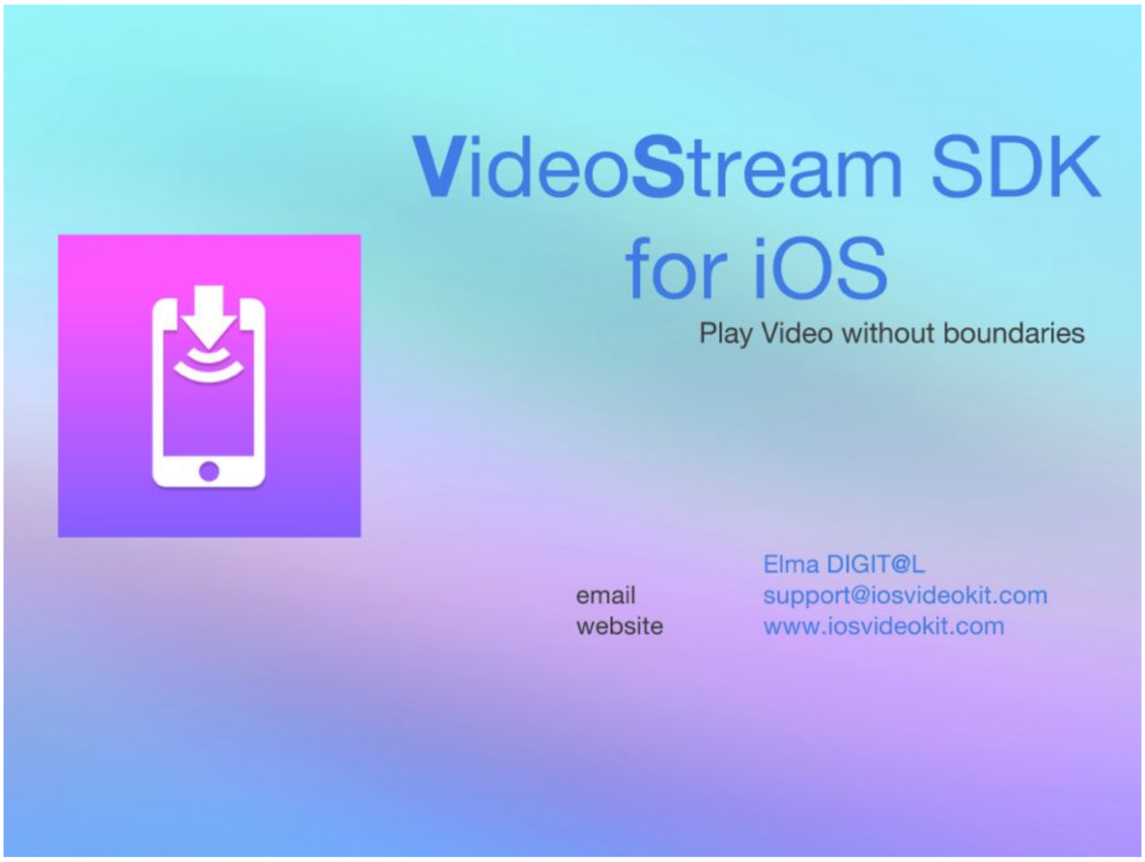


VideoStream SDK for iOS

Play Video without boundaries

Elma DIGIT@L
email support@iosvideokit.com
website www.iosvideokit.com

last updated : May 25, 2016

INTRODUCTION

Apple's video streaming solution is very limited, it supports only http streaming & h264 video codec and its source is closed rendering the task of streaming video a very hard one. If you need a streaming client that uses other popular streaming protocols such as MMS, RTSP, RTMP, or if you stream from an http server but using different audio/video codecs or you stream from a server which you can use also the apple API (MPMoviePlayerController or AVPlayerItem), but need to modify the video player, like buffering duration, audio or video raw data and so on
then that means you will need **VideoStream SDK**.

FEATURES

- Stream from popular protocols (http, mms, rtsp & rtmp)
- Supports all popular audio & video codecs
- Supports mjpeg streams - ipcams
- 720p HD streams are supported by iPad 1 and above.
- Supports real time video effects with using pixel shaders
- Successful Audio & video syncing
- Very easy to use (similar to Apple's MPMoviePlayerViewController API)
- Look & feel like Apple's MPMoviePlayerViewController
- Works with Wifi & 3G
- Shows detailed information about stream (audio & video codecs ,total streamed data in bytes, connection type)
- Works on all latest iOS (iPhone 4/4S, iPhone 5, 5S, 6, 6+ and all iPads) devices & supports all screen types and rotations
- Supports pausing stream
- Supports streaming in background
- Robust error handling
- Supports audio resampling
- Supports seeking in remote file streams
- Supports changing audio streams in realtime
- Supports disabling audio stream in file/stream
- Supports looping for remote file streams

REQUIREMENTS (Paid Version)

iOS **7.x SDK** or above is needed to compile VideoStream SDK however, if you need under 7.x, VideoStream SDK can be built with small modifications.

