VideoKit SDK

for iOS & tvOS

Play Video without boundaries

Elma DIGIT@L

email support@iosvideokit.com

website www.iosvideokit.com

last updated : Dec 29, 2020

INTRODUCTION

Apple's video playing and streaming solution is very limited, it supports only playing specific video file formats, http streaming & h264/h265 video codecs and its source is closed rendering the task of streaming and playing video a very hard one.
If you need a custom video player that play divx, mkv, flv, etc… or uses other popular streaming protocols such as MMS, RTSP, RTMP, secure protocols or if you play video files/stream from an http server but using different audio/video codecs or you play video files/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 **VideoKit**

FEATURES

- iOS and tvOS support
- Bitcode support
- Xcode 11 and iOS13 support
- Play most popular media files locally that Apple doesn't support
- Stream from popular protocols (http, mms, rtsp & rtmp)
- Supports secure streams with openSSL support (https, rtmps, rtmpe, rtmpte, rtsps & tls)
- Supports all popular audio & video codecs
- Supports mjpeg streams - mostly for ipcams
- Supports recording functionality (No transcoding, record as it is)
- Supports animated GIF files with full transparency
- Supports real time video effects with using pixel shaders
- Successful Audio & video syncing
- Very easy to use (very similar to Apple's MPMoviePlayerViewController API & MPMoviePlayerController)
- Look & feel like Apple's MPMoviePlayerViewController & MPMoviePlayerController
- Works with Wifi & 3G,LTE
- Shows detailed information about stream (audio & video codecs,total streamed data in bytes, connection type)
- Works on these iOS devices (iPhone 5, 6, 6+, 6s, 6s+,7, 7+, 8, 8+, X, 11, and all iPad minis, iPads) devices & supports all screen types and rotations
- Works on new AppleTV (4th Generation & above)
- Supports pausing stream
- Supports streaming in background
- Robust error handling
- Supports audio resampling
- Supports seeking in local & remote file streams
- Airplay
- Showing files/streams both in fullscreen and embedded
- Supports multiple players on same view
- Player is an NSObject instance, so can be used without UI
- Supports transition from embedded/fullscreen to fullscreen/embedded

- Supports fullscreen, embedded and non UI control modes
- Supports initial playback time for files
- Supports looping for files
- Supports adjusting volume level for each player
- Supports changing audio streams in realtime
- Supports disabling audio stream in file/stream
- Supports pinch in/out to zoom in/out in any where of video
- Support autolayout for both iOS & tvOS players
- Support HW Video acceleration for h264 codec
- Support HW Audio acceleration for ac3, aac, eac3, amr, and mp3 codecs
- Custom I/O support (support playing audio/video stream in memory)
- Shoutcast/Icy metadata support
- Supports changing audio streams in realtime
- iPhone X user interface support
- Support HW Video acceleration for h265 codec (for devices that has hardware decoder chip for h265)

REQUIREMENTS

iOS **10.x SDK** or above is needed to compile VideoKit however, if you need under 10.x, VideoKit can be built with small modifications.

The VideoKit frameworks
for iOS requires the following Apple frameworks:
(Note : If You create a new project with Xcode 10.x, then you do not need to add these)

- 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**)
- VideoToolbox (using **Hardware video acceleration**)
- CoreVideo (used for some structures for **Hardware video acceleration**)
- CoreMedia (used for some structures for **Hardware video acceleration**)
- GLKit (using to render **Portrait videos**)
- CoreImage (using to capture original size video frames)

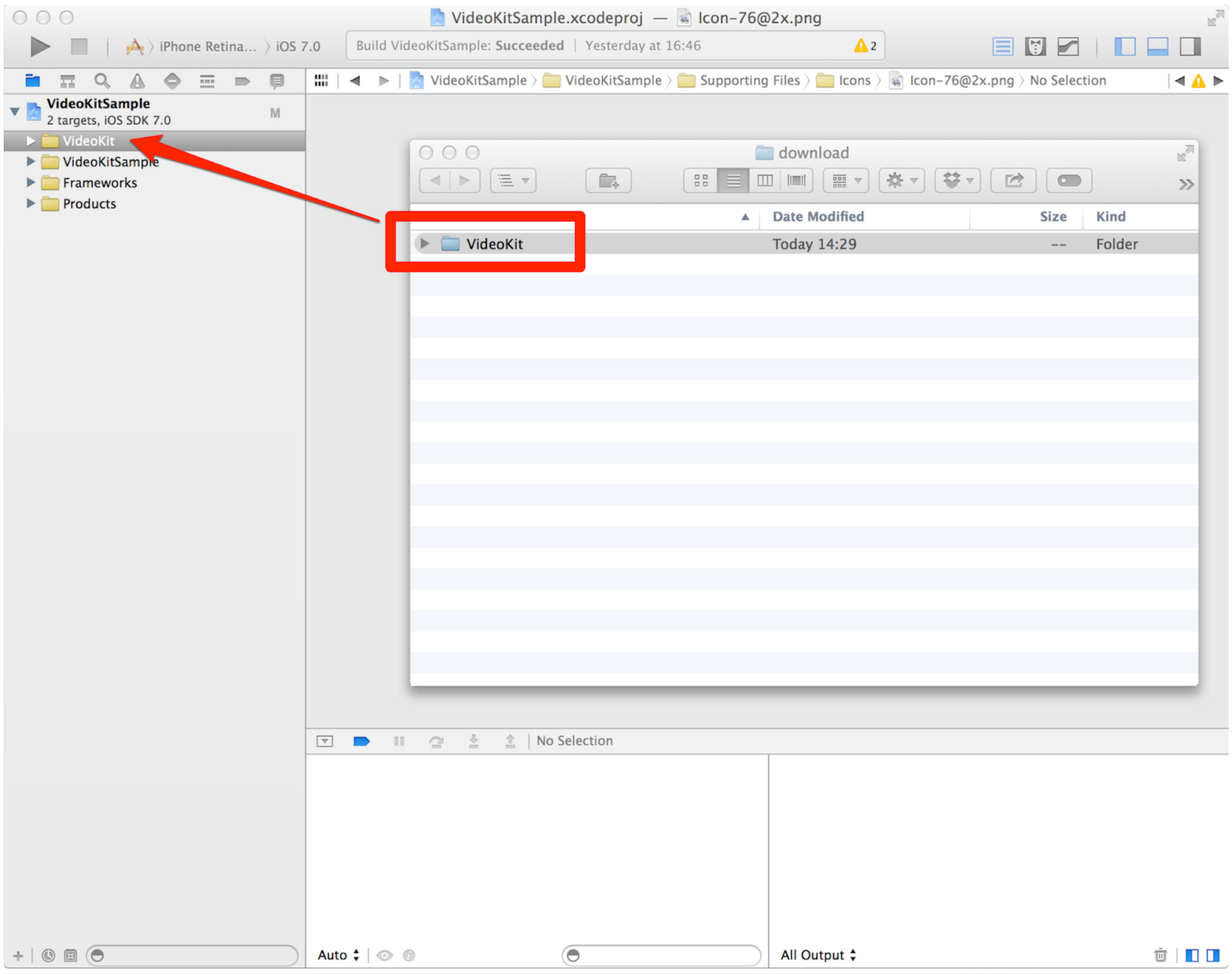
and VideoKit also needs to these frameworks (for connecting to stream, decoding Audio&Video packets, etc...):

