG      nTransparent
                                               ULONG      nSequenceStyle =
                                                          QCAP_SEQUENCE_STYLE_FOREMOST );
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output
height	**IN**, Specify the vertical height of OSD output
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer
nFrameWidth	**IN**, Specify the horizontal width of frame contained in a buffer
nFrameHeight	**IN**, Specify the vertical height of frame contained in a buffer
nFramePitch	**IN**, Specify the distance in bytes between the start of one scan line to the next. If it is 0, we will auto calculate it by width and color space format.

nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

When the users create a broadcast server object,

In the i-th video stream session we also support to show a frame buffer objects used for on screen display of information.

The function’s purpose is used to point directly to a valid frame buffer whose entire pixel data,

which is stored in available color space, and display it.

Beside, you also can adjust the width and height parameters to scale the OSD output.

Example

```
QCAP_SET_OSD_BROADCAST_SERVER_BUFFER( pServer, 0, 0, 0, 0, 1920, 1080,
                                      QCAP_COLORSPACE_TYEP_YUY2, pFrameBuffer,
                                      800, 600, 800 * 2, 128 );
```

11.22 QCAP_CREATE_BROADCAST_CLIENT

```
QRESULT QCAP_CREATE_BROADCAST_CLIENT (  UINT       iCliNum,
                                        CHAR *     pszURL,
                                        PVOID *    ppClient,
                                        HWND       hAttacheWindow = NULL,
                                        BOOL       bThumbDraw = FALSE,
                                        BOOL       bMaintainAspectRatio = FALSE )
```

Parameter

Name	Description
iCliNum	**IN**, Specify broadcast client number to get client parameters, start from 0
pszURL	**IN**, In the client side, user can use the URL to access it. For example rtsp://account:password@ip:port/session#num.mpg
ppClient	**OUT**, Handle of the client object
hAttacheWindow	**IN**, Handle of the window to show the preview video, the default string is NULL
bThumbDraw	**IN**, Enable/Disable the thumb draw renderer
bMaintainAspectRatio	**IN**, Enable/Disable the maintain aspect ratio

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a broadcast client object.

Example

```
QCAP_CREATE_BROADCAST_CLIENT( 0, "rtsp://root:root@127.0.0.1:554/session0.mpg", &pClient, hWindow, TRUE, );
```

11.23 QCAP_DESTROY_BROADCAST_CLIENT

```
QRESULT QCAP_DESTROY_BROADCAST_CLIENT( PVOID pClient )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Destroy broadcast client object to release its resource.

Example

```
QCAP_DESTROY_BROADCAST_CLIENT( pClient );
```

11.24 QCAP_START_BROADCAST_CLIENT

```
QRESULT QCAP_START_BROADCAST_CLIENT( PVOID pClient,
                                     ULONG nProtocol = QCAP_BROADCAST_PROTOCOL_UDP )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
nProtocol	**IN,** BROADCAST PROTOCOL, QCAP_BROADCAST_PROTOCOL_UDP = 0x00000000, QCAP_BROADCAST_PROTOCOL_TCP = 0x00000001, QCAP_BROADCAST_PROTOCOL_HTTP = 0x00000002, The default protocol is QCAP_BROADCAST_PROTOCOL_UDP.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start broadcast client.

Beside, user also can adjust the broadcast protocol to connect the video and audio stream.

Example

```
QCAP_START_BROADCAST_CLIENT( pClient );
```

11.25 QCAP_STOP_BROADCAST_CLIENT

QRESULT QCAP_STOP_BROADCAST_CLIENT(PVOID pClient)

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop broadcast client.

Example

```
QCAP_STOP_BROADCAST_CLIENT( pClient );
```

11.26 QCAP_GET_BROADCAST_CLIENT_AUDIO_VOLUME

```
QRESULT QCAP_GET_BROADCAST_CLIENT_AUDIO_VOLUME( PVOID pClient, ULONG * pVolume )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pVolume	**OUT**, Pointer to the storage of audio volume

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get audio volume of broadcast client.

Example

```
QCAP_GET_BROADCAST_CLIENT_AUDIO_VOLUME( pClient, &nVolume );
```

11.27 QCAP_SET_BROADCAST_CLIENT_AUDIO_VOLUME

```
QRESULT QCAP_SET_BROADCAST_CLIENT_AUDIO_VOLUME( PVOID pClient, ULONG  nVolume )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
nVolume	**IN**, Audio volume, from 0 ~ 100, 0 is mute

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio volume of broadcast client.

Example

```
QCAP_SET_BROADCAST_CLIENT_AUDIO_VOLUME( pClient, 50 );
```

11.28 QCAP_START_BROADCAST_CLIENT_RECORD

```
QRESULT QCAP_START_BROADCAST_CLIENT_RECORD( PVOID pClient, CHAR *  pszFilePathName )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pszFilePathName	**IN**, Specify the file name of storage for recording. Recording file format is AVI, if extension name is AVI. Recording file format is MP4, if extension name is MP4 Recording file format is ASF, if extension name is ASF Recording file format is FLV, if extension name is FLV Recording file format is TS, if extension name is TS

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start recording of broadcast client.

Example

```
QCAP_START_BROADCAST_CLIENT_RECORD( pClient, "D:\\XXX.AVI" );
```

11.29 QCAP_STOP_BROADCAST_CLIENT_RECORD

```
QRESULT QCAP_STOP_BROADCAST_CLIENT_RECORD( PVOID pClient, BOOL bIsAsync = TRUE, ULONG nMilliseconds = 0 )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
bIsAsync	**IN**, Synchronized stop recording flag
nMilliseconds	**IN**, Time-out interval, in milliseconds. The parameter is used at sync mode only. If nMilliseconds is zero, the function returns immediately. If it is INFINITE, the function's time-out interval never elapses.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop recording of broadcast client.

Example

```
QCAP_STOP_BROADCAST_CLIENT_RECORD( pClient, 0 );
```

11.30 QCAP_RESUME_BROADCAST_CLIENT_RECORD

```
QRESULT QCAP_RESUME_BROADCAST_CLIENT_RECORD( PVOID pClient )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to resume recording of broadcast client.

Example

```
QCAP_RESUME_BROADCAST_CLIENT_RECORD( pClient );
```

11.31 QCAP_PAUSE_BROADCAST_CLIENT_RECORD

```
QRESULT QCAP_PAUSE_BROADCAST_CLIENT_RECORD( PVOID pClient )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to pause recording of broadcast client.

Example

```
QCAP_PAUSE_BROADCAST_CLIENT_RECORD( pClient );
```

11.32 QCAP_SET_OSD_BROADCAST_CLIENT_TEXT

```
QRESULT QCAP_SET_OSD_BROADCAST_CLIENT_TEXT( PVOID     pClient,
                                            UINT      iOsdNum,
                                            INT       x,
                                            INT       y,
                                            INT       w,
                                            INT       h,
                                            CHAR *    pszString,
                                            CHAR *    pszFontFamilyName,
                                            ULONG     nFontStyle,
                                            ULONG     nFontSize,
                                            DWORD     dwFontColor,
                                            DWORD     dwBackgroundColor,
                                            ULONG     nTransparent,
                                            ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output

y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output, if width is 0, QCAP will auto calculate width.
height	**IN**, Specify the vertical height of OSD output, if height is 0, QCAP will auto calculate height.
pszString	**IN**, Specify to display the text of OSD output
pszFontFamilyName	**IN**, Specify the font name used to display the text of OSD output

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`

nFontSize	**IN**, Specify the font size used to display the text of OSD output
dwFontColor	**IN**, Specify the font color used to display the text of OSD output
dwBackgroundColor	**IN**, Specify the background color used to display the text of OSD output
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a text field objects used for on broadcast client of information.

If you want to get a text field object auto calculated actual width and height value; you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Note

Color space of FontColor is ABGR, don’t forgot to set its alpha value.

Color space of BackgroundColor is ABGR, don’t forgot to set its alpha value.

Transparent is one global parameter for all colors.

Example

```
QCAP_SET_OSD_BROADCAST_CLIENT_TEXT(  pClient, 0, 0, 0, 0, 0, “broadcast_client”, “Arial”,
                                     QCAP_FONT_STYLE_BOLD, 12, 0xFF000000,
                                     0xFFFFFFFF, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

11.33 QCAP_SET_OSD_BROADCAST_CLIENT_PICTURE

```
QRESULT QCAP_SET_OSD_BROADCAST_CLIENT_PICTURE( PVOID     pClient,
                                               UINT      iOsdNum,
                                               INT       x,
                                               INT       y,
                                               INT       width,
                                               INT       height,
                                               CHAR *    pszFilePathName,
                                               ULONG     nTransparent
                                               ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output If width is 0, QCAP will use original picture width as default.
height	**IN**, Specify the vertical height of OSD output. If height is 0, QCAP will use original picture height as default.

pszFilePathName	**IN**, Specify the file name to display image OSD output, support a file extension of "BMP" as 24 or 32-bit, “JPG” and “PNG”
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255
nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a static image objects used for on broadcast client of information.

If you want to get a static image object auto calculated actual width and height value;

you should set height and width parameter as 0.

Beside, you also can adjust the OSD sequence style to video-streams recording the OSD output.

Example

```
QCAP_SET_OSD_BROADCAST_CLIENT_PICTURE(  pClient, 0, 0, 0, 0, 0, “C:\\SAMPLE.PNG”, 128,
                                        QCAP_SEQUENCE_STYLE_FOREMOST );
```

11.34 QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER

```
QRESULT QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER(  PVOID     pClient,
                                               UINT      iOsdNum,
                                               INT       x,
                                               INT       y,
                                               INT       width,
                                               INT       height,
                                               ULONG     nColorSpaceType,
                                               BYTE *    pFrameBuffer,
                                               ULONG     nFrameWidth,
                                               ULONG     nFrameHeight,
                                               ULONG     nFramePitch,
                                               ULONG     nTransparent
                                               ULONG     nSequenceStyle = QCAP_SEQUENCE_STYLE_FOREMOST )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
iOsdNum	**IN**, Specify the i-th layer number of OSD output
x	**IN**, Specify the x-coordinate of the upper-left corner of OSD output
y	**IN**, Specify the y-coordinate of the upper-left corner of OSD output
width	**IN**, Specify the horizontal width of OSD output
height	**IN**, Specify the vertical height of OSD output
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer
nFrameWidth	**IN**, Specify the horizontal width of frame contained in a buffer
nFrameHeight	**IN**, Specify the vertical height of frame contained in a buffer
nFramePitch	**IN**, Specify the distance in bytes between the start of one scan line to the next. If it is 0, we will auto calculate it by width and color space format.
nTransparent	**IN**, Specify the transparent ratio of whole OSD output, the range of 0 to 255

nSequenceStyle	**IN**, Specify OSD sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002

* The performance of BGR type is slower than RGB in our alpha blending algorithm.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create a frame buffer objects used for on broadcast client of information.

The function’s purpose is used to point directly to a valid frame buffer whose entire pixel data,

which is stored in available color space, and display it.

Beside, you also can adjust the width and height parameters to scale the OSD output

and OSD sequence style to video-streams video-recording the OSD output.

Example

```
QCAP_SET_OSD_BROADCAST_CLIENT_BUFFER( pClient, 0, 0, 0, 1920, 1080,
                                      QCAP_COLORSPACE_TYEP_YUY2,
                                      pFrameBuffer, 800, 600, 800 * 2, 128, QCAP_SEQUENCE_STYLE_FOREMOST );
```

11.35 An Broadcasting Software Programming Guides

```
// BROADCASTING SERVER EXAMPLE:
//
// STEP#01
//
QCAP_CREATE_BROADCAST_RTSP_SERVER( 0, 4 /*4 CHANNELS*/, &pServer, "root", "root", 554 );

//        CH01 URL -> rtsp://root:root@127.0.0.1:554/session0.mpg
//
//        CH02 URL -> rtsp://root:root@127.0.0.1:554/session1.mpg
//
//        CH03 URL -> rtsp://root:root@127.0.0.1:554/session2.mpg
//
//        CH04 URL -> rtsp://root:root@127.0.0.1:554/session3.mpg

// STEP#02
//
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_VIDEO_BROADCAST_SERVER_PROPERTY(   pServer,
                                        2 /*CH03*/,
                                        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,
                                        3 /*CH04*/,
                                        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,
                                        0 );

QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY(   pServer,
                                        1 /*CH02*/,
                                        QCAP_ENCODER_TYPE_SOFTWARE,
                                        QCAP_ENCODER_FORMAT_AAC,
                                        2,
                                        16,
                                        48000,
                                        0 );

QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY(   pServer,
                                        2 /*CH03*/,
                                        QCAP_ENCODER_TYPE_SOFTWARE,
                                        QCAP_ENCODER_FORMAT_AAC,
                                        2,
                                        16,
                                        48000,
                                        0 );

QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY(   pServer,
                                        3 /*CH04*/,
                                        QCAP_ENCODER_TYPE_SOFTWARE,
                                        QCAP_ENCODER_FORMAT_AAC,
                                        2,
                                        16,
                                        48000,
```

```
                                                0 );
// STEP#03
//
QCAP_START_BROADCAST_SERVER( pServer );


// STEP#04
//
QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(pServer,
                                                     0 /*CH01*/,
                                                     QCAP_COLORSPACE_TYEP_YUY2,
                                                     1920,
                                                     1080,
                                                     pFrameBuffer,
                                                     nFrameBufferLen );

QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(pServer,
                                                     1 /*CH02*/,
                                                     QCAP_COLORSPACE_TYEP_YUY2,
                                                     1920,
                                                     1080,
                                                     pFrameBuffer,
                                                     nFrameBufferLen );

QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(pServer,
                                                     2 /*CH03*/,
                                                     QCAP_COLORSPACE_TYEP_YUY2,
                                                     1920,
                                                     1080,
                                                     pFrameBuffer,
                                                     nFrameBufferLen );
```

```
QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(pServer,
                                                     3 /*CH04*/,
                                                     QCAP_COLORSPACE_TYEP_YUY2,
                                                     1920,
                                                     1080,
                                                     pFrameBuffer,
                                                     nFrameBufferLen );

QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER(pServer, 0 /*CH01*/, pFrameBuffer, nFrameBufferLen );

QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 1 /*CH02*/, pFrameBuffer, nFrameBufferLen );

QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 2 /*CH03*/, pFrameBuffer, nFrameBufferLen );

QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( pServer, 3 /*CH04*/, pFrameBuffer, nFrameBufferLen );

// STEP#05
//
QCAP_STOP_BROADCAST_SERVER( pServer );

// STEP#06
//
QCAP_DRSTROY_BROADCAST_SERVER( pServer );
```

12 Animation Function API

12.1 QCAP_CREATE_ANIMATION_CLIP

```
QRESULT QCAP_CREATE_ANIMATION_CLIP(  CHAR  *   pszAniFileName,
                                     PVOID *   ppClip,
                                     ULONG *   pTotalFrames,
                                     ULONG *   pTotalSprites,
                                     ULONG *   pColorSpaceType,
                                     ULONG *   pWidth,
                                     ULONG *   pHeight )
```

Parameter

Name	Description
pszAniFileName	**IN**, Specify file name of animation
ppClip	**OUT**, Pointer to the storage of the Clip
pTotalFrames	**OUT**, Pointer to the storage of the total frames
pColorSpaceType	**OUT**, Pointer to the storage of animation clip color space type
pWidth	**OUT**, Pointer to the storage of animation clip width
pHeight	**OUT**, Pointer to the storage of animation clip height

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create animation clip, the parameters, pTotalFrames, pColorSpaceType, pWidth and pHeight are definded in XML File.

Example

```
QCAP_CREATE_ANIMATION_CLIP( "EXAMPLE.COOL.QUAD.XML", &pClip, &nFrames, &nSprites,
                            &nColorSpaceType, &nWidth, &nHeight );
```

12.2 QCAP_DESTROY_ANIMATION_CLIP

```
QRESULT QCAP_DESTROY_ANIMATION_CLIP( PVOID pClip )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Destroy animation clip to release its resource.

Note

If you have use `QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE`,

you must to destroy the Animation clip before device destroy or to set source to NULL.

Example

```
QCAP_DESTROY_ANIMATION_CLIP( pClip );
```

12.3 QCAP_START_ANIMATION_CLIP

```
QRESULT QCAP_START_ANIMATION_CLIP( PVOID pClip )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start animation clip.

Example

```
QCAP_START_ANIMATION_CLIP( pClip );
```

12.4 QCAP_STOP_ANIMATION_CLIP

QRESULT QCAP_STOP_ANIMATION_CLIP(PVOID pClip)

Parameter

Name	Description
pClip	**IN**, Handle of the clip object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop animation clip.

Example

```
QCAP_STOP_ANIMATION_CLIP( pClip );
```

12.5 QCAP_STEP_ANIMATION_CLIP

```
QRESULT QCAP_STEP_ANIMATION_CLIP(  PVOID      pClip,
                                   UINT  *    pFrameNum,
                                   BYTE  * *  ppFrameBuffer,
                                   ULONG *    pFrameBufferLen,
                                   BOOL       bClearBackground = TRUE )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
pFrameNum	**OUT**, Pointer to the storage of the frame num
ppFrameBuffer	**OUT**, Pointer to the storage of the frame buffer
pFrameBufferLen	**OUT**, Pointer to the storage of the frame buffer len
bClearBackground	**IN**, Enable/Disable the Clear Background

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to execution animation clip by continuous step by step.

Every time, the function is called, the Animation clip will go to next frame and output current frame result as show in Fig.2.

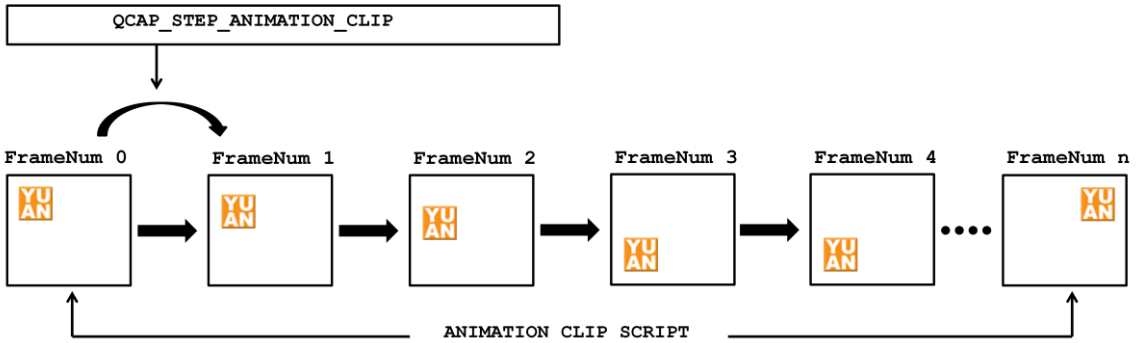


Fig.2

Example

```
QCAP_STEP_ANIMATION_CLIP ( pClip, &pFrameNum, &pFrameBuffer, &nFrameBufferLen, TRUE );
```

12.6 QCAP_SEEK_ANIMATION_CLIP

```
QRESULT QCAP_SEEK_ANIMATION_CLIP(  PVOID      pClip,
                                   UINT       iFrameNum,
                                   BYTE  * *  ppFrameBuffer,
                                   ULONG *    pFrameBufferLen,
                                   BOOL       bClearBackground = TRUE )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
iFrameNum	**IN**, Specify the number of frame
ppFrameBuffer	**OUT**, Pointer to the input source buffer
pFrameBufferLen	**OUT**, Pointer to the input source buffer's size
bClearBackground	**IN**, Enable/Disable the Clear Background

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to execution animation clip by seek.