The VideoStream SDK requires additionally the following Apple frameworks:

- SystemConfiguration (used in **Reachability** framework by Apple)
- MediaPlayer (needed by **MPVolumeView**)
- AudioToolbox (needed by **AudioUnit**)
- AVFoundation (needed by **AVAudioSession**)
- OpenGLES (using when **rendering** pictures to screen)
- QuartzCore (using to draw **StreamInfo view**)

and VideoKit also needs to these frameworks (for connecting to stream, decoding Audio&Video packets, etc...):

- libz
- libiconv
- libavcodec
- libavformat
- libavutil
- libswscale
- libswresample

REQUIREMENTS (Free Version with Ads)

The Free version of VideoStream SDK requires additionally the following Apple frameworks:

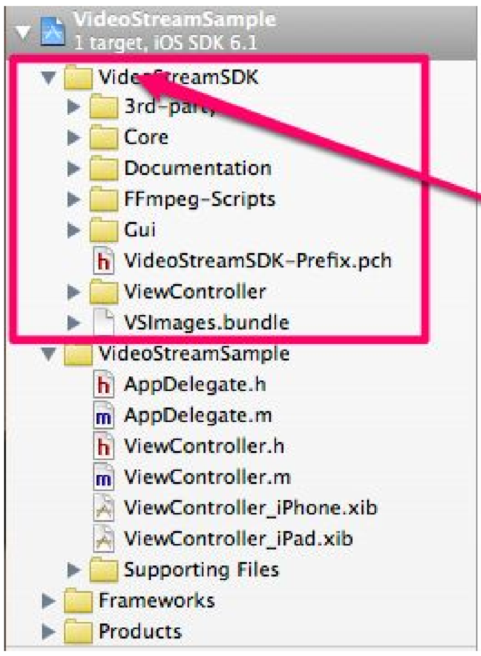
- SystemConfiguration (used in **Reachability** framework by Apple)
- MediaPlayer (needed by **MPVolumeView**)
- AudioToolbox (needed by **AudioUnit**)
- AVFoundation (needed by **AVAudioSession**)
- OpenGLES (using when **rendering** pictures to screen)
- QuartzCore (using to draw **StreamInfo view**)
- iAd
- CoreMotion
- EventKitUI
- MessageUI
- StoreKit
- AdSupport
- CoreTelephony
- EventKit
- GoogleMobileAds
- CoreMedia
- UIKit
- Foundation
- CoreGraphics

and VideoKit also needs to these frameworks (for connecting to stream, decoding Audio&Video packets, etc...):

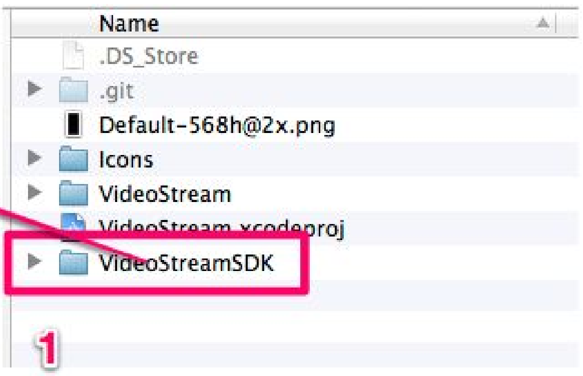
- libz
- libiconv
- libavcodec
- libavformat
- libavutil
- libswscale
- libswresample
- libsqlite3
- libAdapterSDKMDotM
- libAdapterInMobi
- libAdapterIAd
- libAdapterFacebook
- libInMobi-5.2.0
- FBAudienceNetwork

INTEGRATION (Paid Version)

Adding VideoStream SDK to your Xcode project.

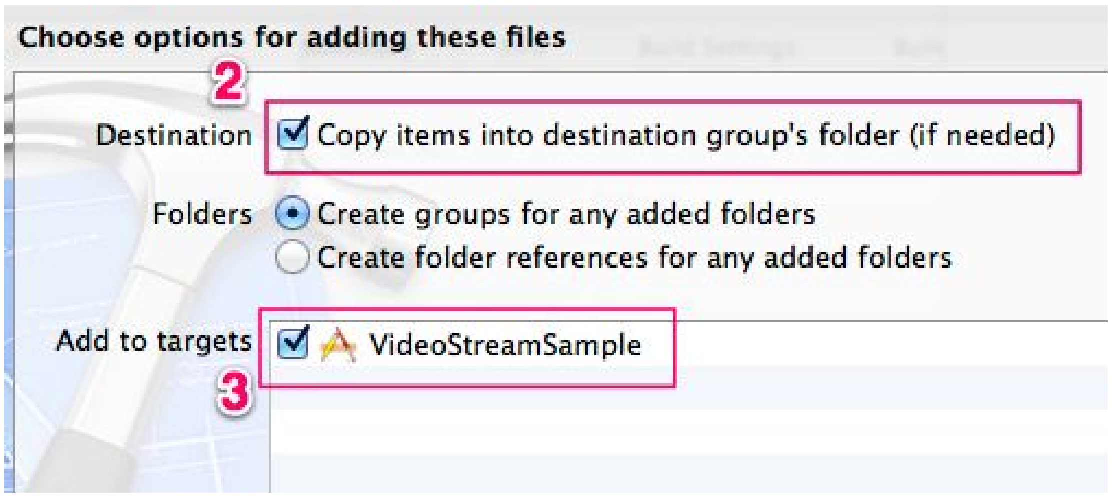


```
▼ VideoStreamSample
  1 target, iOS SDK 6.1
  ▼ VideoStreamSDK
    ▶ 3rd-party
    ▶ Core
    ▶ Documentation
    ▶ FFmpeg-Scripts
    ▶ Gui
      VideoStreamSDK-Prefix.pch
    ▶ ViewController
    ▶ VSImages.bundle
  ▼ VideoStreamSample
      AppDelegate.h
      AppDelegate.m
      ViewController.h
      ViewController.m
      ViewController_iPhone.xib
      ViewController_iPad.xib
    ▶ Supporting Files
  ▶ Frameworks
  ▶ Products
```



```
Name
  .DS_Store
▶ .git
  Default-568h@2x.png
▶ Icons
▶ VideoStream
  VideoStream.xcodeproj
▶ VideoStreamSDK
```