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

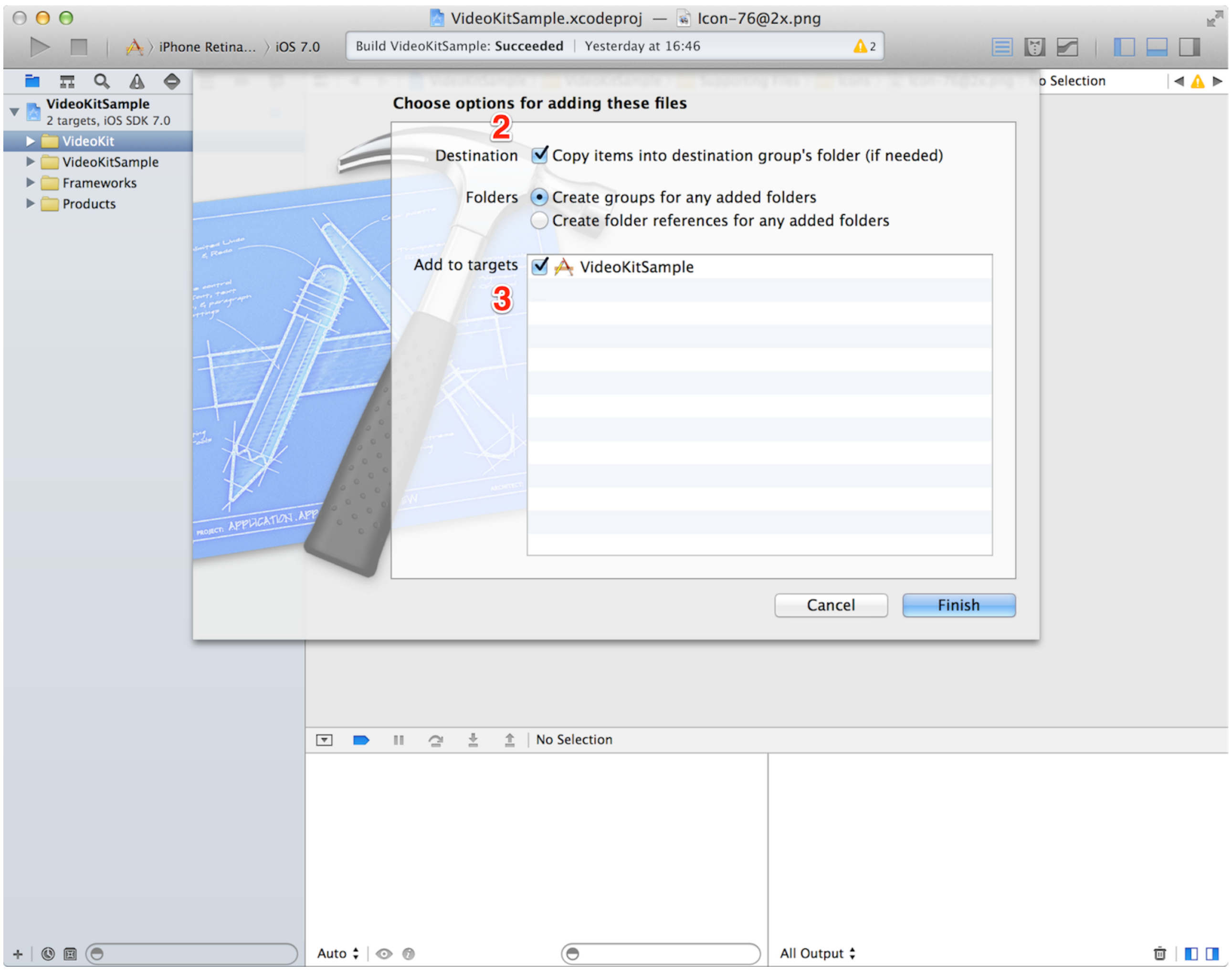
- libswresample
- libssl
- libcrypto

INTEGRATION for iOS version

Adding VideoKit to your Xcode project


VideoKitSample.xcodeproj — Icon-76@2x.png
iPhone Retina... iOS 7.0
Build VideoKitSample: Succeeded | Yesterday at 16:46
VideoKitSample > VideoKitSample > Supporting Files > Icons > Icon-76@2x.png > No Selection
VideoKitSample
2 targets, iOS SDK 7.0
VideoKit
VideoKitSample
Frameworks
Products
download
Date Modified
Size
Kind
VideoKit
Today 14:29
--
Folder
No Selection
Auto
All Output

1. Go to the VideoKit folder, and move the VideoKit folder to your Xcode project. (Please note that, VideoKit folder must be added to **root** not under any folder)

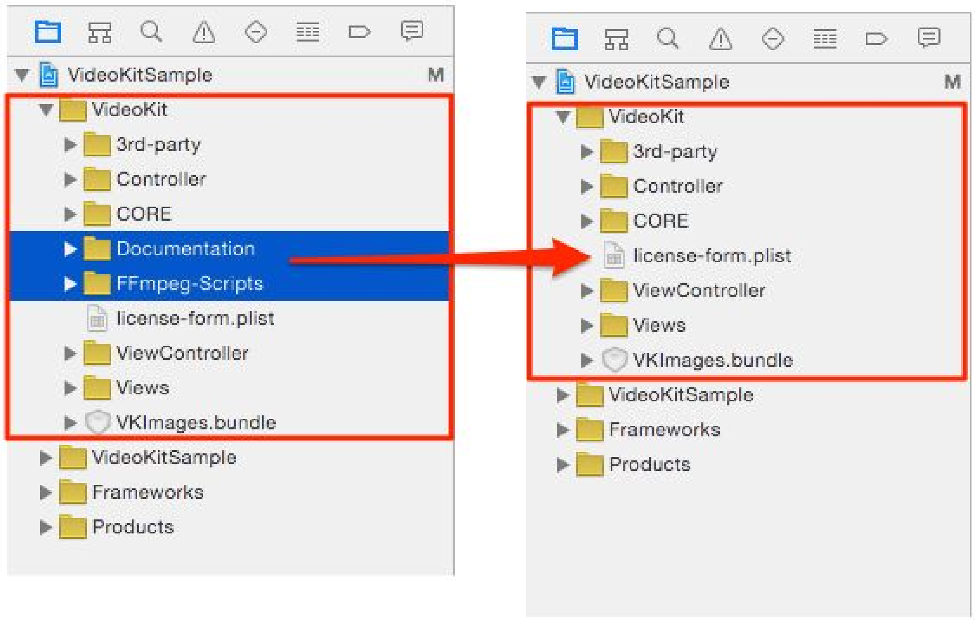

Choose options for adding these files

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

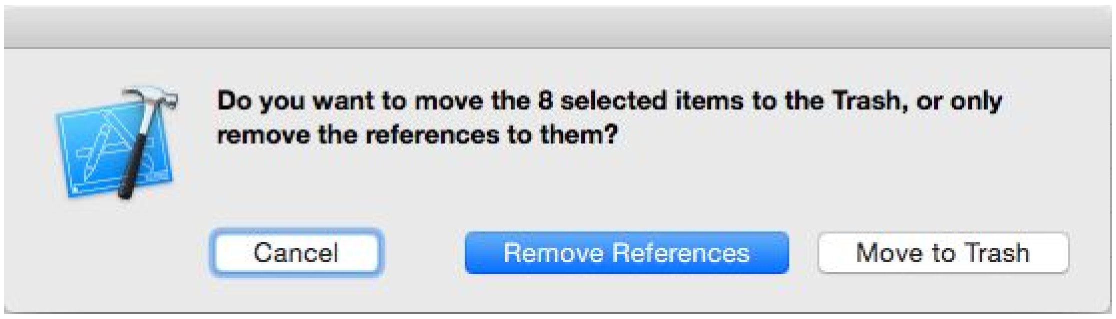
Folders [● Create groups for any added folders] ○ Create folder references for any added folders

