# VXG Media Camera Capture SDK for Android Programmer's Guide

VXG Inc.,

Dec 21, 2016

## Content

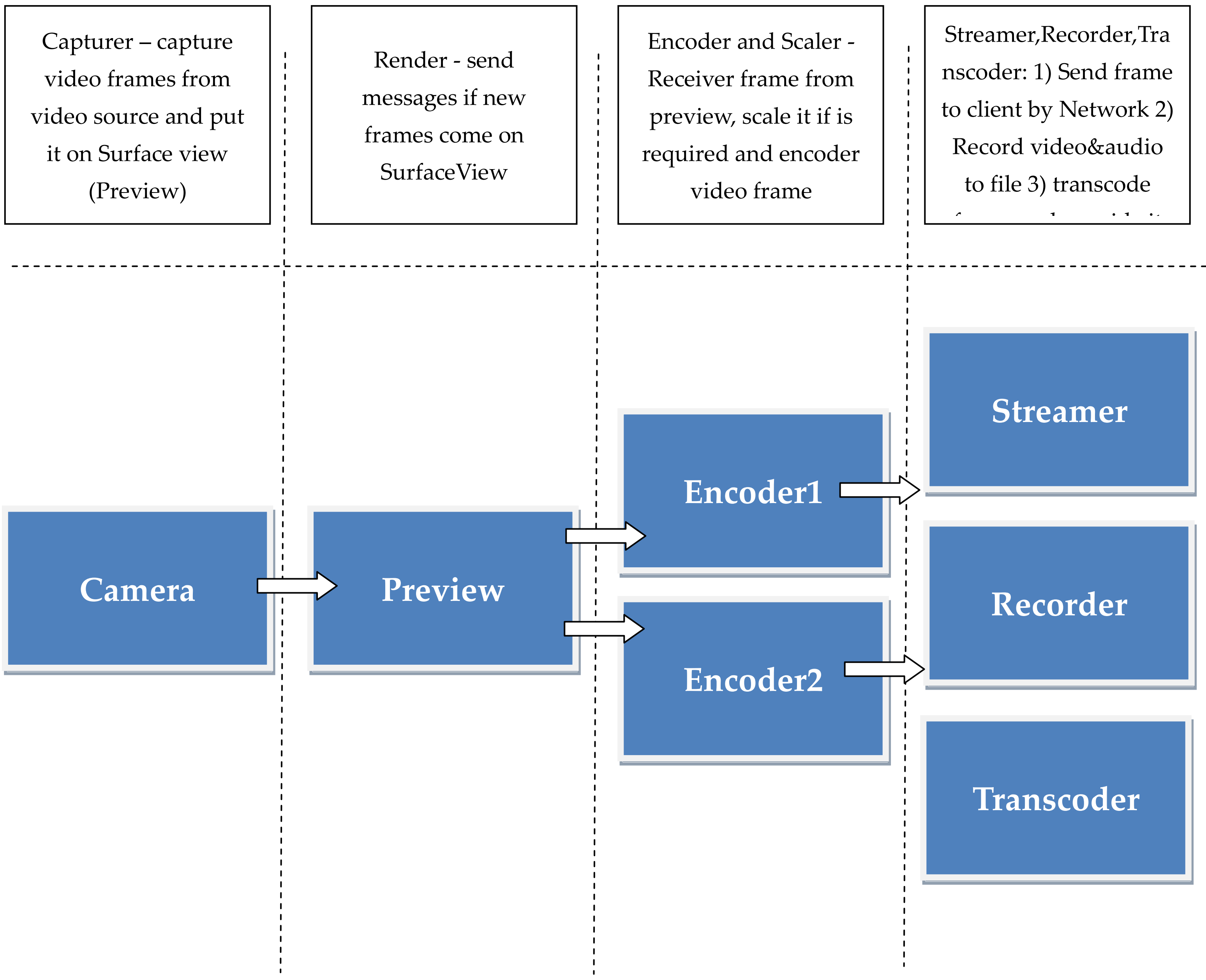
# 1. Overview

Mobile Camera Capture SDK consists of a set of resources for fast and convenient development of mobile applications to capture video or audio stream and provide it by network using Publish RTMP, RTSP. The core of the SDK is a library for application development.