1. Go to VideoStream folder, and add VideoStreamSDK folder to your Xcode project. (Please note that, VideoStreamSDK folder must be added to **root** not under any folder)



Choose options for adding these files

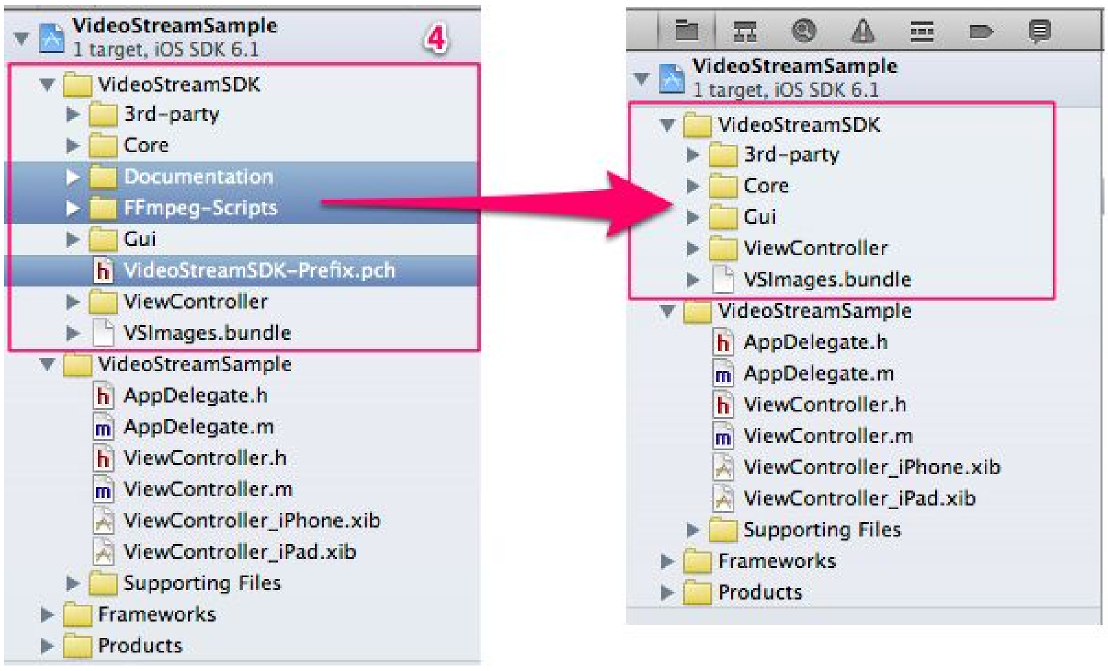
Destination: ☑ Copy items into destination group's folder (if needed)

Folders: (●) Create groups for any added folders
( ) Create folder references for any added folders

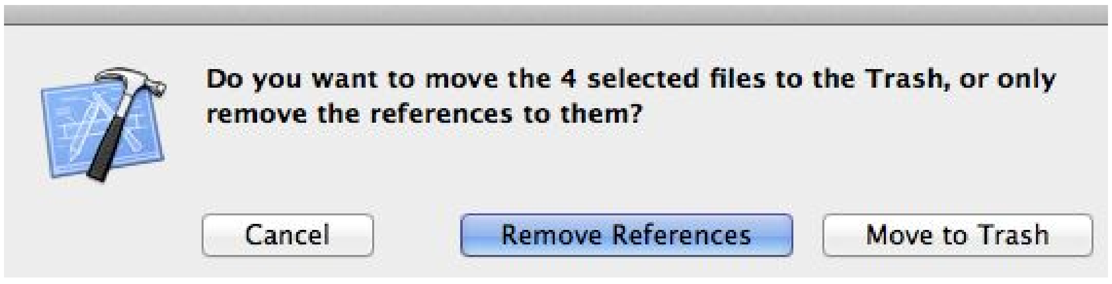
Add to targets: ☑ VideoStreamSample

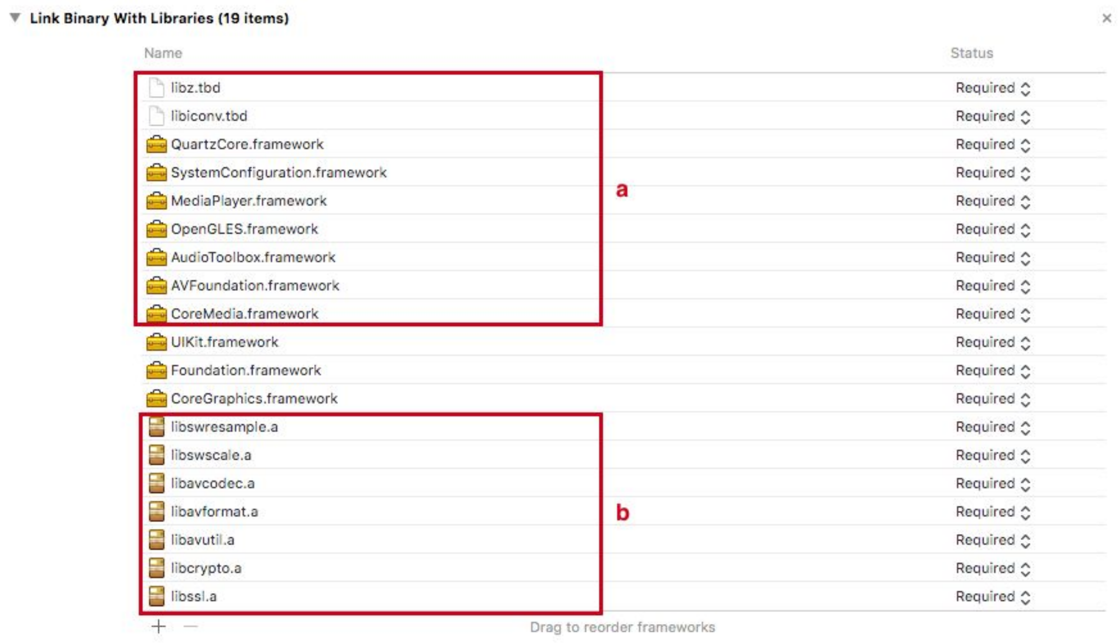
2. Be sure that ‘Copy items into destination group’s folder’ is checked