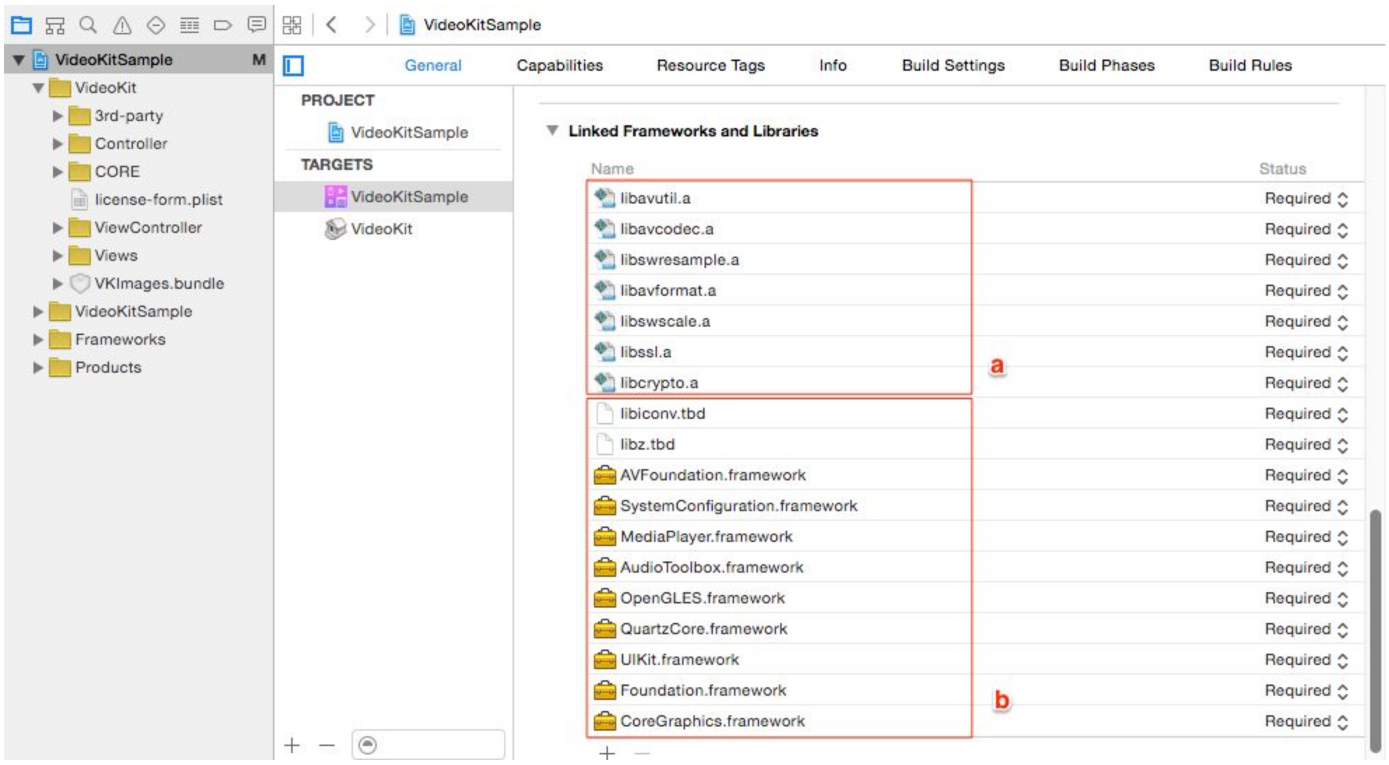
Add to targets ✓ VideoKitSample

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



4. Remove selected unnecessary folders (Documentation, FFmpeg-Scripts). When asking about the removal process, select 'Remove References'. After doing this, our VideoKit folder is seen like above (at right).


Do you want to move the 8 selected items to the Trash, or only remove the references to them?



Linked Frameworks and Libraries

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

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

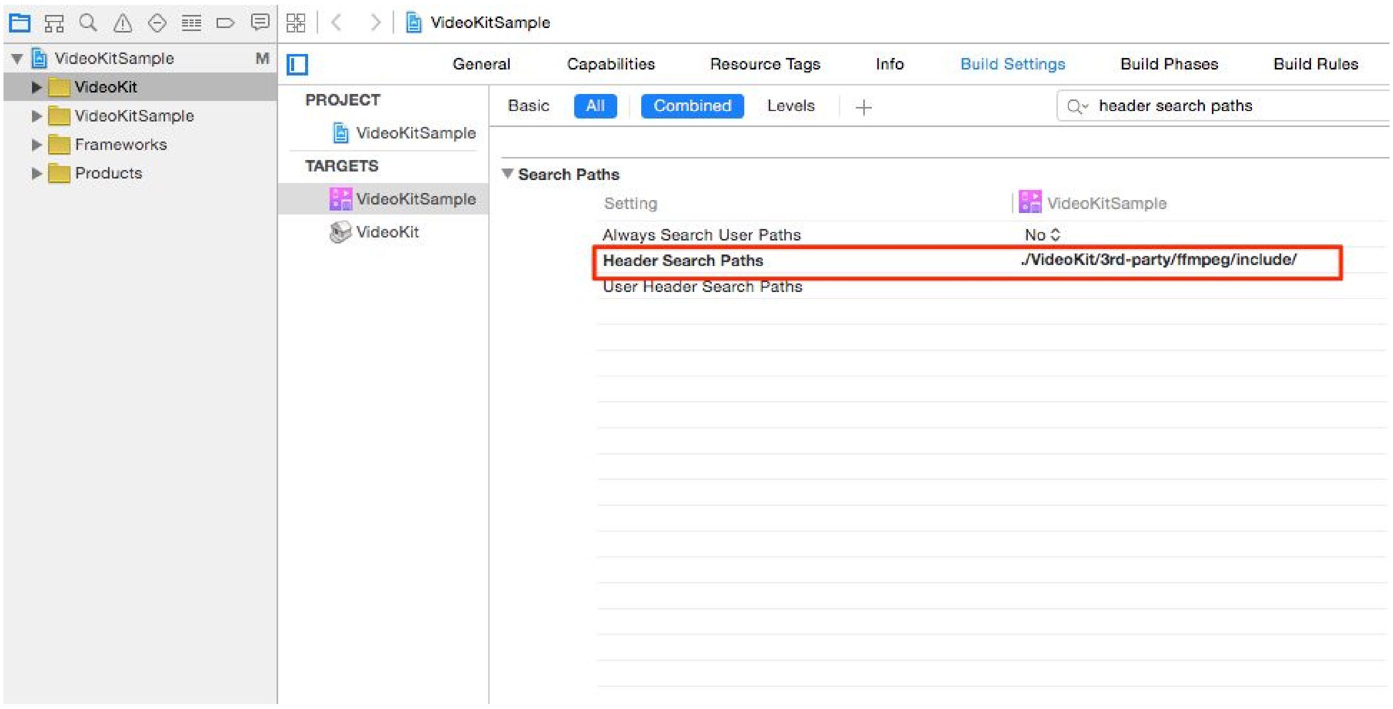
a. Those libraries (located in a rectangle labelled as "a" in above picture) are **automatically added** when you add the VideoKit folder to your xcode project.

b. Also add these frameworks/libraries

1. libz
2. Libiconv

(Note : If You create a new project with Xcode 9.x, then you do not need to add below, since they are automatically added)

3. QuartzCore
4. SystemConfiguration
5. MediaPlayer
6. OpenGLES
7. AudioToolbox
8. AVFoundation
9. VideoToolbox
10. CoreVideo
11. CoreMedia
12. GLKit
13. CoreImage

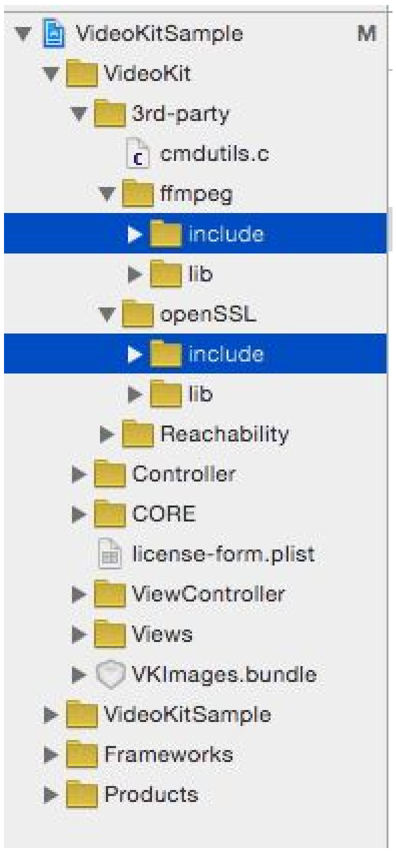


Setting	VideoKitSample
Always Search User Paths	No
Header Search Paths	./VideoKit/3rd-party/ffmpeg/include/
User Header Search Paths	

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

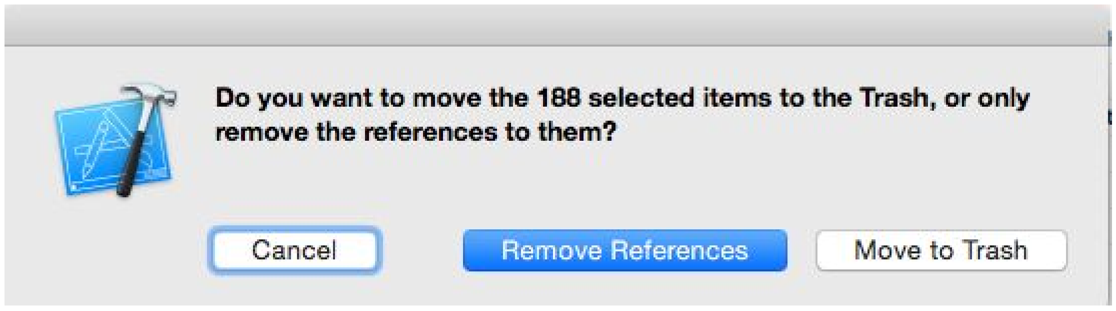
Find "**Header Search Paths",** set it as below