<u>**Key Features:**</u>

**Hardware acceleration** – a new hardware accelerated encoder up to UHD resolution.
**Multi-core encoding** – support of the multiple processor cores for decoding.
**Multi-channel support** – simultaneous encoding of 2 streams: Main and Secondary channels.
**Video integration with any Activity** – is based on SurfaceView and can be integrated into any Activity.
**Hardware pre and post video processing –** hardware de-interlacing and various pre and post video processing using OpenGL shaders.
**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.
**Transcoding** – obtain the raw video or elementary streams video or/and audio.
**RTSP server** – complete functional of RTSP server ,RTP by UDP,TCP,.
**Encrypted channel** – RTSP server transfers data by HTTPS tunnel.
**Streaming from file** – Streaming from file by RTP.
**Network bandwidth test** – client can check bandwidth using RTSP/RTP channel

## 2. Block diagram


Capturer – capture video frames from video source and put it on Surface view (Preview)

Render - send messages if new frames come on SurfaceView

Encoder and Scaler - Receiver frame from preview, scale it if is required and encoder video frame

Streamer,Recorder,Transcoder: 1) Send frame to client by Network 2) Record video&audio to file 3) transcode

# 3. How to Use

## 3.1 Android version

The SDK works with Android version 4.1 (API 16+) or newer.

## 3.2 Folders and files

The SDK package consists of the following folders.

**bin** *(Sample application package)*
MediaStreamTest.apk
**libs** *(Library files to be linked to the application)*
mediacapturedk.jar
libstreamer.so
librtstm.so
**src** *(Sample project to test the SDK)*
**doc** *(Documentation including this document)*

## 3.3 Development tools

Build environment is Eclipse, Android Studio and using gradle.

## 3.4 Integration with an application

### 3.4.1 Integration using a resource file in 2 steps:

**Step1:** Add to layout xml for your activity as below:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >

   <veg.mediacapture.sdk.MediaCapture
      android:id="@+id/captureView"
      android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    />

    </FrameLayout>
```

**Step 2**: Change main activity
(MainActivity.java)

```
public class MainActivity extends Activity implements
MediaCapture.MediaCaptureCallback
{
...
   // callback handler
  @Override
      public int OnCaptureStatus(int arg) { return 0; };
  @Override
      public int OnCaptureReceiveData(ByteBuffer buffer, int type, int size, long pts){
return 0; };


  @Override
  public void onCreate(Bundle savedInstanceState)
  {
   ...
            // Create Capturer instance
            capturer = (MediaCapture)findViewById(R.id.captureView);

            //adjust Capturer’ config
            MediaCaptureConfig config = capturer.getConfig();
            config.setUrl(“rtmp://srv”);
            config.setStreaming(true);
            //etc
            //open the Capturer
            capturer.Open(null, this);
```

```
protected void onPause()
{
        Log.e(TAG, "onPause()");
        super.onPause();

        if (capturer != null)
                capturer.onPause();
}

@Override
protected void onResume()
{
        Log.e(TAG, "onResume()");
        super.onResume();
        if (capturer != null)
                capturer.onResume();
}

@Override
protected void onStart()
{
Log.e(TAG, "onStart()");
        super.onStart();
        sMainActivity = this;

        // Lock screen
        mWakeLock.acquire();

        if (capturer != null)
                capturer.onStart();
}

@Override
protected void onStop()
{
        Log.e(TAG, "onStop()");
```

```
                super.onStop();
                if (capturer != null)
                        capturer.onStop();

                // A WakeLock should only be released when isHeld() is true !
                if (mWakeLock.isHeld()) mWakeLock.release();

                if (toastShot != null)
                        toastShot.cancel();

                if(misSurfaceCreated){
                        finish();
                }
        }

    @Override
    public void onBackPressed()
    {
                if (toastShot != null)
                        toastShot.cancel();

                if(capturer != null)
                        capturer.Close();

                super.onBackPressed();
    }

        @Override
        public void onWindowFocusChanged(boolean hasFocus)
        {
                Log.e(TAG, "onWindowFocusChanged(): " + hasFocus);
                super.onWindowFocusChanged(hasFocus);
                if (capturer != null)
                        capturer.onWindowFocusChanged(hasFocus);
        }
        @Override
        public void onLowMemory()
        {
                Log.e(TAG, "onLowMemory()");
                super.onLowMemory();
                //if (capturer != null)
                //        capturer.onLowMemory();
        }

        @Override
        protected void onDestroy()
        {
                Log.e(TAG, "onDestroy()");
                if (toastShot != null)
                        toastShot.cancel();

                if (capturer != null)
                        capturer.onDestroy();

                System.gc();

                if (multicastLock != null) {
                   multicastLock.release();
                   multicastLock = null;
                }
                super.onDestroy();
        }