Every time, the function is called, the Animation clip can jump any frame and output current frame result as show in Fig.3.

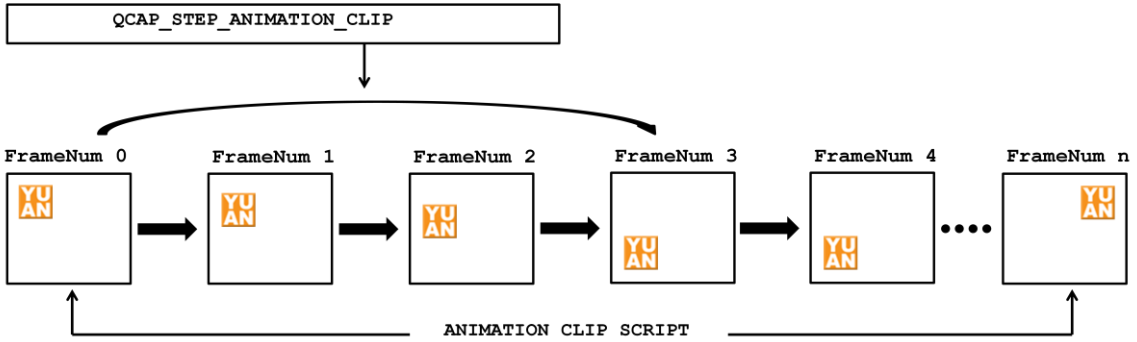


Fig.3

Example

```
QCAP_SEEK_ANIMATION_CLIP ( pClip, iFrameNum, &pFrameBuffer, &nFrameBufferLen, TRUE );
```

12.7 QCAP_SET_ANIMATION_CLIP_SPRITE_PICTURE

```
QRESULT QCAP_SET_ANIMATION_CLIP_SPRITE_PICTURE( PVOID      pClip,
                                                UINT       iSpriteNum,
                                                CHAR *     pszFilePathName )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
iSpriteNum	**IN**, Specify the number of channel sprite, start from 0
pszFilePathName	**IN**, Specify the file name to store BMP, JPG, or PNG data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to set a picture on animation clip’s sprite.

Example

```
QCAP_SET_ANIMATION_CLIP_SPRITE_PICTURE( pClip, 0, “C:\\PICTURE1.BMP” );
```

12.8 QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER

```
QRESULT QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER(  PVOID      pClip,
                                                UINT       iSpriteNum,
                                                ULONG      nColorSpaceType,
                                                ULONG      nWidth,
                                                ULONG      nHight,
                                                BYTE *     pFrameBuffer,
                                                ULONG      nFrameBufferLen,
                                                BOOL       bCloneCopy = FALSE )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
iSpriteNum	**IN**, Specify the number of channel sprite, start from 0
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
nWidth	**IN**, Specify the horizontal width of frame contained in a buffer

nHeight	**IN**, Specify the vertical height of frame contained in a buffer
pFrameBuffer	**IN**, Specify the input source buffer
nFrameBufferLen	**IN**, Specify the input source buffer's size
bCloneCopy	**IN**, Enable/Disable the Clone copy

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to set each channels buffer of animation clip.

Developer can call it to update sprites buffer.

Example

```
QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER( pClip, 0, QCAP_COLORSPACE_TYEP_YUY2, FALSE );
```

12.9 QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX

```
QRESULT QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX( PVOID       pClip,
                                                  UINT        iSpriteNum,
                                                  ULONG       nColorSpaceType,
                                                  ULONG       nWidth,
                                                  ULONG       nHight,
                                                  BYTE *      pFrameBuffer,
                                                  ULONG       nFrameBufferLen,
                                                  ULONG       nCropX,
                                                  ULONG       nCropY,
                                                  ULONG       nCropW,
                                                  ULONG       nCropH,
                                                  BOOL        bCloneCopy = FALSE )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
iSpriteNum	**IN**, Specify the number of channel sprite, start from 0
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24

	QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32 QCAP_COLORSPACE_TYEP_H264
nWidth	**IN**, Specify the horizontal width of frame contained in a buffer
nHeight	**IN**, Specify the vertical height of frame contained in a buffer
pFrameBuffer	**IN**, Specify the input source buffer
nFrameBufferLen	**IN**, Specify the input source buffer's size
nCropW	**IN**, Specify the horizontal width of crop
nCropH	**IN**, Specify the vertical height of crop
nDstW	**IN**, Specify the horizontal width of current recording down scale
nDstH	**IN**, Specify the vertical height of crop
bCloneCopy	**IN**, Enable/Disable the Clone copy

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to set crop parameters of each channels buffer of animation clip.

Developer can call it to update sprites buffer.

Example

```
QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER_EX( pClip, 0, QCAP_COLORSPACE_TYEP_YUY2, 10, 40, 1900, 1000, FALSE );
```

12.10 QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE

```
QRESULT QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE ( PVOID      pClip,
                                                UINT       iSpriteNum,
                                                PVOID      pDevice,
                                                ULONG      nSequenceStyle,
                                                BOOL       bCloneCopy = FALSE )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
iSpriteNum	**IN**, Specify the number of channel sprite, start from 0
pDevice	**IN**, Handle of the capture card object
nSequenceStyle	**IN**, Specify sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002
bCloneCopy	**IN**, Enable/Disable the Clone copy

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to set each channels source of animation clip.

It is one helper function, it can help user to use complex QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER function then to one attach live device channel on it, when device's preview date coming, the QCAP_SET_ANIMATION_CLIP_SPRITE_BUFFER will be called AUTO.

Example

```
QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE( pClip, 0, QCAP_COLORSPACE_TYEP_YUY2, FALSE );
```

12.11 QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX

```
QRESULT QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX (    PVOID       pClip,
                                                      UINT        iSpriteNum,
                                                      PVOID       pDevice,
                                                      ULONG       nCropX,
                                                      ULONG       nCropY,
                                                      ULONG       nCropW,
                                                      ULONG       nCropH,
                                                      ULONG       nSequenceStyle =
                                                                  QCAP_SEQUENCE_STYLE_FOREMOST,
                                                      BOOL        bCloneCopy = FALSE )
```

Parameter

Name	Description
pClip	**IN**, Handle of the clip object
iSpriteNum	**IN**, Specify the number of channel sprite, start from 0
pDevice	**IN**, Handle of the capture card object
nCropW	**IN**, Specify the horizontal width of crop
nCropH	**IN**, Specify the vertical height of crop
nDstW	**IN**, Specify the horizontal width of current recording down scale
nDstH	**IN**, Specify the vertical height of crop

nSequenceStyle	**IN**, Specify sequence style type. Available values are QCAP_SEQUENCE_STYLE_FOREMOST = 0x00000000 QCAP_SEQUENCE_STYLE_BEFORE_ENCODE = 0x00000001 QCAP_SEQUENCE_STYLE_AFTERMOST = 0x00000002
bCloneCopy	**IN**, Enable/Disable the Clone copy

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
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_GENERAL	Fail due to some errors

Introduction

User can use this function to set crop parameters of each channels source of animation clip.

Example

```
QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE_EX( pClip, 0, 10, 40, 1900, 1000, QCAP_COLORSPACE_TYEP_YUY2, FALSE );
```

12.12 An Animation Clip Software Programming Guides

```
// ANIMATION CLIP EXAMPLE:
// STEP#01
//
// QCAP_CREATE_ANIMATION_CLIP( "EXAMPLE.FADEOUT.XML",
                               &m_pClip,
                               &m_nClipFrames,
                               &m_nClipSprites,
                               &m_nClipColorSpaceType,
                               &m_nClipWidth,
                               &m_nClipHeight );

// STEP#02
//
// QCAP_SET_ANIMATION_CLIP_SPRITE_SOURCE( m_pClip,
                                          0,
                                          m_pDevices,
                                          QCAP_COLORSPACE_TYEP_YUY2,
                                          FALSE );

// STEP#03
//
// QCAP_SRART_ANIMATION_CLIP( m_pClip )
```

```
// STEP#04
//
// QCAP_STEP_ANIMATION_CLIP( m_pClip,
                             &iFrameNum,
                             &pClipFrameBuffer,
                             &nClipFrameBufferLen,
                             TRUE );

// STEP#05
//
// QCAP_STOP_ANIMATION_CLIP( m_pClip );

// STEP#06
//
// QCAP_DESTROY_ANIMATION_CLIP( m_pClip );
```

13 Virtual Camera Function API

13.1 QCAP_CREATE_VIRTUAL_CAMERA

```
QRESULT QCAP_CREATE_VIRTUAL_CAMERA(  UINT      iCamNum,
                                     PVOID *   ppCamera )
```

Parameter

Name	Description
iCamNum	**IN**, Specify the camera num
ppCamera	**OUT**, Handle of the camera object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to create one virtual camera object.