3. Be sure that ‘VideoStreamSample’ target is check



4. Remove selected unnecessary folders (Documentation, FFmpeg-Scripts) and the file (VideoStreamSDK-Prefix.pch). When asking about remove process, select ‘Remove References’. After doing this, our VideoStreamSDK folder is seen like above (at right).


Do you want to move the 4 selected files to the Trash, or only remove the references to them?



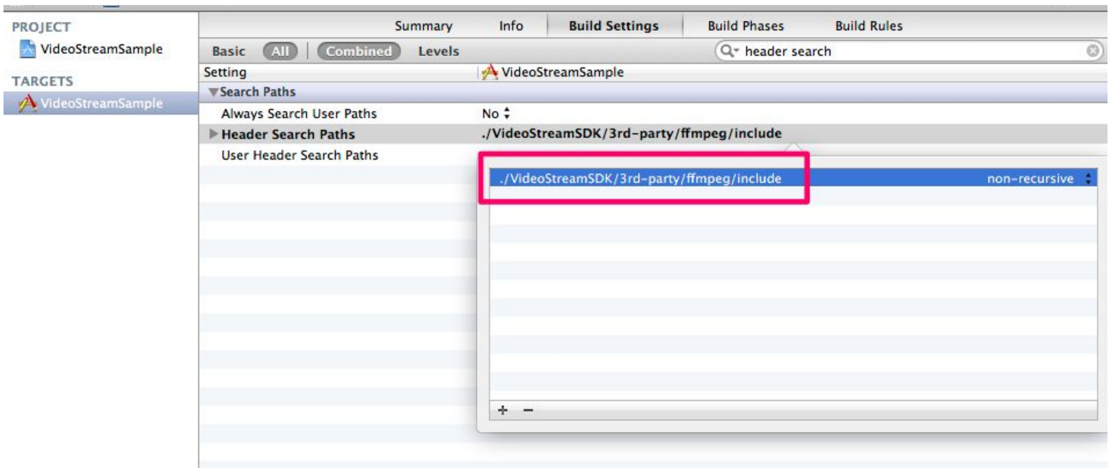
▼ Link Binary With Libraries (19 items)

Name	Status
libz.tbd	Required
libiconv.tbd	Required
QuartzCore.framework	Required
SystemConfiguration.framework	Required
MediaPlayer.framework	Required
OpenGLES.framework	Required
AudioToolbox.framework	Required
AVFoundation.framework	Required
CoreMedia.framework	Required
UIKit.framework	Required
Foundation.framework	Required
CoreGraphics.framework	Required
libswresample.a	Required
libswscale.a	Required
libavcodec.a	Required
libavformat.a	Required
libavutil.a	Required
libcrypto.a	Required
libssl.a	Required

5. Select your target, and go to the **Build Phases** tab.

a. Add these frameworks
   1. libz
   2. libiconv
   2. QuartzCore
   3. SystemConfiguration
   4. MediaPlayer
   5. OpenGLES
   6. AudioToolbox
   7. AVFoundation

b. Those libraries (located in rectangle labelled as “b” in above picture) are automatically added when you add VideoStreamSDK folder to your xcode project.