}
```

### 3.4.2 Integration dynamically (without modifying resources)

**Step 1**: The approach is similar to 2.4.1 except the capturer is created dynamically within onCreate() method:

```
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
     ...
                // Create Capturer instance
                capturer = new MediaCapture(this, null);

                FrameLayout.LayoutParams          params          =          new
        FrameLayout.LayoutParams(250,250, Gravity.CENTER);
        capturer.setLayoutParams(params);

      //
      // Add Capture Instance to layout
      FrameLayout lp = (FrameLayout)findViewById(R.id.captureView);
      lp.addView(capturer);

                //adjust Capturer’ config
                MediaCaptureConfig config = capturer.getConfig();
                config.setUrl(“rtmp://srv”);
                config.setStreaming(true);
                //etc
                //open the Capturer
                capturer.Open(null, this);

protected void onPause()
        {
                Log.e(TAG, "onPause()");
                super.onPause();

                if (capturer != null)
                        capturer.onPause();
        }

        @Override
        protected void onResume()
        {
                Log.e(TAG, "onResume()");
                super.onResume();
                if (capturer != null)
                        capturer.onResume();
        }

        @Override
        protected void onStart()
        {
        Log.e(TAG, "onStart()");
                super.onStart();
                sMainActivity = this;

                // Lock screen
                mWakeLock.acquire();

                if (capturer != null)
                        capturer.onStart();
        }

        @Override
        protected void onStop()
        {
                Log.e(TAG, "onStop()");
                super.onStop();
                if (capturer != null)
                        capturer.onStop();

                // A WakeLock should only be released when isHeld() is true !
                if (mWakeLock.isHeld()) mWakeLock.release();

                if (toastShot != null)
                        toastShot.cancel();

                if(misSurfaceCreated){
                        finish();
                }
        }

    @Override
    public void onBackPressed()
    {
                if (toastShot != null)
                        toastShot.cancel();

                if(capturer != null)
                        capturer.Close();

                super.onBackPressed();
    }

        @Override
        public void onWindowFocusChanged(boolean hasFocus)
        {
                Log.e(TAG, "onWindowFocusChanged(): " + hasFocus);
                super.onWindowFocusChanged(hasFocus);
                if (capturer != null)
                        capturer.onWindowFocusChanged(hasFocus);
        }

        @Override
        public void onLowMemory()
        {
                Log.e(TAG, "onLowMemory()");
                super.onLowMemory();
                //if (capturer != null)
                //      capturer.onLowMemory();
        }

        @Override
        protected void onDestroy()
        {
                Log.e(TAG, "onDestroy()");
                if (toastShot != null)
                        toastShot.cancel();

                if (capturer != null)
                        capturer.onDestroy();

                System.gc();

                if (multicastLock != null) {
                   multicastLock.release();
                   multicastLock = null;
                }
                super.onDestroy();

   }
```

### 3.4.3 Integration with Activity

The SDK is based on SurfaceView and can be integrated into any Activity using the code below:

```
        <FrameLayout
                android:id="@+id/captureViewLayout"
                android:layout_width="fill_parent"
                android:layout_height=" fill_parent " >

                < veg.mediacapture.sdk.MediaCapture
                   android:id="@+id/captureView"
                   android:layout_width="fill_parent"
                   android:layout_height="fill_parent"
                   android:layout_gravity="center" />
           </FrameLayout>
```

## 3.5 Manifest requirements

Following settings should be set in manifest to avoid any issues with camera using and SDK.

```
android:launchMode="singleInstance"
android:noHistory="true"
android:configChanges="orientation|screenSize"
```

# 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 |
|---|---|---|---|
| 700 | CAP_OPENED | NOTIFICATION | Capturer has been opened successfully |
| 701 | CAP_STARTED | NOTIFICATION | Capturer has been started successfully |
| 702 | CAP_STOPPED | NOTIFICATION | Capturer has been stopped successfully |
| 703 | CAP_CLOSED | NOTIFICATION | Capturer has been closed successfully |
| 704 | CAP_ERROR | NOTIFICATION | Error is happened, details can be got by call function: ErrorGetRTMPStatus or getRECStatus |
| 705 | CAP_TIME | NOTIFICATION | Modules statistics were refreshed |
| 706 | CAP_SURFACE_CREATED | NOTIFICATION | Surface is created, Important notification start function is to be called after this notification |
| 707 | CAP_SURFACE_DESTROYED | NOTIFICATION | Surface is destroyed |
| 708 | CAP_RECORD_STARTED | NOTIFICATION | File record started, see the record property PP_RECORD_STAT_FILE_NAME |
| 709 | CAP_RECORD_STOPPED | NOTIFICATION | File record stopped see the record property PP_RECORD_STAT_FILE_NAME_STOPPED |

## 4.2 Functions description

Following functions are member of MediaCapture class. These functions should be used to playback network content and media files.

### Open

Open camera, create preview and initialize all modules.

Definition
int Open(final MediaCaptureConfig config, final MediaCaptureCallback callback)

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