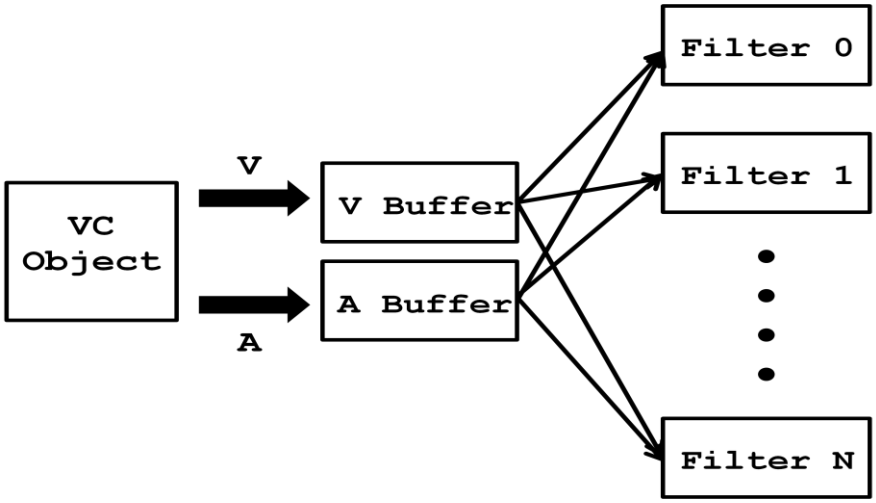
Virtual camera can allow you to create one share mapping of video frame buffer to another 3rd party software, such as QQ or Skype.

Customer wants to use virtual camera need register the VIRCAM.X86.AX filter into your system,

we call it as Custom Virtual Camera Filter.

You can find it from our QCAP\TOOLS folder.

The virtual camera object likes one streaming sender and the Custom Virtual Camera Filter likes one streaming receiver.



Example

```
QCAP_CREATE_VIRTUAL_CAMERA( 0, &pCamera );
```

13.2 QCAP_DESTROY_VIRTUAL_CAMERA

```
QRESULT QCAP_DESTROY_VIRTUAL_CAMERA( PVOID pCamera )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

Destroy camera object to release its resource.

Example

```
QCAP_DESTROY_VIRTUAL_CAMERA( pCamera );
```

13.3 QCAP_GET_VIDEO_VIRTUAL_CAMERA_PROPERTY

```
QRESULT QCAP_GET_VIDEO_VIRTUAL_CAMERA_PROPERTY (  PVOID     pCamera,
                                                  ULONG *   pColorSpaceType,
                                                  ULONG *   pWidth,
                                                  ULONG *   pHeight,
                                                  ULONG *   pFrameRate )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object
pColorSpaceType	**OUT**, Pointer to the storage of current color space type
pWidth	**OUT**, Pointer to the storage of current video width
pHeight	**OUT**, Pointer to the storage of current video height
pFrameRate	**OUT**, Pointer to the storage of current frame rate

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get the video format of virtual camera.

Example

```
QCAP_GET_VIDEO_VIRTUAL_CAMERA_PROPERTY( pCamera, &nColorSpaceType, &nWidth, &nHeight, &nFrameRate );
```

13.4 QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY

```
QRESULT QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY (  PVOID     pCamera,
                                                  ULONG     nColorSpaceType,
                                                  ULONG     nWidth,
                                                  ULONG     nHeight,
                                                  ULONG     nFrameRate )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object
nColorSpaceType	**IN**, Specify encoder color space type. Available values are 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
nWidth	**IN**, Specify encoder width.
nHeight	**IN**, Specify encoder height.

nFrameRate	**IN**, Specify encoder frame rate.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set video format of virtual camera.

It will change the output format of Virtual Camera Filter’s video output pin.

Example

```
QCAP_GET_VIDEO_VIRTUAL_CAMERA_PROPERTY( pCamera, QCAP_COLORSPACE_TYEP_YV12,
                                        nWidth, nHeight,
                                        nFrameRate );
```

13.5 QCAP_GET_AUDIO_VIRTUAL_CAMERA_PROPERTY

```
QRESULT QCAP_GET_AUDIO_VIRTUAL_CAMERA_PROPERTY (  PVOID     pCamera,
                                                  ULONG *   pChannels,
                                                  ULONG *   pBitPerSample,
                                                  ULONG *   pSampleFrequency )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object
pChannels	**OUT**, Pointer to the storage of current audio encoder channels
pBitsPerSample	**OUT**, Pointer to the storage of current audio encoder bits per sample
pSampleFrequency	**OUT**, Pointer to the storage of current audio encoder sample frequency

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to get audio format of virtual camera.

Example

```
QCAP_GET_AUDIO_VIRTUAL_CAMERA_PROPERTY( pCamera, &nChannels, &nBitsPerSample, &nSampleFrequency );
```

13.6 QCAP_SET_AUDIO_VIRTUAL_CAMERA_PROPERTY

```
QRESULT QCAP_SET_AUDIO_VIRTUAL_CAMERA_PROPERTY(    PVOID     pCamera,
                                                   ULONG     nChannels,
                                                   ULONG     nBitsPerSample,
                                                   ULONG     nSampleFrequency )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object
nChannels	**IN**, Specify encoder channels.
nBitsPerSample	**IN**, Specify encoder bits per sample.
nSampleFrequency	**IN**, Specify encoder sample frequency.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to set audio format of virtual camera.

It will change the output format of Virtual Camera Filter’s audio output pin.

Example

```
QCAP_SET_VIDEO_VIRTUAL_CAMERA_PROPERTY( pCamera, nChannels, nBitsPerSample, nSampleFrequency );
```

13.7 QCAP_SET_VIDEO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER

```
QRESULT QCAP_SET_VIDEO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( PVOID     pCamera,
                                                            ULONG     nColorSpaceType,
                                                            ULONG     nWidth,
                                                            ULONG     nHeight,
                                                            BYTE *    pFrameBuffer,
                                                            ULONG     nFrameBufferLen,
                                                            double    dSampleTime = 0.0 )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camer object
nColorSpaceType	**IN**, Specify encoder color space type. Available values are QCAP_COLORSPACE_TYEP_UYVY QCAP_COLORSPACE_TYEP_I420 QCAP_COLORSPACE_TYEP_YUY2 QCAP_COLORSPACE_TYEP_YV12 QCAP_COLORSPACE_TYEP_RGB24 QCAP_COLORSPACE_TYEP_BGR24 QCAP_COLORSPACE_TYEP_ARGB32 QCAP_COLORSPACE_TYEP_ABGR32
nWidth	**IN**, Specify video width
nHeight	**IN**, Specify video height
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer

nFrameBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

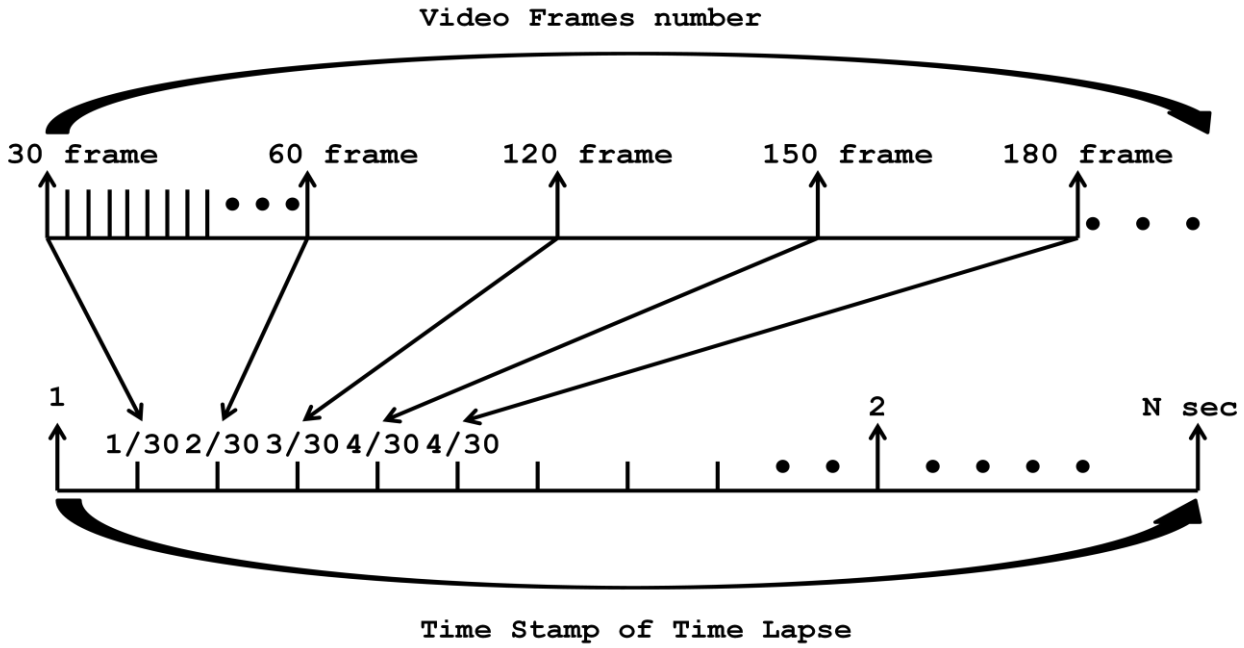
Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to push un-compression video buffer into virtual camera's share memory.