./VideoKit/3rd-party/ffmpeg/include



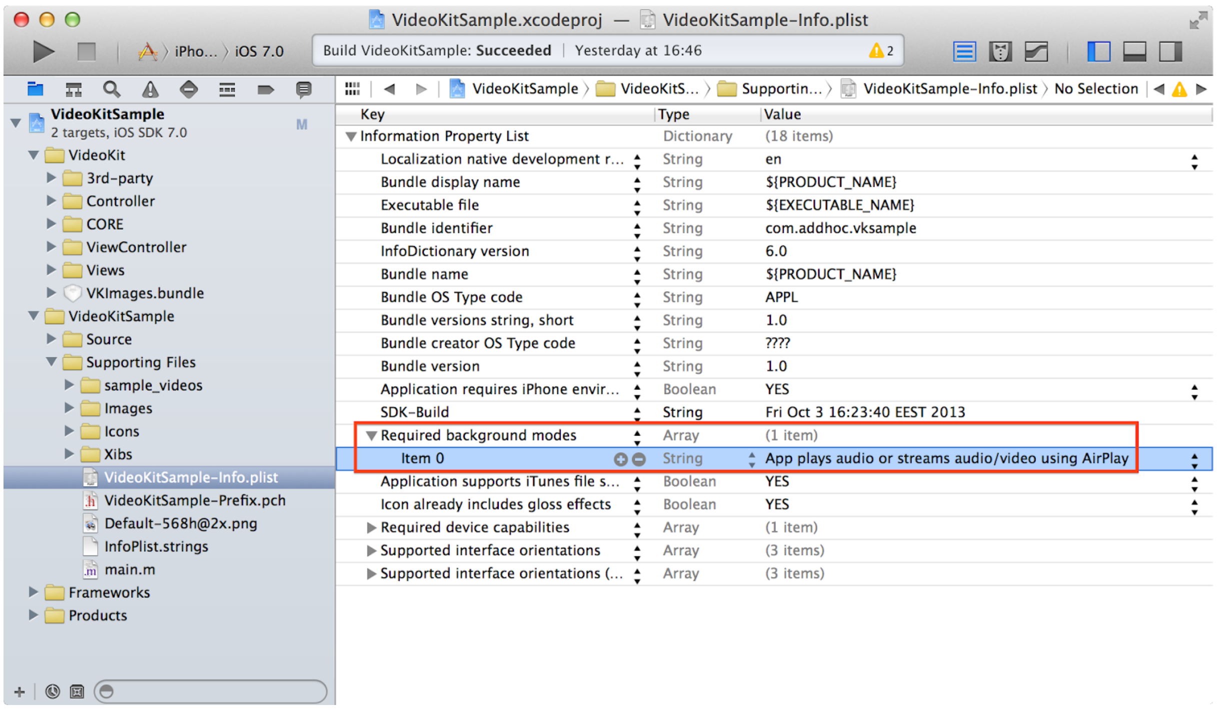
7. Delete include folders of 3rd party libraries from Project Explorer :

Select 3rd-party/ffmpeg/include & 3rd-party/openSSL/include and then remove reference to these folders.



Do you want to move the 188 selected items to the Trash, or only remove the references to them?

When dialog appears, select Remove References option.


VideoKitSample.xcodeproj — VideoKitSample-Info.plist

iPho… 〉 iOS 7.0 | Build VideoKitSample: **Succeeded** | Yesterday at 16:46 | 2

VideoKitSample 〉 VideoKitS… 〉 Supportin… 〉 VideoKitSample-Info.plist 〉 No Selection

```
VideoKitSample                                   M
2 targets, iOS SDK 7.0
  VideoKit
    3rd-party
    Controller
    CORE
    ViewController
    Views
    VKImages.bundle
  VideoKitSample
    Source
    Supporting Files
      sample_videos
      Images
      Icons
      Xibs
      VideoKitSample-Info.plist
      VideoKitSample-Prefix.pch
      Default-568h@2x.png
      InfoPlist.strings
      main.m
  Frameworks
  Products
```

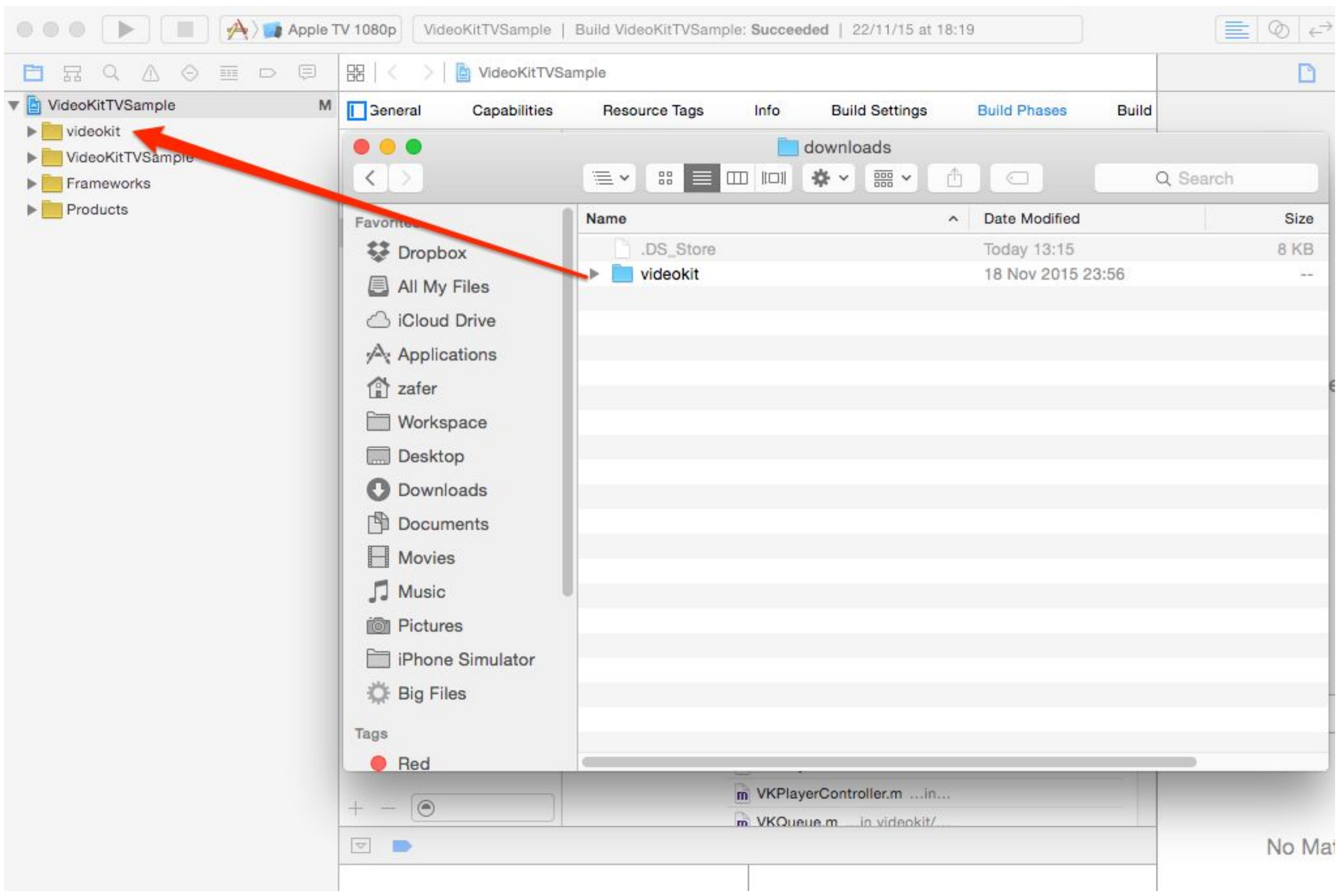
Key	Type	Value
Information Property List	Dictionary	(18 items)
Localization native development r…	String	en
Bundle display name	String	${PRODUCT_NAME}
Executable file	String	${EXECUTABLE_NAME}
Bundle identifier	String	com.addhoc.vksample
InfoDictionary version	String	6.0
Bundle name	String	${PRODUCT_NAME}
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 envir…	Boolean	YES
SDK-Build	String	Fri Oct 3 16:23:40 EEST 2013
Required background modes	Array	(1 item)
Item 0	String	App plays audio or streams audio/video using AirPlay
Application supports iTunes file s…	Boolean	YES
Icon already includes gloss effects	Boolean	YES
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)
Supported interface orientations (…	Array	(3 items)