```
MediaCapture  capturer = new MediaCapture();
// Get config
MediaCaptureConfig config = capturer.getConfig();
        config.setStreaming(true);
        config.setCaptureMode(ncm);
        config.setAudioFormat(MediaCaptureConfig.TYPE_AUDIO_AAC);
        config.setVideoBitrate(abitrate);
        config.setAudioSamplingRate(44100); //hardcoded
        config.setAudioChannels(2);
        config.setUrl(rtmp_url);
        config.setvideoOrientation(0); //landscape
        config.setVideoFramerate(30);
        config.setVideoBitrate(vbitrate);

player.Open(null, This);
```

All configuration parameters are described in the table below:

| Name | Description | Values | Default value | Type |
|---|---|---|---|---|
| Streaming | Set/Get<br>Enable streaming | | True | Boolean |

| | | | | |
|---|---|---|---|---|
| | module | | | |
| UseAVSync | Set/Get<br>Enable AV sync | | True | Boolean |
| AudioFormat | Set/Get<br>Control audio format | TYPE_AUDIO_AAC<br>TYPE_AUDIO_AC3<br>TYPE_AUDIO_AMR_N<br>TYPE_AUDIO_AMR_WB<br>TYPE_AUDIO_EAC3<br>TYPE_AUDIO_FLAC<br>TYPE_AUDIO_G711_ALAW<br>TYPE_AUDIO_G711_MLAW<br>TYPE_AUDIO_RAW<br>TYPE_AUDIO_VORBIS<br>TYPE_AUDIO_MPEG<br>TYPE_AUDIO_MSGSM<br>TYPE_AUDIO_OPUS<br>TYPE_AUDIO_QCELP | TYPE_AUDIO_G711_ALAW | String |
| AudioSamplingRate | Set/Get<br>Control audio sample rate | 8000-96000 (depends on device capabilities) | 44100 | Int |
| AudioChannels | Set/Get<br>Control num of audio channels | 1-5 (depends on device capabilities) | 2 | Int |
| AudioBitrate | Set/Get<br>Control Audio bitrate | Kpbs | 128 | Int |
| VideoBitrate | Set/Get<br>Control Video bitrate | Kpbs | 1000 | Int |
| VideoFramerate | Set/Get<br>Control video frame rate | | 30 | Int |
| videoOrientation | Set/Get<br>Control orientation | 0: landscape; 90: portrait | 0 | Int |
| VideoResolution | Set/Get<br>Control Video resolution | VR_1920x1080(0)<br>VR_1280x720(1)<br>VR_640x480(2)<br>VR_320x240(3)<br>VR_3840x2160(4),<br>VR_720x576(5),<br>VR_640x480(6),<br>VR_352x288(7),<br>VR_176x144(8),<br>VR_640x360(9),<br>VR_720x405(10),<br>VR_864x486(11), | VR_1280x720 | CaptureVideoResolution |

| | | | | |
|---|---|---|---|---|
| | | VR_960x540(12) | | |
| SecVideoBitrate | RTSP only secondary video<br>Set/Get<br>Control Video bitrate | Kpbs | 1000 | Int |
| SecVideoFramerate | RTSP only secondary video<br>Set/Get<br>Control video frame rate | Kbps | 30 | Int |
| SecVideoResolution | RTSP only secondary video<br>Set/Get<br>Control Video resolution | VR_1920x1080(0)<br>VR_1280x720(1)<br>VR_640x480(2)<br>VR_320x240(3)<br>VR_3840x2160(4),<br>VR_720x576(5),<br>VR_640x480(6),<br>VR_352x288(7),<br>VR_176x144(8),<br>VR_640x360(9),<br>VR_720x405(10),<br>VR_864x486(11),<br>VR_960x540(12) | VR_320x240 | CaptureVideoResolution |
| Recording options | | | | |
| Recording | Set/Get<br>Enable video recording | | false | Boolean |
| RecordPath | Set/Get<br>Set full path for recorded files | | "" | String |
| RecordFlags | Set/Get<br>Set setting for recording | PP_RECORD_NO_START(0x00000000)<br>PP_RECORD_AUTO_START(0x00000001)<br>PP_RECORD_SPLIT_BY_TIME(0x00000002)<br>PP_RECORD_SPLIT_BY_SIZE(0x00000004)<br>PP_RECORD_DISABLE_VIDEO(0x00000008)<br>PP_RECORD_DISABLE_AUDIO(0x00000010) | 0 | PlayerRecordFlags |
| RecordSplitTime | Set/Get<br>Split stream on chunks by time if flags are | | 0 | Int |

| | | | | |
|---|---|---|---|---|
| | PP_RECORD_SPLIT_BY_TIME, in seconds | | | |
| RecordSplitSize | Set/Get<br>Split stream on chunks by size if flags are PP_RECORD_SPLIT_BY_ SIZE, in seconds | | 0 | Int |
| RecordPrefix | Set/Get<br>Prefix is added to name of recorded files | | "" | String |
| Transcoding options | | | | |
| Transcoding | Set/Get<br>Enable transcoding | | False | Boolean |
| TransWidth | Set/Get<br>Control width of transcoded picture | | 256 | Int |
| TransHeight | Set/Get<br>Control height of transcoded picture | | 144 | Int |
| TransFps | Set/Get<br>Control height of transcoded picture | | 2 | Int |
| TransFormat | Set/Get | TYPE_VIDEO_RAW | TYPE_VIDEO_RAW | String |