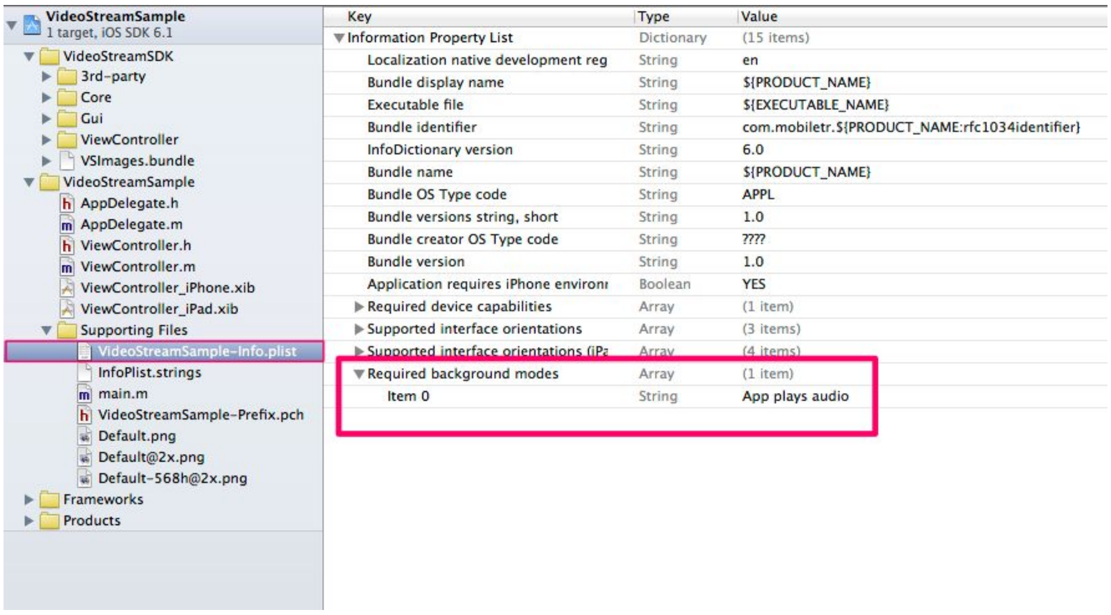


Setting	VideoStreamSample
▼Search Paths	
Always Search User Paths	No
▶Header Search Paths	./VideoStreamSDK/3rd-party/ffmpeg/include
User Header Search Paths	

./VideoStreamSDK/3rd-party/ffmpeg/include	non-recursive

6. Select your target again, and go to **Build Settings** tab

Find “**Header Search Paths”,** set it as below
“Header Search Paths” | **./VideoStreamSDK/3rd-party/ffmpeg/include**



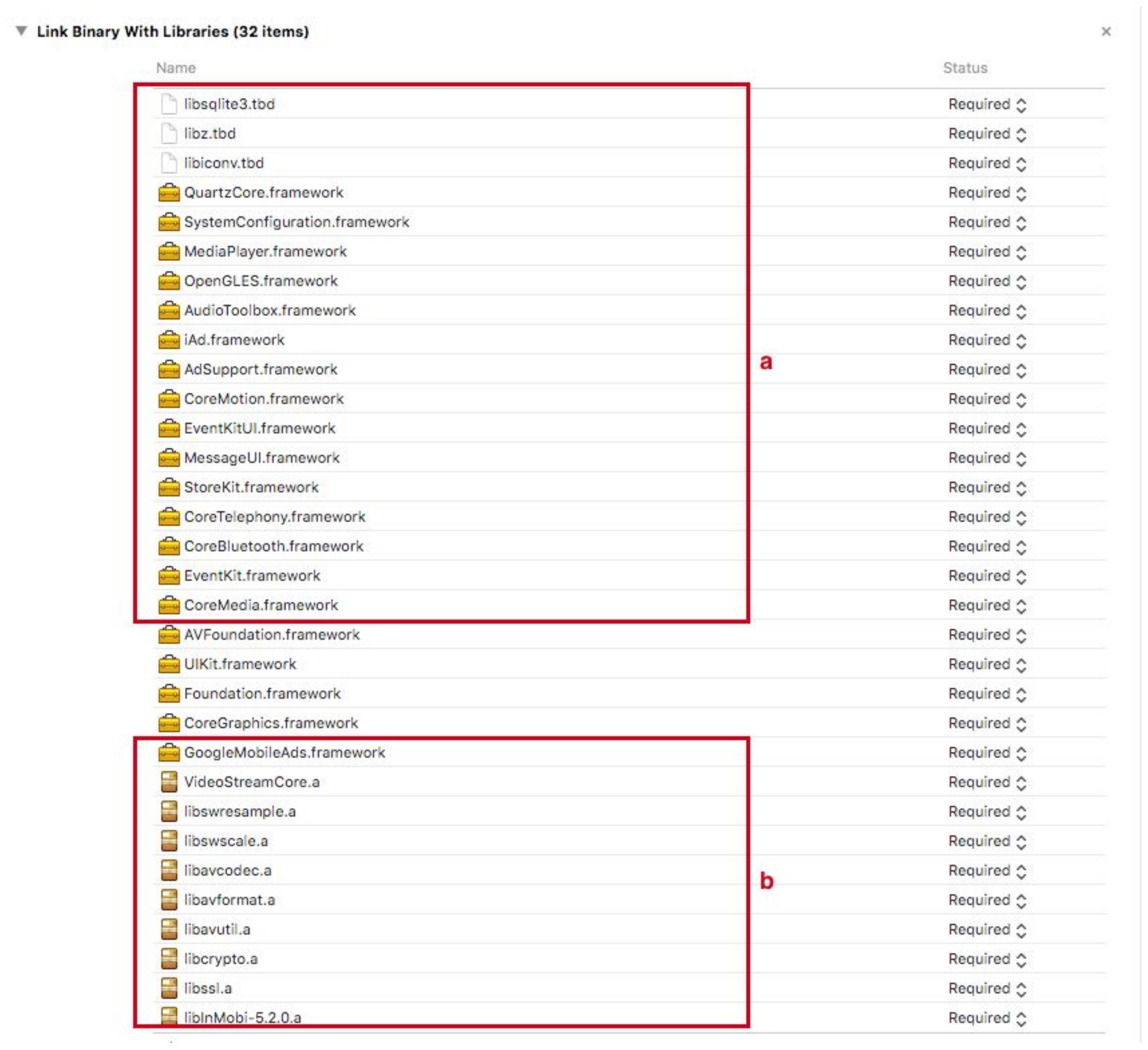
Key	Type	Value
▼Information Property List	Dictionary	(15 items)
Localization native development reg	String	en
Bundle display name	String	${PRODUCT_NAME}
Executable file	String	${EXECUTABLE_NAME}
Bundle identifier	String	com.mobiletr.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	String	6.0
Bundle name	String	${PRODUCT_NAME}
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environı	Boolean	YES
▶Required device capabilities	Array	(1 item)
▶Supported interface orientations	Array	(3 items)
▶Supported interface orientations (iPa	Array	(4 items)
▼Required background modes	Array	(1 item)
Item 0	String	App plays audio