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

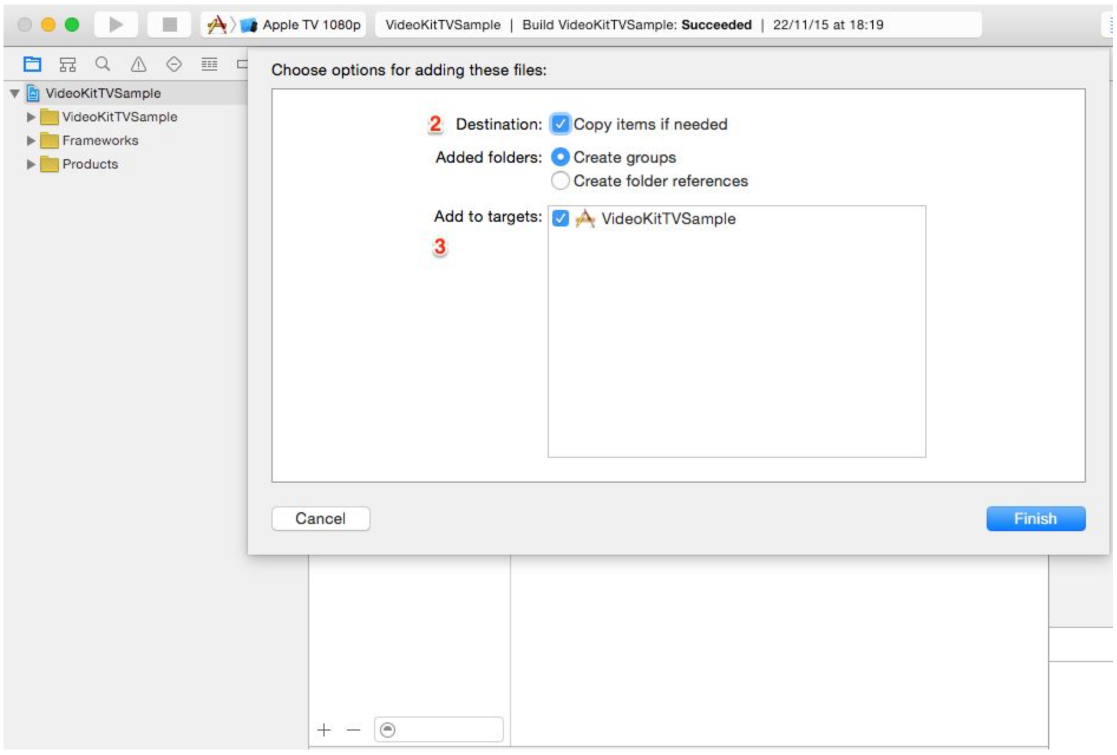
Integration is done, your project should be compiled without any error now. (If your project is ARC enabled, then please see section TN6 (**ARC Supported Projects**) in below)

INTEGRATION for tvOS version

Adding VideoKit to your Xcode project



1. Go to the videokit folder, and move the videokit folder to your Xcode project. (Please note that, videokit folder must be added to **root** not under any folder)


Apple TV 1080p | VideoKitTVSample | Build VideoKitTVSample: **Succeeded** | 22/11/15 at 18:19

Choose options for adding these files:

2 Destination: [✓] Copy items if needed

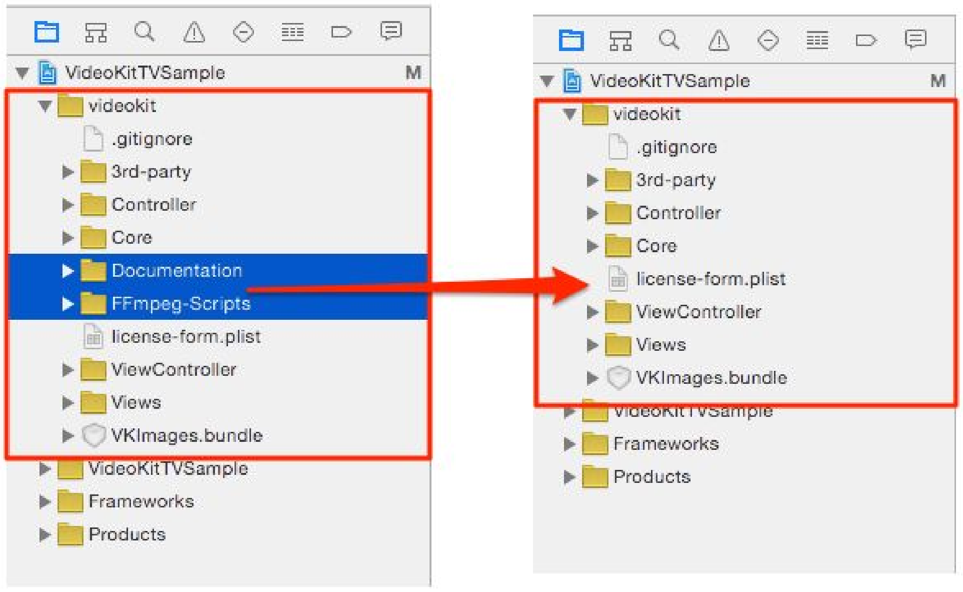
Added folders: [● Create groups] ○ Create folder references