If users want to get Time Lapse result, user can use SampleTime to control time stamp of video. For instance the way as show in below.



Example

```
QCAP_SET_VIDEO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( pCamera, QCAP_COLORSPACE_TYEP_YUY2, 1280, 720,

                                                    &nFrameBuffer, 1280 * 720 * 2, 0 );
```

13.8 QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER

```
QRESULT QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( PVOID     pCamera,
                                                            BYTE *    pFrameBuffer,
                                                            ULONG     nFrameBufferLen,
                                                            double    dSampleTime = 0.0 )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camer object
pFrameBuffer	**IN**, Specify a raw data of frame contained in a buffer
nFrameBufferLen	**IN**, Specify the input source buffer's size
dSampleTime	**IN**, Specify the timestamp of this frame. If the value is 0, QCAP will auto generate it by system clock time.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to push un-compression audio buffer into virtual camera’s share memory.

Note

The SampleTime of audio must same with SampleTime of video.

Example

```
QCAP_SET_AUDIO_VIRTUAL_CAMERA_UNCOMPRESSION_BUFFER( pCamera, &nFrameBuffer, 1280 * 720 * 2, 0 );
```

13.9 QCAP_START_VIRTUAL_CAMERA

```
QRESULT QCAP_START_VIRTUAL_CAMERA( PVOID pCamera )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to start virtual camera.

Example

```
QCAP_START_VIRTUAL_CAMERA( pCamera );
```

13.10 QCAP_STOP_VIRTUAL_CAMERA

```
QRESULT QCAP_STOP_VIRTUAL_CAMERA( PVOID pCamera )
```

Parameter

Name	Description
pCamera	**IN**, Handle of the camera object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to stop virtual camera.

Example

```
QCAP_STOP_VIRTUAL_CAMERA( pCamera );
```

14 Callback Function API

14.1 QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK

```
QRETURN QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK( PVOID                              pDevice,
                                                   PF_NO_SIGNAL_DETECTED_CALLBACK     pCB,
                                                   PVOID                              pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_NO_SIGNAL_DETECTED_CALLBACK)( PVOID pDevice /*IN*/,
                                                                ULONG nVideoInput /*IN*/,
                                                                ULONG nAudioInput /*IN*/,
                                                                PVOID pUserData /*IN*/ );
```

User can use this function to register above callback function.

When capture device cannot detect the signal from current input, the callback function will be called.

User can implement its own function to handle these information.

Note

This function is started after `QCAP_CREATE` right now `(` not after `QCAP_RUN )`.

Example

```
QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK( pDevice, pCB, this );
```

14.2 QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK

```
QRETURN QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK( PVOID                          pDevice,
                                               PF_SIGNAL_REMOVED_CALLBACK     pCB,
                                               PVOID                          pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_SIGNAL_REMOVED_CALLBACK)( PVOID pDevice /*IN*/,
                                                            ULONG nVideoInput /*IN*/,
                                                            ULONG nAudioInput /*IN*/,
                                                            PVOID pUserData /*IN*/ );
```

User can use this function to register above callback function.

When the input signal is loss, the callback function will be called.

The callback function is different from `NO_SIGNAL_DETECTED_CALLBACK`, and it is only called when input is loss from exist state.

User can implement its own functions to handle these information.

Note

This function is started after `QCAP_CREATE` right now `(` not after `QCAP_RUN )`.

Example

```
QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK( pDevice, pCB, this );
```

14.3 QCAP_REGISTER_FORMAT_CHANGED_CALLBACK

```
QRETURN QCAP_REGISTER_FORMAT_CHANGED_CALLBACK( PVOID                          pDevice,
                                               PF_FORMAT_CHANGED_CALLBACK     pCB,
                                               PVOID                          pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_FORMAT_CHANGED_CALLBACK)( PVOID  pDevice /*IN*/,
                                                            ULONG  nVideoInput /*IN*/,
                                                            ULONG  nAudioInput /*IN*/,
                                                            ULONG  nVideoWidth /*IN*/,
                                                            ULONG  nVideoHeight /*IN*/,
                                                            BOOL   bVideoIsInterleaved /*IN*/,
                                                            ULONG  nVideoFrameRate /*IN*/,
                                                            ULONG  nAudioChannels /*IN*/,
                                                            ULONG  nAudioBitsPerSample /*IN*/,
                                                            ULONG  nAudioSampleFrequency /*IN*/,
                                                            PVOID  pUserData /*IN*/ );
```

User can use this function to register above callback function.

When the input signal is detected or input format is changed, the callback function will be called.

The callback function can provide video/audio information, such as video input, audio input, resolution, video interleaved,

frame per second, audio channel number, audio sample width and audio sampling frequency.

User can implement its own function to handle these information.

Note

This function is started after `QCAP_CREATE` right now `(` not after `QCAP_RUN )`.

Example

```
QCAP_REGISTER_FORMAT_CHANGED_CALLBACK( pDevice, pCB, this );
```

14.4 QCAP_REGISTER_SNAPSHOT_DONE_CALLBACK

```
QRETURN QCAP_REGISTER_SNAPSHOT_DONE_CALLBACK(  PVOID                          pDevice,
                                               PF_SNAPSHOT_DONE_CALLBACK      pCB,
                                               PVOID                          pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_SNAPSHOT_DONE_CALLBACK)(  PVOID   pDevice /*IN*/,
                                                            CHAR *  pszFilePathName /*IN*/,
                                                            PVOID   pUserDate /*IN*/ );
```

User can use this function to register above callback function.

The callback function will be called, when one snapshot comand `QCAP_SNAPSHOT` is complete.

The developer, who uses async snapshot, can use the function to know when snapshot file is ready.

Example

```
QCAP_REGISTER_SNAPSHOT_DONE_CALLBACK( pDevice, pCB, this );
```

14.5 QCAP_REGISTER_RECORD_DONE_CALLBACK

```
QRETURN QCAP_REGISTER_RECORD_DONE_CALLBACK(    PVOID                          pDevice,
                                               UINT                           iRecNum,
                                               PF_RECORD_DONE_CALLBACK        pCB,
                                               PVOID                          pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_RECORD_DONE_CALLBACK)( PVOID   pDevice /*IN*/,
                                                         UINT    iRecNum /*IN*/,
                                                         CHAR *  pszFilePathName /*IN*/,
                                                         PVOID   pUserDate /*IN*/ );
```

User can use this function to register above callback function.

The callback function will be called, when the video record file of `QCAP_START_RECORD` is complete.

And developer, who uses async record, can use the function to know when video record file is ready.

Note

Some recording format, such as MP4 the file closing need cost several seconds to finish, developer who use asyc record stop flag, can use this callback to know when file is ready.

Example

```
QCAP_REGISTER_RECORD_DONE_CALLBACK( pDevice, pCB, this );
```

14.6 QCAP_REGISTER_SHARE_RECORD_DONE_CALLBACK

```
QRETURN QCAP_REGISTER_SHARE_RECORD_DONE_CALLBACK( UINT                                iRecNum,
                                                  PF_SHARE_RECORD_DONE_CALLBACK       pCB,
                                                  PVOID                               pUserData )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_SHARE_RECORD_DONE_CALLBACK)( UINT    iRecNum /*IN*/,
                                                               CHAR *  pszFilePathName /*IN*/,
                                                               PVOID   pUserDate /*IN*/ );
```

User can use this function to register above callback function.

The callback function will be called, when the video record file of `QCAP_START_SHARE_RECORD` is complete.

And developer, who uses async record, can use the function to know when video record file is ready.

Note

Some recording format, such as MP4 the file closing need cost several seconds to finish, developer who use asyc record stop flag, can use this callback to know when file is ready.

Example

```
QCAP_REGISTER_SHARE_RECORD_DONE_CALLBACK( pDevice, pCB, this );
```

14.7 QCAP_REGISTER_BROADCAST_CLIENT_RECORD_DONE_CALLBACK

```
QRETURN QCAP_REGISTER_BROADCAST_CLIENT_RECORD_DONE_CALLBACK(
                                                    PVOID                                       pClient,
                                                    PF_BROADCAST_CLIENT_RECORD_DONE_CALLBACK    pCB,
                                                    PVOID                                       pUserData )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT * PF_BROADCAST_CLIENT_RECORD_DONE_CALLBACK )(
                                                                    PVOID   pClient /*IN*/,
                                                                    CHAR *  pszFilePathName /*IN*/,
                                                                    PVOID   pUserDate /*IN*/ );
```

User can use this function to register above callback function.

The callback function will be called, when the video record file of `QCAP_START_BROADCAST_CLIENT_RECORD` is complete.

And developer, who uses async record, can use the function to know when video record file is ready.

Note

Some recording format, such as MP4 the file closing need cost several seconds to finish, developer who use asyc record stop flag, can use this callback to know when file is ready.