7. Now, add the ‘Required Background Modes’ key to your plist file and set item 0’s value to **“App plays audio”**

Integration is done, your project should be compiled without any error now. (If your project is ARC enabled, then please see section 4 (**ARC Supported Projects**) in below)

INTEGRATION (Free Version With Ad)

Steps 1 to 4 are the same as the paid version. Step 5 and 6 should be :



Link Binary With Libraries (32 items)

Name	Status
libsqlite3.tbd	Required
libz.tbd	Required
libiconv.tbd	Required
QuartzCore.framework	Required
SystemConfiguration.framework	Required
MediaPlayer.framework	Required
OpenGLES.framework	Required
AudioToolbox.framework	Required
iAd.framework	Required
AdSupport.framework	Required
CoreMotion.framework	Required
EventKitUI.framework	Required
MessageUI.framework	Required
StoreKit.framework	Required
CoreTelephony.framework	Required
CoreBluetooth.framework	Required
EventKit.framework	Required
CoreMedia.framework	Required
AVFoundation.framework	Required
UIKit.framework	Required
Foundation.framework	Required
CoreGraphics.framework	Required
GoogleMobileAds.framework	Required
VideoStreamCore.a	Required
libswresample.a	Required
libswscale.a	Required
libavcodec.a	Required
libavformat.a	Required
libavutil.a	Required
libcrypto.a	Required
libssl.a	Required
libInMobi-5.2.0.a	Required

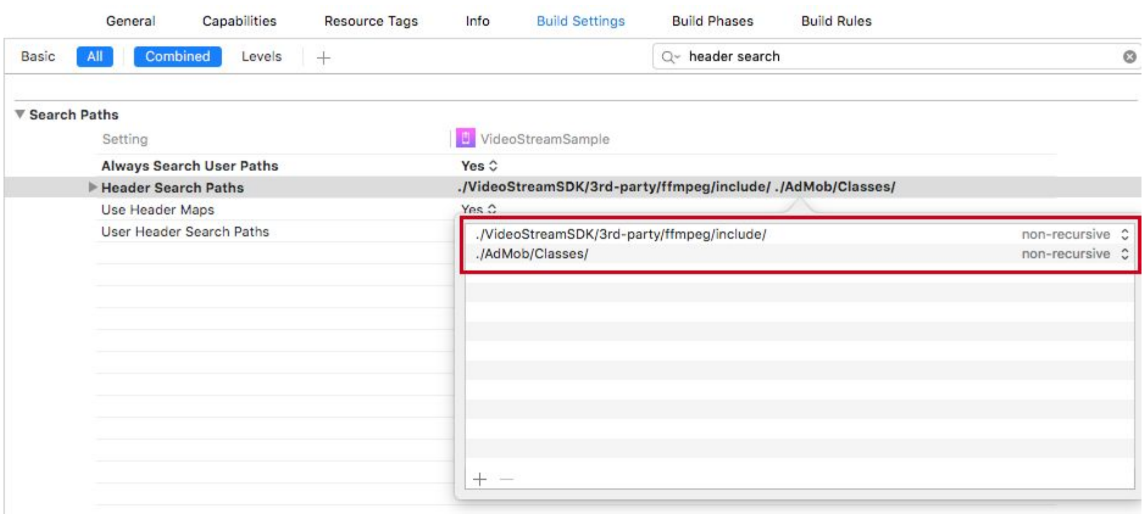
5. Select your target, and go to the **Build Phases** tab.

a. Add these frameworks

- libsqlite3
- libz
- libiconv
- EventKitUI
- iAd
- CoreMotion
- MessageUI
- StoreKit
- AdSupport

- CoreTelephony
- CoreBluetooth
- EventKit
- CoreMedia
- OpenGLES
- QuartzCore
- SystemConfiguration
- MediaPlayer
- AudioToolbox

b. Those libraries (located in rectangle labelled as “b” in above picture) are automatically added when you add VideoStreamSDK folder to your xcode project.



Search Paths

Setting	VideoStreamSample
Always Search User Paths	Yes
Header Search Paths	./VideoStreamSDK/3rd-party/ffmpeg/include/ ./AdMob/Classes/
Use Header Maps	Yes
User Header Search Paths	

./VideoStreamSDK/3rd-party/ffmpeg/include/	non-recursive
./AdMob/Classes/	non-recursive

6. Select your target again, and go to **Build Settings** tab

Find “**Header Search Paths”,** set it as below
“Header Search Paths” | **./VideoStreamSDK/3rd-party/ffmpeg/include**
./AdMob/Classes/

HOW TO USE (Paid Version)

Using VideoStreamSDK is a very easy task, it’s similar to Apple's MPMoviePlayerViewController API.