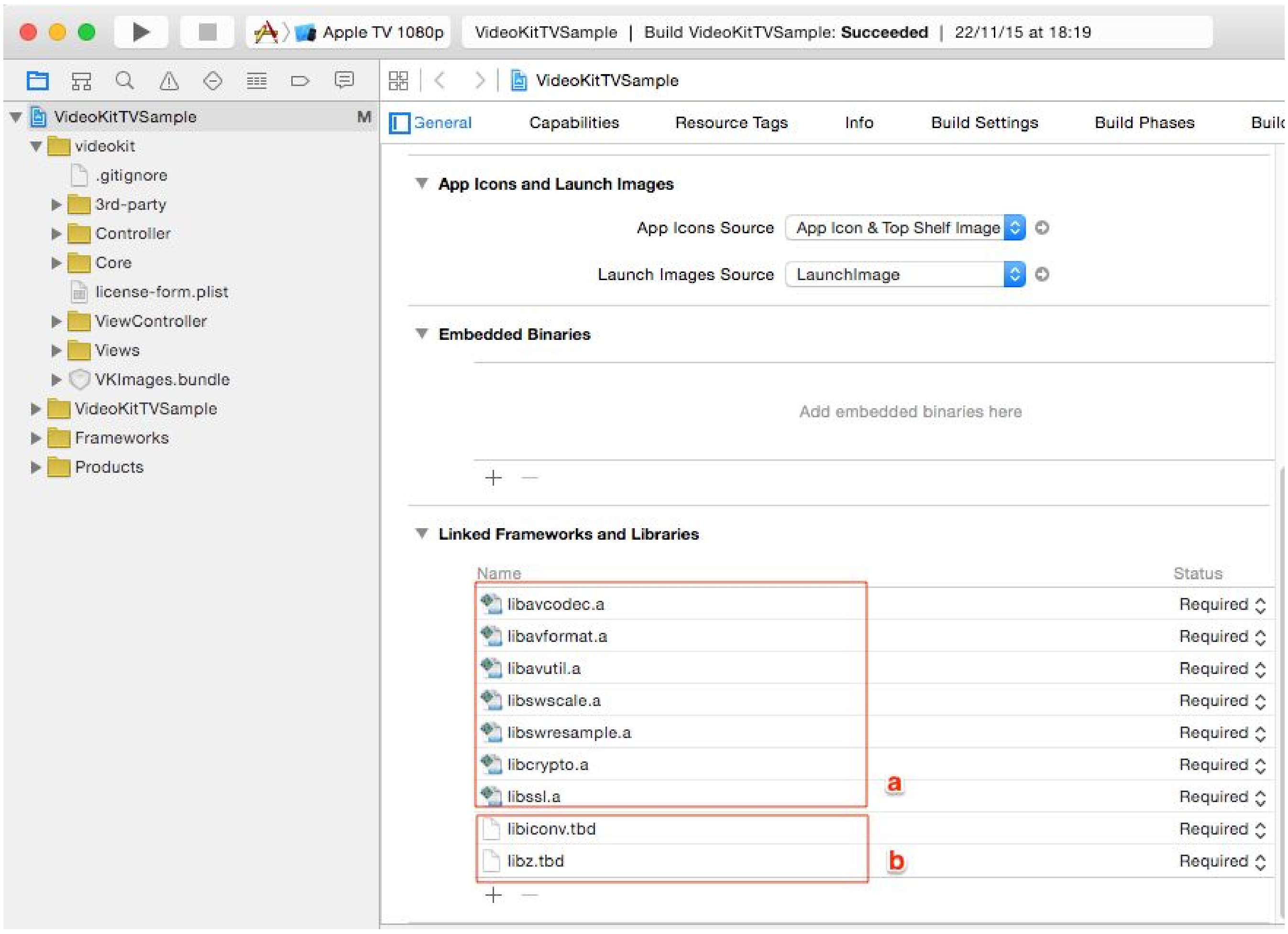
Add to targets: [✓] VideoKitTVSample
3

Cancel | Finish

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

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





App Icons Source: App Icon & Top Shelf Image

Launch Images Source: LaunchImage

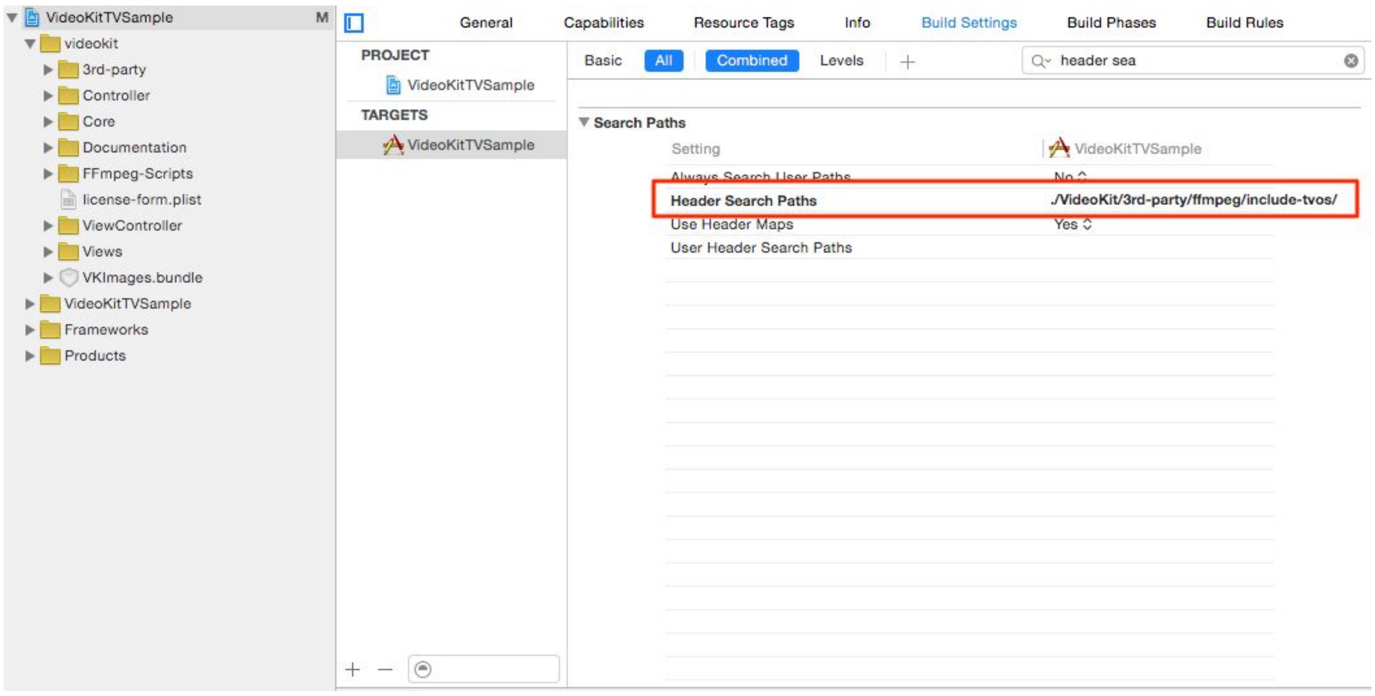
Name	Status
libavcodec.a	Required
libavformat.a	Required
libavutil.a	Required
libswscale.a	Required
libswresample.a	Required
libcrypto.a	Required
libssl.a	Required
libiconv.tbd	Required
libz.tbd	Required

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

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

b. Also add these libraries

1. libz
2. libiconv

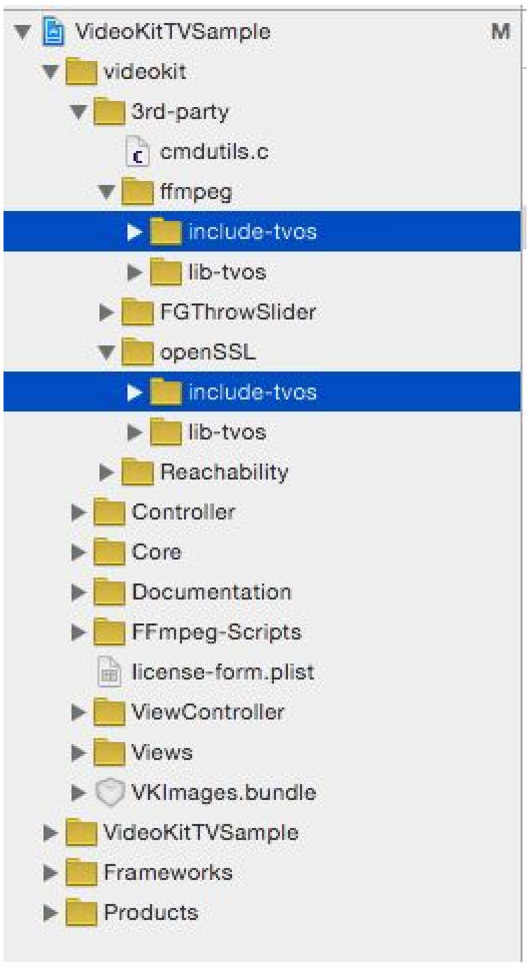


Setting	VideoKitTVSample
Always Search User Paths	No
Header Search Paths	./VideoKit/3rd-party/ffmpeg/include-tvos/
Use Header Maps	Yes
User Header Search Paths	

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

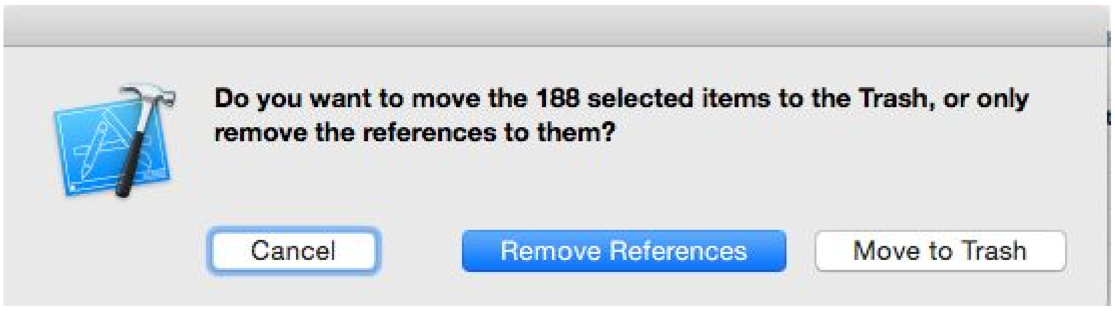
Find “**Header Search Paths”,** set it as below

./VideoKit/3rd-party/ffmpeg/include-tvos



7. Delete include folders of 3rd party libraries from Project Explorer :

Select 3rd-party/ffmpeg/include-tvos & 3rd-party/openSSL/include-tvos and then remove reference to these folders.


Do you want to move the 188 selected items to the Trash, or only remove the references to them?

When dialog appears, select Remove References option.

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)

HOW TO USE

Using VideoKit is a very easy task, it’s similar to Apple's MPMoviePlayerViewController (Apple’s native fullscreen player) & MPMoviePlayerController (Apple’s native embedded player) API.

Note: Code examples are all in Objective-C but You can find a full project (VideoKitSampleTRIAL) in Swift language on our website.