| StreamType | Set/Get<br>Set mode :<br>1) RTMP publish<br>2)RTSP server<br>3) Network Bandwidth test | STREAM_TYPE_RTMP_PUBLISH =0x1,<br>STREAM_TYPE_RTSP_SERVER=0x2;<br>STREAM_TYPE_RTSP_ZIO_SERVER | STREAM_TYPE_RTMP_PUBLISH | Streamer Types |
| VideoUseSrcFile | Set/Get<br>Set mode stream from file | | false | Boolean |
| VideoSrcFilePath | Set/Get<br>Set complete path to file that will be streamed | | "" | String |

## Config.setSecureStreaming

Enable encrypted channel RTP by HTTPS . It works if set StreamType= STREAM_TYPE_RTSP_SERVER.

Definition

public void setSecureStreaming(boolean enabled, String sslCertPEM, String RsaPrivateKey)

Parameters

| | |
|---|---|
| enabled | – enable encrypted channel SSL in HTTP tunnel. |
| sslCertPEM | – ssl certificate in pem format. |
| RsaPrivateKey | - RSA private key. |

Return Value

No value is returned by function SetSecureStreaming.

Remarks

Enable encrypted channel RTP by HTTPS .

URL for tunneled stream is rtsp://TX_IP:8080/ch0 .

Examples

config. SetSecureStreaming (true, certificate, private_key);

## **Close**

Close capturer and release all resources.

Definition

public 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
public void Start()

Parameters
There are no parameters for this call

Return Value
No value is returned by function

Remarks
Start all modules (streaming, recording and transcoding) according configuration. ***Important note:*** **Start** function should be called after CAP_SURFACE_CREATED notification.

Examples
capturer.Start();

## Stop

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

Definition
public 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
public 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.
***Important note:*** **Start** function should be called after CAP_SURFACE_CREATED notification.

Examples
capturer.StartStreaming();

## StopStreaming

Stop streaming module.

Definition
public 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
public void StartRecording()

Parameters
There are no parameters for this call.

Return Value
No value is returned by function

Remarks
Start recording module.
***Important note:*** **Start** function should be called after CAP_SURFACE_CREATED

notification.

Examples

capturer.StartRecording();

## StopRecording

Stop recording module.

Definition

public 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 ();

## StartTranscoding

Start only transcoding module.

Definition

public void StartTranscoding()

Parameters

There are no parameters for this call

Return Value
No value is returned by function

Remarks
Start transcoding module.
***Important note:*** **Start** function should be called after CAP_SURFACE_CREATED notification.

Examples
capturer.StartTranscoding();

## StopTranscoding

Stop transcoding module.

Definition
public void StopTranscoding()

Parameters
There are no parameters  for this call

Return Value
No value is returned by function

Remarks
Stop trancoding module.

Examples
capturer.StopTranscoding ();

## getState

Return capturer state.

Definition

public CaptureState getState()

Parameters
There are no parameters for this call

Return Value
Following states are provided:
0 - Opening
1 - Opened
2 - Started
3 - Paused
4 - Stopped
5 - Closing
6 - Closed

Remarks
Provide the current state of capturer.

Examples
if (capturer.getState() == CapturerState.Closing) ;

### getRTMPStatus

Return status of RTPM.

Definition
public CaptureState getRTMPState()

Parameters
There are no parameters for this call

Return Value
Following states are provided:
0 – NO ERROR
-1 – Try to connect
-5 – Connecting error

-12 – Out of memory
-999 – Demo version

Remarks
Provide the current state of capturer.

Examples
if (capturer.getRTMPState() == CapturerState.Closing) ;

## getRecStatus

Return status of Recording module.

Definition
public CaptureState getRecState()

Parameters
There are no parameters for this call

Return Value
Following states are provided:
0 – NO ERROR
-1 – Try to open file
-5 – File open error
-12 – Out of memory
-999 – Demo version

Remarks
Provide the current state of capturer.

Examples
if (capturer.getRecState() == CapturerState.Closing) ;

## getDuration

Return time from that is expired from starting of capturer.

Definition
public long getDuration()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, getDurarion**()** returns time in milliseconds from capturer start. Otherwise, -1 is returned. All errors are provided in callback status.

Remarks
Return time from that is expired from starting of capturer.

Examples
int duration = capturer.getDuration() ;

### getVideoPackets

Provide the number of video frames in buffer before streaming.

Definition
public long getVideoPackets()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, getVideoPackets**()** returns number of frames. Otherwise, -1 is returned. All errors are provided in callback status.

Remarks
Provide the number of video frames in buffer before streaming. It is used for streaming only, mode :Publish RTMP.

Examples
int duration = capturer. getVideoPackets () ;

## getAudioPackets

Provide the number of audio frames in buffer before streaming.

Definition

public long getAudioPackets()

Parameters

There are no parameters for this call.

Return Value

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

Remarks

Provide the number of audio frames in buffer before streaming. It is used for streaming only, mode :Publish RTMP.

Examples

int duration = capturer. getAudioPackets () ;

## getLastVideoPTS

Provide the timestamp for last video frame is sent by streaming module by network.

Definition

public long getLastVideoPTS()

Parameters

There are no parameters for this call.

Return Value

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

Remarks
Provide the timestamp for last video frame is sent by streaming module by network. It is used for only streaming module in case if mode is Publish RTMP.

Examples
int v_pts = capturer. getLastVideoPTS () ;

### getLastAudioPTS

Provide the timestamp for last audio sample is sent by streaming module by network.

Definition
public long getLastAudioPTS()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, getLastVideoPTS **()** returns timestampt. Otherwise, -1 is returned. All errors are provided in callback status.

Remarks
Provide the timestamp for last audio sample is sent by streaming module by network. It is used for only streaming module in case if mode is Publish RTMP.

Examples
Int a_pts = capturer. getLastAudioPTS () ;

### getStatReconnectCount

Provide the number or reconnections to RTMP server that happened from the streaming start.

Definition
public long getStatReconnectCount()

Parameters

There are no parameters for this call.

Return Value

Upon successful completion, getStatReconnectCount returns number of reconnection. Otherwise, -1 is returned. All errors are provided in callback status.

Remarks

Provide the number or reconnections to RTMP server that happened from streaming start. It is used for only streaming module in case if mode is Publish RTMP.

Examples

Int a_pts = capturer. getStatReconnectCount () ;

**get Record Properties**

| Name | Description | Function | Type |
|---|---|---|---|
| PP_RECORD_STAT_DURATION | Get current recording file duration in milliseconds | getPropLong(PP_RECORD_STAT_DURATION) | long |
| PP_RECORD_STAT_DURATION_TOTAL | Get total recording duration in milliseconds | getPropLong(PP_RECORD_STAT_DURATION_TOTAL) | long |
| PP_RECORD_STAT_SIZE | Get current recording file size in bytes | getPropLong(PP_RECORD_STAT_SIZE) | long |
| PP_RECORD_STAT_SIZE_TOTAL | Get total recorded bytes | getPropLong(PP_RECORD_STAT_SIZE_TOTAL) | long |
| PP_RECORD_STAT_FILE_NAME | Get current recording file name | getPropString(PP_RECORD_STAT_FILE_NAME) | String |
| PP_RECORD_STAT_FILE_NAME_STOPPED | Get file name of file just recorded | getPropString(PP_RECORD_STAT_FILE_NAME_STOPPED) | String |