1. First of all, add **include file (#import “VSPlayerViewController.h”)** to your ViewController header file,

```
#import <UIKit/UIKit.h>
#import “VSPlayerViewController.h”

@interface MyViewController : UIViewController
```

2. Then, create and call VSPlayerViewController instance with a valid URL as below in your ViewController source file

```
//Call below for example when pressed a button
VSPlayerViewController playerVc = [[[VSPlayerViewController alloc] initWithURL:urlString
decoderOptions:NULL] autorelease];
playerVc.barTitle = [channel name];
playerVc.statusBarHidden = YES;
[self.navigationController presentViewController:playerVc animated:YES completion:NULL];
```

VSDecoder supports some options for some specific protocols and formats, the options can be pass to VSDecoder via VSPlayerViewController during initialization.
Using options and other protocol related infos are shown below,

A. RTSP

A1. RTSP Transport Options

RTSP is not technically a protocol handler in libavformat,
so it must use a lower transport protocol to connect to streaming server,
The following options are supported:

udp
Use UDP as lower transport protocol.

tcp
Use TCP (interleaving within the RTSP control channel) as lower transport protocol.

udp_multicast
Use UDP multicast as lower transport protocol.

http
Use HTTP tunneling as lower transport protocol, which is useful for passing proxies.

```
//a sample option for rtsp - setting transport layer as TCP
NSDictionary *option = [NSDictionary
dictionaryWithObject:VSDECODER_OPT_VALUE_RTSP_TRANSPORT_TCP
forKey:VSDECODER_OPT_KEY_RTSP_TRANSPORT];

VSPlayerViewController playerVc = [[[VSPlayerViewController alloc] initWithURL:urlString
decoderOptions:option] autorelease];
  ...
```

A2. RTSP Secured Streams & Authentication

VideoStream SDK supports RTSP streams with username and password authentication.
Username and your password must be provided as below (general format is rtsp://USERNAME:PASSWORD@IPADDRESS:PORT/...) and must set the RTSP

transport layer(transport layer can be "UDP" or "TCP", "UDP" is used in below sample).

```
//a sample for secure stream
NSDictionary *option = [NSDictionary
dictionaryWithObject:VSDECODER_OPT_VALUE_RTSP_TRANSPORT_UDP
forKey:VSDECODER_OPT_KEY_RTSP_TRANSPORT];
NSString *urlString = @"rtsp://username:password@192.168.1.5:554/11";

VSPlayerViewController playerVc = [[[VSPlayerViewController alloc] initWithURL:urlString
decoderOptions:option] autorelease];
  ...
```

B. MJPEG Streams

```
//for mjpeg streams, below option is a must
NSDictionary *option = [NSDictionary dictionaryWithObject:[NSNumber numberWithBool:YES]
forKey:VSDECODER_OPT_KEY_FORCE_MJPEG];

VSPlayerViewController playerVc = [[[VSPlayerViewController alloc] initWithURL:urlString
decoderOptions:option] autorelease];
  ...
```

C. OTHERS

C1. Multiple Audio Supported Streams

```
//for multiple audio streams, a default one can be set before playing by stream index OR by language
string
// If both is set, index option is high priority to other.

NSDictionary *opt1 = [NSDictionary dictionaryWithObject:[NSNumber numberWithInt:2]
forKey:VSDECODER_OPT_KEY_AUD_STRM_DEF_IDX];

NSDictionary *opt2 = [NSDictionary dictionaryWithObject:@"eng"
forKey:VSDECODER_OPT_KEY_AUD_STRM_DEF_STR];

VSPlayerViewController *playerVc = [[[VSPlayerViewController alloc] initWithURL:urlString
decoderOptions:opt1] autorelease];
```

```
...
```

C2. Logs

VideoStream SDK supports detail log mechanism. As SDK consists of some layers, each related logs can be enabled or disabled by setting log level..
To set log level,

```
//to set log level
_decodeManager = [[VSDecodeManager alloc] init];
_decodeManager.delegate = self;
[_decodeManager setLogLevel:kVSLogLevelStateChanges];
```

3. Also, you can get notifications from VSPlayerViewController about state changes and error handling. (This is optional, if you do not need to handle state change events or error handling than **skip this step**)

```
VSPlayerViewController *playerVc = [[[VSPlayerViewController alloc] initWithURLString:urlString
decoderOptions:options] autorelease];
playerVc.delegate = self;
```

Then implement below method to handle state change events and errors

```
- (void)onPlayerViewControllerStateChanged:(VSDecoderState)state errorCode:(VSError)errCode {
    if (state == kVSDecoderStateInitialized) {
    } else if (state == kVSDecoderStateConnecting) {
    } else if (state == kVSDecoderStateConnected) {
    } else if (state == kVSDecoderStateInitialLoading) {
    } else if (state == kVSDecoderStateReadyToPlay) {
    } else if (state == kVSDecoderStateBuffering) {
    } else if (state == kVSDecoderStatePlaying) {
    } else if (state == kVSDecoderStatePaused) {
    } else if (state == kVSDecoderStateStoppedByUser) {
    } else if (state == kVSDecoderStateConnectionFailed) {
    } else if (state == kVSDecoderStateStoppedWithError) {
        if (errCode == kVSErrorStreamReadError) {
        }
    }
}
```

4. ARC Supported Projects

VideoStream SDK is a non ARC library but it can used any ARC or non-ARC projects. SDK can be added to non-ARC project without doing anything.
For ARC enabled projects, after adding SDK to project,

```
-fno-objc-arc
```

> Go to Targets -> Build Phases -> Compile Sources
> double click on the right column of the row under Compiler Flags (It can also be added it to multiple files by holding the cmd button to select the files and then pressing enter to bring up the flag edit box.)
> Write above compiler flag into the box

HOW TO USE (Free Version with Ads)

Using VideoStreamSDK Free version is very easy and it is so similar to using paid version. Let's see how can you do it in just a few minutes.