Example

```
QCAP_REGISTER_BROADCAST_CLIENT_RECORD_DONE_CALLBACK(pClient, pCB, this );
```

14.8 QCAP_REGISTER_BROADCAST_CLIENT_CONNECTED_CALLBACK

```
QRETURN QCAP_REGISTER_BROADCAST_CLIENT_CONNECTED_CALLBACK(
                                                    PVOID                                   pClient,
                                                    PF_VIDEO_HARDWARE_ENCODER_CALLBACK      pCB,
                                                    PVOID                                   pUserData )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
nVideoWidth	**IN**, Specify the horizontal width of video
nVideoHeight	**IN**, Specify the vertical height of video
nVideoIsInterleaved	**IN**, Specify the video Interleaved
nVideoFramRate	**IN**, Specify the video frame rate
nAudioChannels	**IN**, Specify the audio channel
nAudioBitPerSample	**IN**, Specify the bit per sample
nAudioSampleFrequency	**IN**, Specify the audio sample frequency
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_BROADCAST_CLIENT_CONNECTED_CALLBACK)(
                                                                      PVOID   pClient /*IN*/,
                                                                      ULONG   nVideoWidth /*IN*/,
                                                                      ULONG   nVideoHeight /*IN*/,
                                                                      BOOL    bVideoIsInterleaved /*IN*/,
                                                                      ULONG   nVideoFrameRate /*IN*/,
                                                                      ULONG   nAudioChannels /*IN*/,
                                                                      ULONG   nAudioBitsPerSample /*IN*/,
                                                                      ULONG   nAudioSampleFrequency /*IN*/,
                                                                      PVOID   pUserData /*IN*/ );
```

User can use this function to register above callback function.

User can use this function to get format from broadcast client, when the broad cleint connected was successful, the callback function will be called. The callback function can get video/audio information, such as video width, video height, video interleaved, frame per second, audio channel number, audio sample width and audio sampling frequency. User can implement its own function to handle these information.

Example

```
QCAP_REGISTER_BROADCAST_CLIENT_CONNECTED_CALLBACK( pClient, pCB, this );
```

14.9 QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK(  PVOID                        pDevice,
                                               PF_VIDEO_PREVIEW_CALLBACK    pCB,
                                               PVOID                        pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_PREVIEW_CALLBACK)( PVOID    pDevice /*IN*/,
                                                          double   dSampleTime /*IN*/,
                                                          BYTE *   pFrameBuffer /*IN*/,
                                                          ULONG    nFrameBufferLen /*IN*/,
                                                          PVOID    pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide video preview data (YUY2 or YV12) for developer to use.

Developer can directly modify the pixel data whithin the callback function, such as alpha blending.

Then the modifcation will directly geneate one filter effect in recording file and display.

Note

#1 If callback return QCAP_RT_FAIL to QCAP, the frame buffer will be dropped and will not be displayed and not sent into recording engine on the window.

#2 User can use callback return QCAP_RT_SKIP_RECORD_XXX to QCAP for set skip frame buffer and select which channel will be skip frame buffer by himself, specify callback return type. Available values are

```
QCAP_RT_SKIP_RECORD_NUM_00   = 0x00010002
QCAP_RT_SKIP_RECORD_NUM_01   = 0x00020002
QCAP_RT_SKIP_RECORD_NUM_02   = 0x00040002
QCAP_RT_SKIP_RECORD_NUM_03   = 0x00080002
```

#3 If callback return QCAP_RT_SKIP_DISPLAY (CLONE_DISPLAY) to QCAP, the video preview data (YUY2 or YV12) and clone video data will not be displayed on the window.

```
QCAP_RT_SKIP_DISPLAY         = 0x00000004
QCAP_RT_SKIP_CLONE_DISPLAY   = 0x00000008
```

#4 dSampleTime is in second unit.

#5 The video format of device is

"DC1150 USB" is YUY2

"UB658G USB" is YV12

"CY3014 USB" is YUY2

"TW6802 PCI" is YUY2

"CX2581 PCI" is YUY2

"AH8400 PCI" is YUY2

"FH8735 PCI" is YV12

"TW5864 PCI" is YV12

"SA7160 PCI" is YUY2

"FH8735 PCI" is YV12

"TW2809 PCI" is YV12

Example

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

14.10 QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK(  PVOID                        pDevice,
                                               PF_AUDIO_PREVIEW_CALLBACK    pCB,
                                               PVOID                        pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_PREVIEW_CALLBACK)(  PVOID    pDevice /*IN*/,
                                                           double   dSampleTime /*IN*/,
                                                           BYTE *   pFrameBuffer /*IN*/,
                                                           ULONG    nFrameBufferLen /*IN*/,
                                                           PVOID    pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide audio uncompressed data (PCM) for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 User can use callback return `QCAP_RT_SKIP_RECORD_XXX` to `QCAP` for set skip frame buffer, specify callback return type. Available values are

```
QCAP_RT_SKIP_RECORD_NUM_00   = 0x00010002
QCAP_RT_SKIP_RECORD_NUM_01   = 0x00020002
QCAP_RT_SKIP_RECORD_NUM_02   = 0x00040002
QCAP_RT_SKIP_RECORD_NUM_03   = 0x00080002
```

#3 If callback return `QCAP_RT_SKIP_ XXX ( CLONE_DISPLAY )` to `QCAP`, the audio uncompressed data (PCM) and clone data will not be played on the window.

```
QCAP_RT_SKIP_DISPLAY         = 0x00000004
QCAP_RT_SKIP_CLONE_DISPLAY   = 0x00000008
```

#4 dSampleTime is in second unit.

Example

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

14.11 QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_HARDWARE_ENCODER_CALLBACK( PVOID                                pDevice,
                                                       UINT                                 iRecNum,
                                                       PF_VIDEO_HARDWARE_ENCODER_CALLBACK   pCB,
                                                       PVOID                                pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index of hardware encoder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_HARDWARE_ENCODER_CALLBACK)(  PVOID     pDevice /*IN*/,
                                                                    UINT      iRecNum /*IN*/,
                                                                    double    dSampleTime /*IN*/,
                                                                    BYTE *    pStreamBuffer /*IN*/,
                                                                    ULONG     nStreamBufferLen /*IN*/,
                                                                    BOOL      bIsKeyFrame /*IN*/,
                                                                    PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide video hardware compressed data (H.264) for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame stream buffer will be dropped and will not be send into the channel recorder.

#2 dSampleTime is in second unit.

Example

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

14.12 QCAP_REGISTER_VIDEO_RECORD_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_RECORD_CALLBACK( PVOID                    pDevice,
                                             UINT                     iRecNum,
                                             PF_VIDEO_RECORD_CALLBACK pCB,
                                             PVOID                    pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_RECORD_CALLBACK)( PVOID     pDevice /*IN*/,
                                                         UINT      iRecNum /*IN*/,
                                                         double    dSampleTime /*IN*/,
                                                         BYTE *    pStreamBuffer /*IN*/,
                                                         ULONG     nStreamBufferLen /*IN*/,
                                                         BOOL      bIsKeyFrame /*IN*/,
                                                         PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide video recording's compressed data (H.264) for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame stream buffer will be dropped and will not be saved on the file.

#2 dSampleTime is in second unit.

Example

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

14.13 QCAP_REGISTER_AUDIO_RECORD_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_RECORD_CALLBACK( PVOID                    pDevice,
                                             UINT                     iRecNum,
                                             PF_AUDIO_RECORD_CALLBACK pCB,
                                             PVOID                    pUserData )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_RECORD_CALLBACK)(  PVOID     pDevice /*IN*/,
                                                          UINT      iRecNum /*IN*/,
                                                          double    dSampleTime /*IN*/,
                                                          BYTE *    pStreamBuffer /*IN*/,
                                                          ULONG     nStreamBufferLen /*IN*/,
                                                          PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide audio recording's compressed data (PCM or AAC) for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame stream buffer will be dropped and will not be saved on the file.

#2 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_RECORD_CALLBACK( pDevice, pCB, pUserData );
```

14.14 QCAP_REGISTER_VIDEO_SHARE_RECORD_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_SHARE_RECORD_CALLBACK( UINT                          iRecNum,
                                                   PF_VIDEO_RECORD_SHARE_CALLBACK pCB,
                                                   PVOID                         pUserData )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_SHARE_RECORD_CALLBACK)( UINT      iRecNum /*IN*/,
                                                               double    dSampleTime /*IN*/,
                                                               BYTE *    pStreamBuffer /*IN*/,
                                                               ULONG     nStreamBufferLen /*IN*/,
                                                               BOOL      bIsKeyFrame /*IN*/,
                                                               PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide video recording's compressed data (H.264) for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame stream buffer will be dropped and will not be saved on the file.

#2 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_SHARE_RECORD_CALLBACK( pDevice, pCB, pUserData );
```

14.15 QCAP_REGISTER_AUDIO_SHARE_RECORD_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_SHARE_RECORD_CALLBACK( UINT                          iRecNum,
                                                   PF_AUDIO_SHARE_RECORD_CALLBACK pCB,
                                                   PVOID                         pUserData )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_SHARE_RECORD_CALLBACK)( UINT      iRecNum /*IN*/,
                                                               double    dSampleTime /*IN*/,
                                                               BYTE *    pStreamBuffer /*IN*/,
                                                               ULONG     nStreamBufferLen /*IN*/,
                                                               PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide audio recording's compressed data (PCM or AAC) for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame stream buffer will be dropped and will not be saved on the file.