A. Using **VKPlayerViewController** (Full screen player like Apple’s MPMoviePlayerViewController)

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

For iOS & tvOS

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

2. Then, create and call VKPlayerViewController with a valid URL as below in your ViewController.m file

For iOS & tvOS

```
//Call below code,  for example when pressed a button
 NSString *urlString = @"http://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:NULL] autorelease];
[self presentViewController:playerVc animated:YES completion:NULL];
```

B. Using **VKPlayerController** (Embedded player like Apple’s MPMoviePlayerController)

1. First of all, add **include file (#import “VKPlayerController.h”)** & VKPlayerController object property to your ViewController.h file,

For iOS

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

@interface MyViewController : UIViewController

@property (nanotomic, retain) VKPlayerController *player;
```

For tvOS

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

@interface MyViewController : UIViewController

@property (nanotomic, retain) VKPlayerControllerTV *player;
```

2. Then, synthesize the property and call VKPlayerController with a valid URL as below in your ViewController.m file

For iOS

```
@implementation MyViewController
@synthesize player = _player;

.
.
.

//Call below code, for example when pressed a button
NSString *urlString = @"http://xxx.xxx.xxx.xxx";
if (!_player) {
        self.player = [[[VKPlayerController alloc] initWithURLString:urlString] autorelease];
        [self.view addSubview:_player.view];

        UIView *playerView = _player.view;
        playerView.translatesAutoresizingMaskIntoConstraints = NO;

        // align playerView from the left
        [self.view addConstraints:[NSLayoutConstraint
constraintsWithVisualFormat:@"H:|-28-[playerView(==264)]" options:0 metrics:nil
views:NSDictionaryOfVariableBindings(playerView)]];
        // align playerView from the top
        [self.view addConstraints:[NSLayoutConstraint
constraintsWithVisualFormat:@"V:|-38-[playerView(==167)]" options:0 metrics:nil
views:NSDictionaryOfVariableBindings(playerView)]];
}

_player.contentURLString = urlString;
_player.decoderOptions = NULL;
[_player play];
```

For tvOS

```
@implementation MyViewController
@synthesize player = _player;


.
.
.

//Call below code, for example when pressed a button
NSString *urlString = @"http://xxx.xxx.xxx.xxx";
if (!_player) {
        self.player = [[[VKPlayerControllerTV alloc] initWithURLString:urlString] autorelease];
        [self.view addSubview:_player.view];

        UIView *playerView = _player.view;
        playerView.translatesAutoresizingMaskIntoConstraints = NO;

        // align playerView from the left
        [self.view addConstraints:[NSLayoutConstraint
constraintsWithVisualFormat:@"H:|-161-[playerView(==651.0)]" options:0 metrics:nil
views:NSDictionaryOfVariableBindings(playerView)]];
        // align playerView from the top
        [self.view addConstraints:[NSLayoutConstraint
constraintsWithVisualFormat:@"V:|-204-[playerView(==407.0)]" options:0 metrics:nil
views:NSDictionaryOfVariableBindings(playerView)]];
}

_player.contentURLString = urlString;
_player.decoderOptions = NULL;
[_player play];
```

PROTOCOLS & CODECS

VKDecoder supports some options for some specific protocols and formats, the options can be passed to VKDecoder via VKPlayerViewController during initialization and via options property which can be set for VKPlayerController.
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 a lower transport protocol.

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

udp_multicast
Use UDP multicast as a lower transport protocol.

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

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

NSString *urlString = @"rtsp://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:option] autorelease];
[self presentViewController:playerVc animated:YES completion:NULL];
...
```

A2. RTSP Secured Streams & Authentication

VideoKit 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:VKDECODER_OPT_VALUE_RTSP_TRANSPORT_UDP
forKey:VKDECODER_OPT_KEY_RTSP_TRANSPORT];

NSString *urlString = @”rtsp://username:password@192.168.1.5:554/11”;
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:option] autorelease];
[self presentViewController:playerVc animated:YES completion:NULL];
...
```

B. MJPEG Streams

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

NSString *urlString = @"http://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:option] autorelease];
[self presentViewController:playerVc animated:YES completion:NULL];
...
```

C. RTMP

VideoKit supports rtmp streaming, ffmpeg has many server configuration parameters for rtmp streaming protocol. These parameters can be passed through ffmpeg by using VKDECODER_OPT_KEY_PASS_THROUGH as below,

An example of rtmp streaming url,
rtmp://95.211.148.203/live/aflam4youddd?id=152675 -rtmp_swfurl
http://mips.tv/content/scripts/eplayer.swf -rtmp_live live -rtmp_pageurl
http://mips.tv/embedplayer/aflam4youddd/1/600/380 -rtmp_conn S:OK

```
//Add sample channel to channel list with using VKDECODER_OPT_KEY_PASS_THROUGH
Channel *c14 = [Channel channelWithName:@"Aflam Live TV"
addr:@"rtmp://95.211.148.203/live/aflam4youddd?id=152675 -rtmp_swfurl
http://mips.tv/content/scripts/eplayer.swf -rtmp_live live -rtmp_pageurl
http://mips.tv/embedplayer/aflam4youddd/1/600/380 -rtmp_conn S:OK" description:@"Pass through
params - ffplay style" localFile:NO options:[NSDictionary dictionaryWithObject:@"1"
forKey:VKDECODER_OPT_KEY_PASS_THROUGH]];
 [_streamList addObject:c14];
```

For more information, please see our blog post about rtmp streaming in here

D. Cache protocol

Since beginning version 2.6, VideoKit can enable cache protocol in ffmpeg. When cache option is set as a decoder option (see VKPlayerControllerBase.h file), then packets coming from network are saved to a temp file and If user wants to watch a section watched before, then VideoKit will not request any packets from network, indeed VideoKit will play that part from the temporary saved file.

Cache protocol is currently for http and file streams and can not be used real time streams.

Example usage:

```
//a sample option for cache protocol
NSDictionary *option = @{VKDECODER_OPT_KEY_CACHE_STREAM_ENABLE: @(1)};

NSString *urlString = @"http://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:option] autorelease];
[self presentViewController:playerVc animated:YES completion:NULL];
...
```

E. Custom I/O data support

Till VideoKit 2.7 version, VideoKit needs a file url or network url to play an audio/video stream. But, with the version 2.7, It's possible to provide custom audio/video stream data to VideoKit without giving an url. VideoKit can play any audio/video stream in the memory.

We include a new viewcontroller(CustomIOViewController.m) into our sample project In order to show how the custom I/O feature is used. To simulate a/v stream data in memory, We first read the video file's a/v data into memory and play the data stream in memory not from the file url.

The usage:

```
if (!_player) {
        self.player = [[[VKPlayerController alloc] init] autorelease];
        [self.view addSubview:_player.view];

        UIView *playerView = _player.view;
        playerView.translatesAutoresizingMaskIntoConstraints = NO;

        // align playerView from the left
        [self.view addConstraints:[NSLayoutConstraint
constraintsWithVisualFormat:@"H:|-28-[playerView(==264)]" options:0 metrics:nil
views:NSDictionaryOfVariableBindings(playerView)]];
        // align playerView from the top
        [self.view addConstraints:[NSLayoutConstraint
constraintsWithVisualFormat:@"V:|-38-[playerView(==167)]" options:0 metrics:nil
views:NSDictionaryOfVariableBindings(playerView)]];
    }
    _player.decoderOptions = options;

    _player.enableCustomIO = YES;    // (1)
    _player.customIODelegate = self;  // (2)
```

The bold lines are important to enable customIO usage (1) and provide data to VideoKit (2).

At least below required method must be implemented to make customIO feature work successfully,

```
/**
 *  Required delegate method, add this method to your viewcontroller to provide A/V stream data to ffmpe
 *
 *  @param player  The player object that owns the notification
 *  @param data   Indicates the data pointer to be filled
 *  @param size Indicates the size of the data pointer to be filled
 */
- (int)player:(VKPlayerControllerBase *)player ioStreamRead:(uint8_t *)data size:(int)size;
```

To make the usage easy, We have also implemented a special structure for customIO feature in the player,

```
typedef struct VKPlayerCustomIO {
    ///The customIO descriptor to hold custom IO data pointer.
    void *customIODescriptor;

    ///The customIO total size to be used for moving pointer on custom IO descriptor
    size_t customIOSize;

    ///The customIO last byte read index to be used for moving pointer on custom IO descriptor
    unsigned long lastByteIndex;

    ///The data used for passing argument to read_data method callback
    void *opaque;

} VKPlayerCustomIO ;
```

Then, It will be easy to fill **data** variable, hold the last read byte index of data in memory and return the read size.

```
- (int)player:(VKPlayerControllerBase *)player ioStreamRead:(uint8_t *)data size:(int)size {

    size_t lenFinal = 0;
    size_t bytesRemaining = player.customIO->customIOSize - player.customIO->lastByteIndex;

    if (size < bytesRemaining) {
        lenFinal = size;
    } else {
        lenFinal = bytesRemaining;
    }

    void *customIOBuffer = (void *)player.customIO->customIODescriptor;
    memcpy(data, &customIOBuffer[player.customIO->lastByteIndex], lenFinal);
    player.customIO->lastByteIndex += (unsigned int)lenFinal;

    return (int)lenFinal;
}
```

If desired, other optional delegate methods can be implemented,

```
/**
 *  Optional delegate method, add this method to your viewcontroller to be able to seek in your A/V stream
 *
 *  @param player  The player object that owns the notification
 *  @param offset   Indicates the offset bytes of data
 *  @param whence Indicates the size of the data pointer to be filled
 *  Important : Even if You don't want to implement A/V stream seeking functionality, this method must be
implemented with a "-1" return value
 */
- (UInt64)player:(VKPlayerControllerBase *)player ioStreamSeek:(uint64_t)offset whence:(int)whence;

/**
 *  Optional delegate method, this method is a helper method and it's good for including the custom openi
socket/file etc. code and this method must return 0 for successful opening
 *  @param player  The player object that owns the notification
 */
- (VKError)ioStreamOpenForPlayer:(VKPlayerControllerBase *)player;

/**
 *  Optional delegate method, this method is a helper method and it's good for including the custom closin
socket/file etc. code
 *  @param player  The player object that owns the notification
 */
- (void)ioStreamCloseForPlayer:(VKPlayerControllerBase *)player;
```

How to use these methods are already shown in CustomIOViewController.m file in our sample project. Please see the source code for more information about the delegate methods.

F. Shoutcast/ICY metadata support

What is shoutcast ?

The SHOUTcast software uses a client–server model, with each component communicating via a network protocol that intermingles audio or video data with metadata such as song titles and the station name. It uses HTTP as a transport protocol (ref: https://en.wikipedia.org/wiki/SHOUTcast).

Beginning with the 2.8 version, VideoKit is able to handle shoutcast or icy servers' metadata information.

First of all, below property of player object in VKPlayerController class type must be set to YES to enable fetching metadata information from the shoutcast/icy servers.