1. First of all, add **include file (#import "VSPlayerViewControllerWithAd.h")** to your ViewController header file,

```
#import <UIKit/UIKit.h>
#import "VSPlayerViewControllerWithAd.h"

@interface MyViewController : UIViewController
```

2. Then, create and call VSPlayerViewControllerWithAd instance with a valid URL as below in your ViewController source file.

```
//Call below for example when pressed a button
VSPlayerViewControllerWithAd playerVc = [[[VSPlayerViewControllerWithAd alloc] initWithURL:urlString
decoderOptions:NULL] autorelease];
playerVc.barTitle = [channel name];
playerVc.statusBarHidden = YES;
[self.navigationController presentViewController:playerVc animated:YES completion:NULL];

playerVc.clientAdUnitIdBanner = @"ca-app-pub-xxxxxxxxx/xxxxxxxxxx";
playerVc.clientAdUnitIdInterstitial = @"ca-app-pub-xxxxxxx/xxxxxxxxx";
playerVc.interstitialAdFrequencyPerPlay = 2;
```

After creating VSPlayerViewControllerWithAd instance you should set 3 property values of VSPlayerViewControllerWithAd. These are :

clientAdUnitIdBanner : This is your Admob Ad Unit Id for banner advertisements. You should be very careful with entering this value correctly, otherwise SDK will not work as expected.

clientAdUnitIdInterstitial : This is your Admob Ad Unit Id for interstitial advertisements. You should be very careful with entering this value correctly, otherwise SDK will not work as expected.

interstitialAdFrequencyPerPlay : This value defines the interstitial ad frequency per video files played. For example if this value is set to 2, after each 2 video files are played the SDK will present an interstitial ad.

Decoder options and notifications usage for state changes is the same as the paid version.

CHANGELOG

Version: 3.1 – Release Date: 01/06/2016

• Updated for iOS 9 & XCode 7
• Free version with Ad Support (banner and interstitial) added

Version: 3.0 – Release Date: 01/07/2015

• Updated for iOS 8 & XCode 6
• arm64/x86_64 architectures are supported
• Code is cleaned from iOS 5 SDK checks
• iOS VideoKit engine is used under the hood
• Deployment target is set to 6.x
• ffmpeg script is updated for XCode 6.0 cmdline tools
• ffmpeg libraries are updated to latest 1.2.10 stable version
• RTP protocol is supported
• TCP & UDP protocols are supported (tcp://xxx & udp://xxx streams can be playable)
• Supports seeking in remote file streams
• Supports changing audio streams in realtime
• Supports disabling audio stream in file/realtime stream
• Supports software rendering for IPCams that don't support YUV color format
• Supports a new feature: VideoStream SDK shows the first picture immediately without waiting to fetch all frames to start
• Supports auto stop at the end for remote file streams
• Supports loops for remote file streams

Version: 2.2 – Release Date: 01/25/2014

• Updated for iOS 7 & XCode 5
> UI elements are updated with new iOS 7 elements

> AVAudioSession is used for Audio Interruption
> Sample project is updated for XCode 5.x
> Build script is updated (clang is used)

• RTP protocol is supported
• Color formats other than YUV are supported (like RGB, BGR ...)

Version: 2.1 – Release Date: 08/30/2013

• Upgraded ffmpeg library from 1.0 to 1.2.1 version
• Support audio resampling now (so SDK supports all audio codec formats, don't forget to add libswresample.a)
• Supports more decoder management for single license owners

Version: 2.0 – Release Date: 06/08/2013

• Moved from OpenGL ES 1.1 to OpenGL ES 2.0 for rendering and color conversion, this improved smoothness for especially HD streams.
• OpenGL ES 2.0 feature now makes 720p HD streams playable on iPad 1
• OpenGL ES 2.0 feature now makes real time video effects with using pixel shaders
• VSChannelListViewController now supports landscape mode.
• AV Sync log & Pkt Count log frequencies are now limited with a definition
• VSDecoder options are refactored
• Audio default stream selection feature is added
• MJPEG streams are now fully supported
• Dead stream links are replaced with the new working ones
• If no audio exist, a time later player will not get any data for HD streams - [FIXED]
• If video pkt is damaged, pkt's size is not subtracted from total size of pkt array - [FIXED]
• Code is analyzed and 2 minor leaks are fixed

Version: 1.2 – Release Date: 05/01/2013

• Supports smooth playing for 720p HD streams on iPad 2 & above (issue id: 2485)
• Upgraded ffmpeg from 0.11.1 to 1.0 version
• Updated build script & gas-preprocessor (new gas-preprocessor must be installed in /usr/local/bin folder)
• improved streaming quality for RTSP streams via UDP transport

Version: 1.15 – Release Date: 04/24/2013

• Support non-english urls (issue id: 2619)
• Video resolution issue for some streams is fixed

Version: 1.13 – Release Date: 04/05/2013

• RTSP protocol supports "tcp", "udp", "udp_multicast", and "http" transport layers (feature request id: 2410)
• SDK state change events are updated
• SDK supports video-only streams (important feature for ip-cams)
• Build script is updated
• Some small bugs fixed

Version: 1.11 – Release Date: 02/25/2013

• SDK can stream damaged divx & xvid video files

Version: 1.0 – Release Date: 02/18/2013

• First initial release