#2 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_SHARE_RECORD_CALLBACK( pDevice, pCB, pUserData );
```

14.16 QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK(
                                                    UINT                                    iRecNum,
                                                    PF_VIDEO_DECODER_SHARE_RECORD_CALLBACK  pCB,
                                                    PVOID                                   pUserData )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_DECODER_SHARE_RECORD_CALLBACK)(    UINT      iRecNum /*IN*/,
                                                                          double    dSampleTime /*IN*/,
                                                                          BYTE *    pFrameBuffer /*IN*/,
                                                                          ULONG     nFrameBufferLen /*IN*/,
                                                                          PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK`, this callback function will will provide video recording's uncompressed data (YV12) after video decoder for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame buffer will be dropped and will not be displayed on the window.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the video preview data (YV12) will not be displayed on the window.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_DECODER_SHARE_RECORD_CALLBACK( iRecNum, pCB, pUserData );
```

14.17 QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK(
                                                    UINT                                    iRecNum,
                                                    PF_AUDIO_DECODER_SHARE_RECORD_CALLBACK  pCB,
                                                    PVOID                                   pUserData )
```

Parameter

Name	Description
iRecNum	**IN**, Specify recorder index of recorder, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_DECODER_SHARE_RECORD_CALLBACK)(    UINT      iRecNum /*IN*/,
                                                                          double    dSampleTime /*IN*/,
                                                                          BYTE *    pFrameBuffer /*IN*/,
                                                                          ULONG     nFrameBufferLen /*IN*/,
                                                                          PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK`, this callback function will will provide audio uncompressed data (PCM) after audio decoder for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the audio uncompressed data (PCM) will not be played on the window.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_DECODER_SHARE_RECORD_CALLBACK( iRecNum, pCB, pUserData );
```

14.18 QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK(
                                        PVOID                                   pServer,
                                        UINT                                    iSessionNum,
                                        PF_VIDEO_BROADCAST_SERVER_CALLBACK      pCB,
                                        PVOID                                   pUserData )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_BROADCAST_SERVER_CALLBACK)( PVOID     pServer /*IN*/,
                                                                   UINT      iSessionNum /*IN*/,
                                                                   double    dSampleTime /*IN*/,
                                                                   BYTE   *  pStreamBuffer /*IN*/,
                                                                   ULONG     nStreamBufferLen /*IN*/,
                                                                   BOOL      bIsKeyFrame /*IN*/,
                                                                   PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide video recording's compressed data (H.264) for developer of broadcast server to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame stream buffer will be dropped and will not be saved on the file.

#2 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_BROADCAST_SERVER_CALLBACK( pDevice, pCB, pUserData );
```

14.19 QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK(
                                                PVOID                                pServer,
                                                UINT                                 iSessionNum,
                                                PF_AUDIO_BROADCAST_SERVER_CALLBACK   pCB,
                                                PVOID                                pUserData )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_BROADCAST_SERVER_CALLBACK)( PVOID     pServer /*IN*/,
                                                                   UINT      iSessionNum /*IN*/,
                                                                   double    dSampleTime /*IN*/,
                                                                   BYTE   *  pStreamBuffer /*IN*/,
                                                                   ULONG     nStreamBufferLen /*IN*/,
                                                                   PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The callback function will provide audio recording's compressed data (PCM or AAC) for developer of broadcast server to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_BROADCAST_SERVER_CALLBACK( pDevice, pCB, pUserData );
```

14.20 QCAP_REGISTER_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK(
                                        PVOID                                       pServer,
                                        UINT                                        iSessionNum,
                                        PF_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK   pCB,
                                        PVOID                                       pUserData )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK)(
                                                                    PVOID     pServer /*IN*/,
                                                                    UINT      iSessionNum /*IN*/,
                                                                    double    dSampleTime /*IN*/,
                                                                    BYTE   *  pFrameBuffer /*IN*/,
                                                                    ULONG     nFrameBufferLen /*IN*/,
                                                                    PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK`, this callback function will will provide video recording's uncompressed data (YV12) after video decoder for developer of broadcast server to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame buffer will be dropped and will not be displayed on the window.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the video preview data (YV12) will not be displayed on the window of broadcast server.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_DECODER_BROADCAST_SERVER_CALLBACK( pDevice, pCB, pUserData );
```

14.21 QCAP_REGISTER_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK(
                                        PVOID                                       pServer,
                                        UINT                                        iSessionNum,
                                        PF_AUDIO_DEOCDER_BROADCAST_SERVER_CALLBACK  pCB,
                                        PVOID                                       pUserData )
```

Parameter

Name	Description
pServer	**IN**, Handle of the broadcast server object
iSessionNum	**IN**, Specify the number of channel session connections, start from 0
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK)(
                                                                    PVOID     pServer /*IN*/,
                                                                    UINT      iSessionNum /*IN*/,
                                                                    double    dSampleTime /*IN*/,
                                                                    BYTE   *  pFrameBuffer /*IN*/,
                                                                    ULONG     nFrameBufferLen /*IN*/,
                                                                    PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK`, this callback function will will provide audio uncompressed data (PCM) after audio decoder for developer of broadcast server to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the audio uncompressed data (PCM) will not be played on the window of broadcast server.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_DECODER_BROADCAST_SERVER_CALLBACK( pDevice, pCB, pUserData );
```

14.22 QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK(
                                        PVOID                                   pClient,
                                        PF_VIDEO_BROADCAST_CLIENT_CALLBACK      pCB,
                                        PVOID                                   pUserData )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

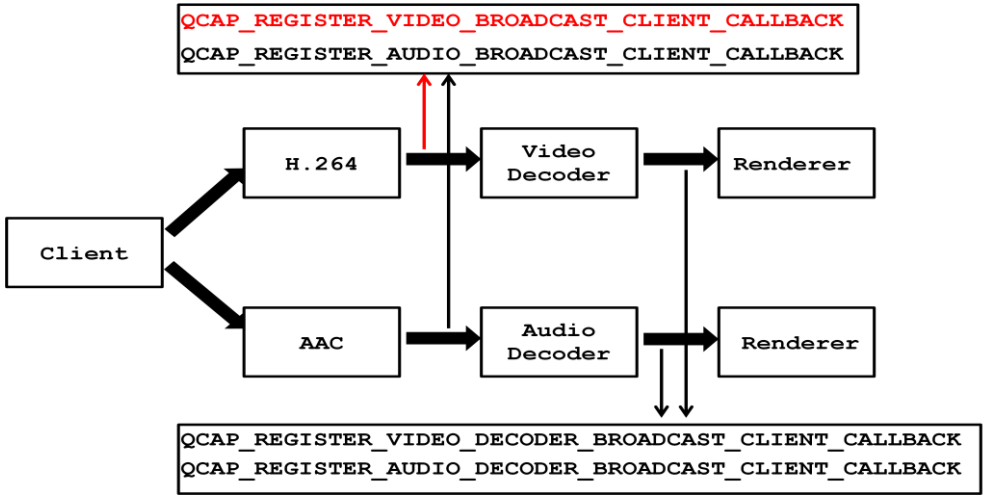
Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_BROADCAST_CLIENT_CALLBACK)( PVOID     pClient /*IN*/,
                                                                   double    dSampleTime /*IN*/,
                                                                   BYTE   *  pStreamBuffer /*IN*/,
                                                                   ULONG     nStreamBufferLen /*IN*/,
                                                                   BOOL      bIsKeyFrame /*IN*/,
                                                                   PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

The flowchart of the video & audio recording approach of broadcast client is shown in figure below.

If user register QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK, this callback function will will provide video compressed data (H.264) before video decoder for developer of broadcast client to use.



Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and will not be displayed and not sent into recording engine on the window.

#2 User can use callback return `QCAP_RT_SKIP_RECORD_NUM_00` to `QCAP` for set skip frame buffer, specify callback return type. Available values is

```
QCAP_RT_SKIP_RECORD_NUM_00 = 0x00010002
```

#3 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`, the video preview data (YUY2 or YV12) will not be displayed on the window of broadcast client.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#4 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_BROADCAST_CLIENT_CALLBACK( pClient, pCB, pUserData );
```

14.23 QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK(
                                                PVOID                                   pClient,
                                                PF_AUDIO_BROADCAST_CLIENT_CALLBACK      pCB,
                                                PVOID                                   pUserData )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_BROADCAST_CLIENT_CALLBACK)( PVOID     pClient /*IN*/,
                                                                   double    dSampleTime /*IN*/,
                                                                   BYTE   *  pStreamBuffer /*IN*/,
                                                                   ULONG     nStreamBufferLen /*IN*/,
                                                                   PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK, this callback function will will provide audio compressed data (PCM / AAC) before audio decoder for developer of broadcast client to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 User can use callback return `QCAP_RT_SKIP_RECORD_NUM_00` to `QCAP` for set skip frame buffer, specify callback return type.

