VXG Encoder SDK for iOS
Programmer's Guide

VXG Inc.
August 26, 2017

Content

1. Overview

VXG Encoder SDK consists of a set of resources for fast and convenient development of mobile applications to capture and encode video or audio data and broadcast it using Publish RTMP, RTSP and other protocols. The core of the SDK is a library for application development.

Key Features:

Hardware acceleration – a new hardware accelerated encoder up to UHD resolution.

Multi-core encoding – support of the multiple processor cores for encoding.

Custom and standard notifications – notifies application about connection, disconnection and other events. It is possible to add custom events.

Low latency for network stream – special API to control encoder latency.

Record streams – special API to record streams into mp4 file.

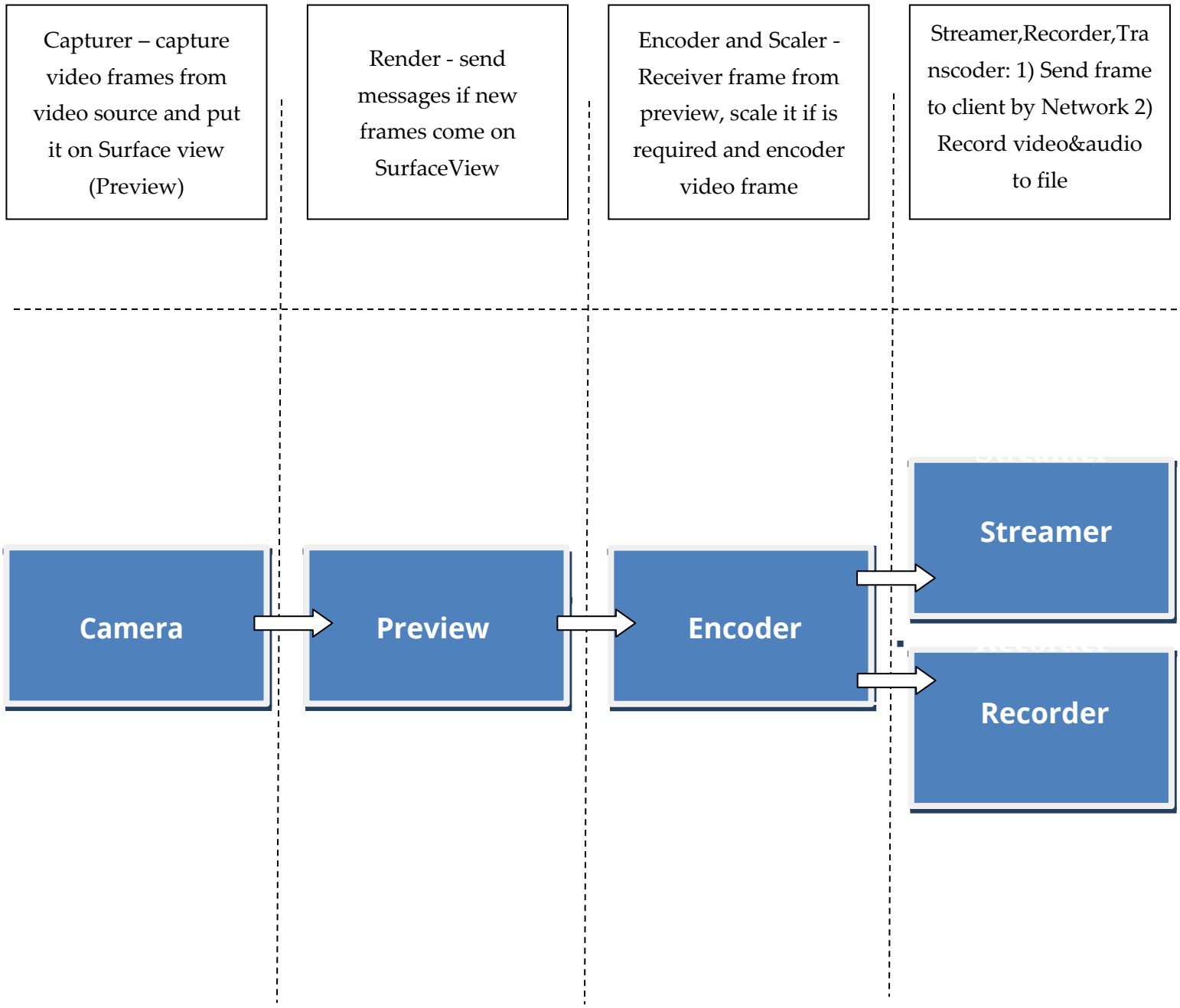
RTSP server – complete functional of RTSP server, RTP by UDP, TCP.

Encrypted channel – RTSP server transfers data by HTTPS tunnel.

Raw video and audio – access to raw data.

Network bandwidth test – client can check bandwidth using RTSP/RTP channel

2. Block diagram



3. How to Use

3.1 iOS version

The SDK works with iOS version 8 or newer.

3.2 Folders and files

The SDK package consists of the following folders.

lib *(Library files to be linked to the application)*
MediaCaptureSDK.h
libMediaCaptureSDK.a
src *(Sample project to test the SDK)*
doc ***(****Documentation including this document)*

3.3 Development tools

Our build environment is XCode. Please import the project to XCode for building the sample application.

3.4 Integration with an application

3.4.1 Integration dynamically (without modifying resources)

1. Create new encoder
MediaRecorder* test_recorder = [[MediaRecorder alloc] init];

2. Create new config
MediaCaptureConfig* test_config = [[MediaCaptureConfig alloc] init];

3. Fill config
3.1. Set previewView (already created UIView* object) if it needs
[test_config setPreview: previewView];

3.2. Set video-format
[test_config setVideoFormat: VIDEO_FORMAT_H264];
[test_config setVideoConfig: 640 : 480: 30: 256*1024];

3.3. Set audio-format if it needs
[test_config setAudioFormat: AUDIO_FORMAT_AAC];

3.4. Set stream params
[test_config setStreamType: STREAM_TYPE_RTMP_PUBLISH];
[test_config setRTMPurl: URL];

4. Open capture
[test_recorder Open: test_config callback: self];

4. Media Capture

4.1 Notifications

SDK notifies about results, errors and notifications using “MediaCapture” callback. All messages are synchronous and SDK core waits until the application handles a message.

Value	Name	Type	Description
0	CAPTURE_VIDEO_OPENED	NOTIFICATION	Capturer has been opened successfully
1	CAPTURE_VIDEO_STARTED	NOTIFICATION	Capturer has been started successfully
2	CAPTURE_VIDEO_STOPED	NOTIFICATION	Capturer has been stopped successfully
3	CAPTURE_VIDEO_CLOSED	NOTIFICATION	Capturer has been closed successfully
4	CAPTURE_AUDIO_OPENED	NOTIFICATION	Audio Capturer has been opened successfully
5	CAPTURE_AUDIO_STARTED	NOTIFICATION	Audio Capturer has been started successfully
6	CAPTURE_AUDIO_STOPED	NOTIFICATION	Audio Capturer has been stopped successfully
7	CAPTURE_AUDIO_CLOSED	NOTIFICATION	Audio Capturer has been closed successfully
8	ENCODER_VIDEO_OPENED	NOTIFICATION	Video Encoder has been opened successfully
9	ENCODER_VIDEO_STARTED	NOTIFICATION	Video Encoder has been started successfully
10	ENCODER_VIDEO_STOPED	NOTIFICATION	Video Encoder has been stopped successfully
11	ENCODER_VIDEO_CLOSED	NOTIFICATION	Video Encoder has been closed successfully

12	ENCODER_AUDIO_OPENED	NOTIFICATION	Audio Encoder has been opened successfully
13	ENCODER_AUDIO_STARTED	NOTIFICATION	Audio Encoder has been started successfully
14	ENCODER_AUDIO_STOPED	NOTIFICATION	Audio Encoder has been stopped successfully
15	ENCODER_AUDIO_CLOSED	NOTIFICATION	Audio Encoder has been closed successfully
16	MUXREC_OPENED	NOTIFICATION	Recorder has been opened successfully
17	MUXREC_STARTED	NOTIFICATION	Recorder has been started successfully
18	MUXREC_STOPED	NOTIFICATION	Recorder has been stopped successfully
19	MUXREC_CLOSED	NOTIFICATION	Recorder has been closed successfully
20	MUXRTMP_OPENED	NOTIFICATION	RTMP publisher has been opened successfully
21	MUXRTMP_STARTED	NOTIFICATION	RTMP publisher has been started successfully
22	MUXRTMP_STOPED	NOTIFICATION	RTMP publisher has been stopped successfully
23	MUXRTMP_CLOSED	NOTIFICATION	RTMP publisher has been closed successfully
24	MUXRTSP_OPENED	NOTIFICATION	RTSP server has been opened successfully
25	MUXRTSP_STARTED	NOTIFICATION	RTSP server has been started successfully
26	MUXRTSP_STOPED	NOTIFICATION	RTSP server has been stopped successfully
27	MUXRTSP_CLOSED	NOTIFICATION	RTSP server has been closed successfully
28	RENDER_OPENED	NOTIFICATION	Preview has been opened successfully
29	RENDER_STARTED	NOTIFICATION	Preview has been started successfully
30	RENDER_STOPED	NOTIFICATION	Preview has been stopped successfully
31	RENDER_CLOSED	NOTIFICATION	Preview has been closed successfully
32	FULL_CLOSE	NOTIFICATION	All modules in pipeline were closed

4.2 Functions description

Following functions are member of MediaCapture class. These functions should be used to broadcast and record media data.