Available values is

```
QCAP_RT_SKIP_RECORD_NUM_00 = 0x00010002
```

#3 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the audio uncompressed data (PCM) will not be played on the window of broadcast client.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#4 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_BROADCAST_CLIENT_CALLBACK( pClient, pCB, pUserData );
```

14.24 QCAP_REGISTER_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK(
                                                    PVOID                                          pClient,
                                                    PF_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK pCB,
                                                    PVOID                                          pUserData )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK)(
                                                                    PVOID     pClient /*IN*/,
                                                                    double    dSampleTime /*IN*/,
                                                                    BYTE   *  pFrameBuffer /*IN*/,
                                                                    ULONG     nFrameBufferLen /*IN*/,
                                                                    PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK`, this callback function will will provide video recording's uncompressed data (YV12) after video decoder for developer of broadcast client to use.

Note

#1 If callback return `QCAP_RT_FAIL` to `QCAP`, the frame buffer will be dropped and will not be displayed on the window.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the video preview data (YV12) will not be displayed on the window of broadcast client.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK( pClient, pCB, pUserData );
```

14.25 QCAP_REGISTER_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK(
                                        PVOID                                       pClient,
                                        PF_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK pCB,
                                        PVOID                                       pUserData )
```

Parameter

Name	Description
pClient	**IN**, Handle of the broadcast client object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_DECODER_BROADCAST_CLIENT_CALLBACK)(
                                                                    PVOID     pClient /*IN*/,
                                                                    double    dSampleTime /*IN*/,
                                                                    BYTE   *  pFrameBuffer /*IN*/,
                                                                    ULONG     nFrameBufferLen /*IN*/,
                                                                    PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK`, this callback function will will provide audio uncompressed data (PCM) after audio decoder for developer of broadcast client to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the audio uncompressed data (PCM) will not be played on the window of broadcast client.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_DECODER_BROADCAST_CLIENT_CALLBACK( pClient, pCB, pUserData );
```

14.26 QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK

```
QRETURN QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK(   PVOID                              pFile,
                                                     PPF_VIDEO_DECODER_FILE_CALLBACK    pCB,
                                                     PVOID                              pUserData )
```

Parameter

Name	Description
pFile	**IN**, Handle of the file object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_VIDEO_DECODER_FILE_CALLBACK)( PVOID     pFile /*IN*/,
                                                               double    dSampleTime /*IN*/,
                                                               BYTE   *  pFrameBuffer /*IN*/,
                                                               ULONG     nFrameBufferLen /*IN*/,
                                                               PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK`, this callback function will will provide video uncompressed data (YV12) after video decoder for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the audio uncompressed data (YV12) will not be played on the window.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_VIDEO_DECODER_FILE_CALLBACK( pFile, pCB, pUserData );
```

14.27 QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK

```
QRETURN QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK(   PVOID                              pFile,
                                                     PPF_AUDIO_DECODER_FILE_CALLBACK    pCB,
                                                     PVOID                              pUserData )
```

Parameter

Name	Description
pFile	**IN**, Handle of the file object
pCB	**IN**, Callback function
pUserData	**IN**, Additional data

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

```
typedef QRETURN (QCAP_EXPORT *PF_AUDIO_DECODER_FILE_CALLBACK)( PVOID     pFile /*IN*/,
                                                               double    dSampleTime /*IN*/,
                                                               BYTE   *  pFrameBuffer /*IN*/,
                                                               ULONG     nFrameBufferLen /*IN*/,
                                                               PVOID     pUserData /*IN*/ );
```

User can use this function to register above callback function.

If user register `QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK`, this callback function will will provide audio uncompressed data (PCM) after audio decoder for developer to use.

Note

#1 If callback return `QCAP_RT_FAIL to QCAP`, the frame buffer will be dropped and its sound will be muted.

#2 If callback return `QCAP_RT_SKIP_DISPLAY` to `QCAP`,

the audio uncompressed data (PCM) will not be played on the window.

```
QCAP_RT_SKIP_DISPLAY = 0x00000004
```

#3 dSampleTime is in second unit.

Example

```
QCAP_REGISTER_AUDIO_DECODER_FILE_CALLBACK( pFile, pCB, pUserData );
```

15 Lock/Unlock Functions API

15.1 QCAP_LOCK_VIDEO_PREVIEW_BUFFER

```
QRETURN QCAP_LOCK_VIDEO_PREVIEW_BUFFER( PVOID      pDevice,
                                        DOUBLE *   pSampleTime,
                                        CHAR   * * ppFrameBuffer,
                                        ULONG  *   pFrameBufferLen )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pSampleTime	**OUT**, Pointer to the storage of sample time
ppFrameBuffer	**OUT**, Pointer to the storage of the frame buffer
pFrameBufferLen	**OUT**, Pointer to the input source buffer's size

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is support for LABVIEW user, ths function is same as video_preview_callback function

User can use this function to lock video data, and use SampleTime to control the time stamp of video to get lock video data.

Note

This function is started after `QCAP_RUN`.

Example

```
QCAP_LOCK_VIDEO_PREVIEW_BUFFER( pDevice, &sampletime, &FrameBuffer, &FrameBufferlen );
```

15.2 QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER

```
QRETURN QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER( PVOID   pDevice )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is support for LABVIEW user, user can use this function to unlock video date,

Example

```
QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER( pDevice );
```

15.3 QCAP_LOCK_AUDIO_PREVIEW_BUFFER

```
QRETURN QCAP_LOCK_AUDIO_PREVIEW_BUFFER( PVOID      pDevice,
                                        DOUBLE *   pSampleTime,
                                        CHAR   * * ppFrameBuffer,
                                        ULONG  *   pFrameBufferLen )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object
pSampleTime	**OUT**, Pointer to the storage of sample time
ppFrameBuffer	**OUT**, Pointer to the storage of the frame buffer
pFrameBufferLen	**OUT**, Pointer to the input source buffer's size

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is support for LABVIEW user, ths function is same as audio_preview_callback function.

User can use this function to lock audio data, and use SampleTime to control the time stamp of audio to get audio data.

Note

This function is started after `QCAP_RUN`.

Example

```
QCAP_LOCK_AUDIO_PREVIEW_BUFFER( pDevice, &sampletime, &FrameBuffer, &FrameBufferlen );
```

15.4 QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER

```
QRETURN QCAP_UNLOCK_VIDEO_PREVIEW_BUFFER( PVOID   pDevice )
```

Parameter

Name	Description
pDevice	**IN**, Handle of the capture card object

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is support for LABVIEW user, user can use this function to unlock audio date,

Example

```
QCAP_UNLOCK_AUDIO_PREVIEW_BUFFER( pDevice );
```

16 Camera Function API

16.1 QCAP_CAMERA_ENUMERATION

```
QRESULT QCAP_CAMERA_ENUMERATION( CHAR * * ppszCameraDevName, BOOL bNext = FALSE )
```

Parameter

Name	Description
ppszCameraDevName	**OUT**, Pointer to the address of all camera devices
bNext	**IN**, Specifies whether to use number n of the camera device on the platform. The value FALSE is to use the first camera device on the platform.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_OUT_OF_RESOURCE	Fail due to out of resource
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is used to enumerate all camera devices on the platform.

Example

```
CHAR * psz_camera_device_name = NULL;

BOOL bNext = FALSE;

QRESULT returns = QCAP_CAMERA_ENUMERATION( &m_pMainDialog->psz_camera_device_name, FALSE );

while( returns == QCAP_RS_SUCCESSFUL ) {

returns = QCAP_CAMERA_ENUMERATION( &m_pMainDialog->psz_camera_device_name, TRUE );

}
```

16.2 QCAP_SET_DEFAULT_CAMERA

```
QRESULT QCAP_SET_DEFAULT_CAMERA( CHAR * pszCameraDevName )
```

Parameter

Name	Description
pszCameraDevName	**IN**, Specify capture filter name of microphone, default is “Camera”

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to modify the default DirectShow Video Capture Filter’s name.

In Win7, the usual friendly name of Video Capture Filter is “XXX” in different language systems;

So QCAP use string, “Camera”, as default.

Example

```
QCAP_SET_DEFAULT_CAMERA( “Camera1” ); // ENGLISH WIN7

QCAP_SET_DEFAULT_CAMERA( “攝影機” ); // CHT WIN7
```

17 Sound Card Function API

17.1 QCAP_SOUNDCARD_ENUMERATION

```
QRESULT QCAP_SOUNDCARD_ENUMERATION(  CHAR * * ppszSoundCardDevName, BOOL bNext = FALSE )
```

Parameter

Name	Description
ppszSoundCardDevName	**OUT**, Pointer to the address of all sound card devices
bNext	**IN**, Specifies whether to use number n of the sound card device on the platform. The value FALSE is to use the first sound card device on the platform.

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_OUT_OF_RESOURCE	Fail due to out of resource
QCAP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

This function is used to enumerate all sound card devices on the platform.

Example

```
CHAR * psz_soundcard_device_name = NULL;

BOOL bNext = FALSE;

QRESULT returns = QCAP_SOUNDCARD_ENUMERATION( &m_pMainDialog->psz_soundcard_device_name, FALSE );

while( returns == QCAP_RS_SUCCESSFUL ) {

returns = QCAP_SOUNDCARD_ENUMERATION( &m_pMainDialog->psz_soundcard_device_name, TRUE );

}
```

17.2 QCAP_SET_DEFAULT_SOUNDCARD

```
QRESULT QCAP_SET_DEFAULT_SOUNDCARD(  CHAR * pszMicrophoneDevName, CHAR * pszLineInDevName )
```

Parameter

Name	Description
pszMicrophoneDevName	**IN**, Specify capture filter name of microphone, default is “High”
pszLineInDevName	**IN**, Specify capture filter name of line in, default is “High”

Returned Value

Name	Description
QCAP_RS_SUCCESSFUL	Success
QCAP_RS_ERROR_GENERAL	Fail due to some errors

Introduction

User can use this function to modify the default DirectShow Audio Capture Filter’s name.

In Win7, the usual friendly name of Audio Capture Filter is “XXX (XXX High Definition Audio)” in different language systems;

So QCAP use string, “High”, as default.

Example

```
QCAP_SET_DEFAULT_SOUNDCARD( “Microphone”, “Line In” ); // ENGLISH WIN7

QCAP_SET_DEFAULT_SOUNDCARD( “麥克風”, “線路輸入” ); // CHT WIN7

QCAP_SET_DEFAULT_SOUNDCARD( “麦克风”, “线路输入” ); // CHS WIN7

QCAP_SET_DEFAULT_SOUNDCARD( “High”, “High” ); // DEFAULT
```