Open

Open camera, create preview and initialize pipeline.

Definition

(int) Open:(MediaCaptureConfig*)cfg callback:(id<MediaCaptureCallback>) cbk;

Parameters:

MediaCaptureConfig	Initialize parameters
MediaPlayerCallback	notification callback, event is provided over this callback

Return Value

Upon successful completion **Open**() returns 0. Otherwise -1 is returned. All errors are provided in callback status.

Remarks

Connect to network resource or open local media file, create pipeline, allocate resource and start video playback.

Example

```
MediaCaptureConfig *config = [[MediaCaptureConfig alloc] init];
[config setAudioFormat: AUDIO_FORMAT_AAC];
[config setVideoFormat: VIDEO_FORMAT_H264];
[config setVideoConfig: 640:480:30:256*1024];
[config setRTMPurl: rtmp_url];

MediaRecorder* recorder = [[MediaRecorder alloc] init];

[recorder Open: config callback: nil];
```

All configuration parameters are described in the table below:

Name	Description	Values	Default value	Type
SetPreview GetPreview	Set/Get Enable Preview		True	Boolean
setVideoConfig getInputWidth getInputHeight getInputFramerate getBitrate	Set/Get Video encoder settings: width, height, framerate,			Int

	bitrate			
SerVideoFormat GervideoFormat	Set/Get Control video formats	VIDEO_FORMAT_h264, VIDEO_FORMAT_NONE	VIDEO_FORMAT_H264	Int
SerAudioFormat GerAudioFormat	Set/Get Control audio format	AUDIO_FORMAT_AAC AUDIO_FORMAT_ULAW AUDIO_FORMAT_NONE	AUDIO_FORMAT_AAC	Int
getRTMPurl setRTMPurl	Set/Get RTMP url on that stream will be broadcasted			String
setRTSPPort getRTSPPort	Set/Get TCP port for RTSP server		554	Int
setStreamType getStreamType	Set/Get Stream type	STREAM_TYPE_RTMP_PUBLISH STREAM_TYPE_RTSP_SERVER STREAM_TYPE_UNKNOWN	STREAM_TYPE_UNKNOWN	Int

Close

Close capturer and release all resources.

Definition

(void) Close;

Parameters

There are no parameters for this call

Return Value

No value is returned by function

Remarks

Close capturer, destroy pipeline, free all resources that were allocated on Open call.

Examples
[capturer Close];

Start

Start all modules (streaming, recording and transcoding) according configuration.

Definition
(void) Start;

Parameters
There are no parameters for this call

Return Value
No value is returned by function

Remarks
Start all modules (streaming, recording) according configuration.

Examples
[capturer Start];

Stop

Stop all started modules. State is changed from Started to Stopped.

Definition
(void) Stop;

Parameters
There are no parameters for this call

Return Value
No value is returned by function

Remarks
Stop all started modules and change state from Started to Stopped.

Examples
[capturer Stop];

StartStreaming

Start only streaming module.

Definition
(void) StartStreaming;

Parameters
There are no parameters for this call

Return Value
No value is returned by function

Remarks
Start streaming module. Format of streaming is set configuration.

Examples
[capturer StartStreaming];

StopStreaming

Stop streaming module.

Definition
(void) StopStreaming;

Parameters
There are no parameters for this call

Return Value
No value is returned by function

Remarks
Stop streaming module.

Examples
[capturer StopStreaming];

StartRecodring

Start only recording module.

Definition
(void) StartRecording;

Parameters
There are no parameters for this call.

Return Value
No value is returned by function

Remarks
Start recording module.

Examples

[capturer StartRecording];

StopRecording

Stop recording module.

Definition
(void) StopRecording;

Parameters
There are no parameters for this call

Return Value
No value is returned by function

Remarks
Stop only recording module.

Examples
[capturer StopRecording];

getvideoPackets
Return number of captured video frames on video capturer.

Definition
(long) GetVideoPackets;

Parameters
There are no parameters for this call

Return Value
Number of video frames that are captured from start.

Remarks
Provide the current state of capturer.

Examples
capturer.GetVideoPackets();

GetAudioPackets

Provide the number of audio samples that are captured by audio captured

Definition

(long) GetAudioPackets;

Parameters

There are no parameters for this call.

Return Value

Upon successful completion, getAudioPackets**()** returns number of samples. Otherwise, -1 is returned. All errors are provided in callback status.

Remarks

Provide the number of audio samples in audio capturer.

Examples

long n= capturer. getAudioPackets () ;