```
///Specify YES to enable metadata information for http shoutcast/icy protocols, default is NO.
@property (nonatomic, assign) BOOL enableICYMetadata;
```

Then, if the metadata is applicable, below data is parsed,

```
@interface VKICYMetadata : NSObject

///Initialize VKICYMetadata class object instance with title and headers in raw string format (unformatted)
- (id)initWithTitle:(NSString *)title headersRaw:(NSString *)headersRaw;

///Holds stream title information
@property (nonatomic, readonly) NSString *title;

///Holds stream bitrate information
@property (nonatomic, readonly) NSString *bitrate;

///Holds stream description information
@property (nonatomic, readonly) NSString *desc;

///Holds stream genre information
@property (nonatomic, readonly) NSString *genre;

///Holds stream name information
@property (nonatomic, readonly) NSString *name;

///Holds stream pub information
@property (nonatomic, readonly) NSString *pub;

///Holds stream url information
@property (nonatomic, readonly) NSString *url;

@end
```

The object in VKICYMetadata class type will be passed with a delegation method which is below,

Using **VKPlayerViewController,**

```
/**
 *  Optional delegate method, add this method to your viewcontroller if you want to be notified about meta
information for http shoutcast/icy streams
 *
 *  @param icyMetadata   Holds the metadata information as an instance of VKICYMetadata class
 */
- (void)onPlayerViewControllerDidICYMetadataChanged:(VKICYMetadata *)icyMetadata;
```

Using **VKPlayerController,**

```
/**
 *  Optional delegate method, add this method to your viewcontroller if you want to be notified about meta
information for http shoutcast/icy streams
 *
 *  @param player  The player object that owns the notification
 *  @param icyMetadata   Holds the metadata information as an instance of VKICYMetadata class
 */
- (void)player:(VKPlayerControllerBase *)player didICYMetadataChanged:(VKICYMetadata *)icyMetadata;
```

As a result, the media title, description, and some other information can be retrieved, and later if any change is happened about the media's metadata, the delegation method is called again for the update.

Note: With the help of itunes Search API (https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/), the cover image of media can be searched and found using media's title.

TECHNICAL NOTES

TN1. 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:VKDECODER_OPT_KEY_AUD_STRM_DEF_IDX];

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

NSString *urlString = @"rtsp://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:opt1] autorelease];
  ...
```

TN2. Logs

VideoKit supports advanced & configurable log mechanism. As Kit consists of some layers, each related logs can be enabled or disabled by setting log level.. To set/change log level, find below lines in **play** method in **VKPlayerControllerBase.m** file, and set the log level as below

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

TN3. Secure links

Since version 2.0, secure streams are supported by VideoKit, https, rtmps, rtmpe, rtmpte, rtmpte & tls (and with 2.4, rtsps protocol is also supported) protocols are playable with the support of **openSSL** library. If you are using ffmpeg libraries that are different from ones found in sample project, and want to play secure streams, please check the build scripts …

TN4. Recording functionality

Since version 2.1, recording functionality is added to VideoKit (please note that, this feature is not available for some license types), by default this functionality is disabled, it can be enabled as below,

when VKPlayerViewController is used, then

```
NSString *urlString = @"http://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:NULL] autorelease];
playerVc.recordingEnabled = YES;
[self presentViewController:playerVc animated:YES completion:NULL];
```

when VKPlayerController is used, then

For iOS

```
NSString *urlString = @"http://xxx.xxx.xxx.xxx";
self.player  =  [[[VKPlayerController alloc] initWithURLString:urlString] autorelease];
player.view.frame = CGRectMake(28.0, 38.0, 264.0, 167.0);
[self.view addSubview:player.view];
player.recordingEnabled = YES;
[player play];
```

For tvOS

```
NSString *urlString = @"http://xxx.xxx.xxx.xxx";
self.player  =  [[[VKPlayerControllerTV alloc] initWithURLString:urlString] autorelease];
player.view.frame = CGRectMake(28.0, 38.0, 264.0, 167.0);
[self.view addSubview:player.view];
player.recordingEnabled = YES;
[player play];
```

iOS only,

Beginning with the version 2.4, even the stream has codecs which are compatible with iOS (see the whole list of compatible video formats by iOS in here), the video can not be recorded in a mp4 container. Thus, recorded video can not be saved to iOS camera roll and can not be played with iOS native video layer. Since beginning VideoKit version 2.4, ffmpeg’s **aac_adtstoasc** bitstream filter is supported and this filter let VideoKit record supported codecs in mp4 container. If you want to save the recorded video to camera roll, below sample code may be helpful,

```
// don't forget to link to the AssetsLibrary framework
// and also #import <AssetsLibrary/AssetsLibrary.h>

ALAssetsLibrary *library = [[ALAssetsLibrary alloc] init];

NSString *filePathString = [NSString stringWithFormat:@"%@/%@",tmpDirectory,itemName];
NSURL *filePathURL = [NSURL fileURLWithPath:filePathString isDirectory:NO];

if ([library videoAtPathIsCompatibleWithSavedPhotosAlbum:filePathURL]) {
    [library writeVideoAtPathToSavedPhotosAlbum:filePathURL completionBlock:^(
                                                    NSURL *assetURL, NSError *error){
        if (error) {
            NSLog(@"error");
        } else {
            NSLog(@"Success");
        }
    }];
}

[library release];
```

TN5. Notification handling

It’s possible to get notifications from VKPlayerViewController & VKPlayerController or VKPlayerControllerTV about state changes, error handling & recording status. (This is optional, if you do not need to handle state change events or error handling or recording status, then **skip this step**)

```
NSString *urlString = @"http://xxx.xxx.xxx.xxx";
VKPlayerViewController *playerVc = [[[VKPlayerViewController alloc] initWithURLString:urlString
decoderOptions:NULL] autorelease];
playerVc.delegate = self;
```

Then implement below method to handle state change events and/or errors

```
- (void)onPlayerViewControllerStateChanged:(VKDecoderState)state errorCode:(VKError)errCode {
    if (state == kVKDecoderStateInitialized) {
    } else if (state == kVKDecoderStateConnecting) {
    } else if (state == kVKDecoderStateConnected) {
    } else if (state == kVKDecoderStateInitialLoading) {
    } else if (state == kVKDecoderStateReadyToPlay) {
    } else if (state == kVKDecoderStateBuffering) {
    } else if (state == kVKDecoderStatePlaying) {
    } else if (state == kVKDecoderStatePaused) {
    } else if (state == kVKDecoderStateStoppedByUser) {
    } else if (state == kVKDecoderStateConnectionFailed) {
    } else if (state == kVKDecoderStateStoppedWithError) {
        if (errCode == kVKErrorStreamReadError) {
        }
    }
}
```

About recording callbacks,

There is no a different delegate property for recording, thus, when the VKPlayerController or VKPlayerControllerTV instance’s delegate is set, then if below optional callbacks are implemented, they will be called,

For whom uses VKPlayerController or VKPlayerControllerTV,

```
- (void)player:(VKPlayerControllerBase *)player didStartRecordingWithPath:(NSString *) recordPath {
    NSLog(@"Recording started with path = %@", recordPath);
}

- (void)player:(VKPlayerControllerBase *)player didStopRecordingWithPath:(NSString *)recordPath
error:(VKErrorRecorder)error {
        if (error == kVKErrorNone) {
            NSLog(@"Recording is ended with success");
        } else {
            NSLog(@"Recording is ended with error = %lu", error);
        }
}
```

For whom uses VKPlayerViewController,

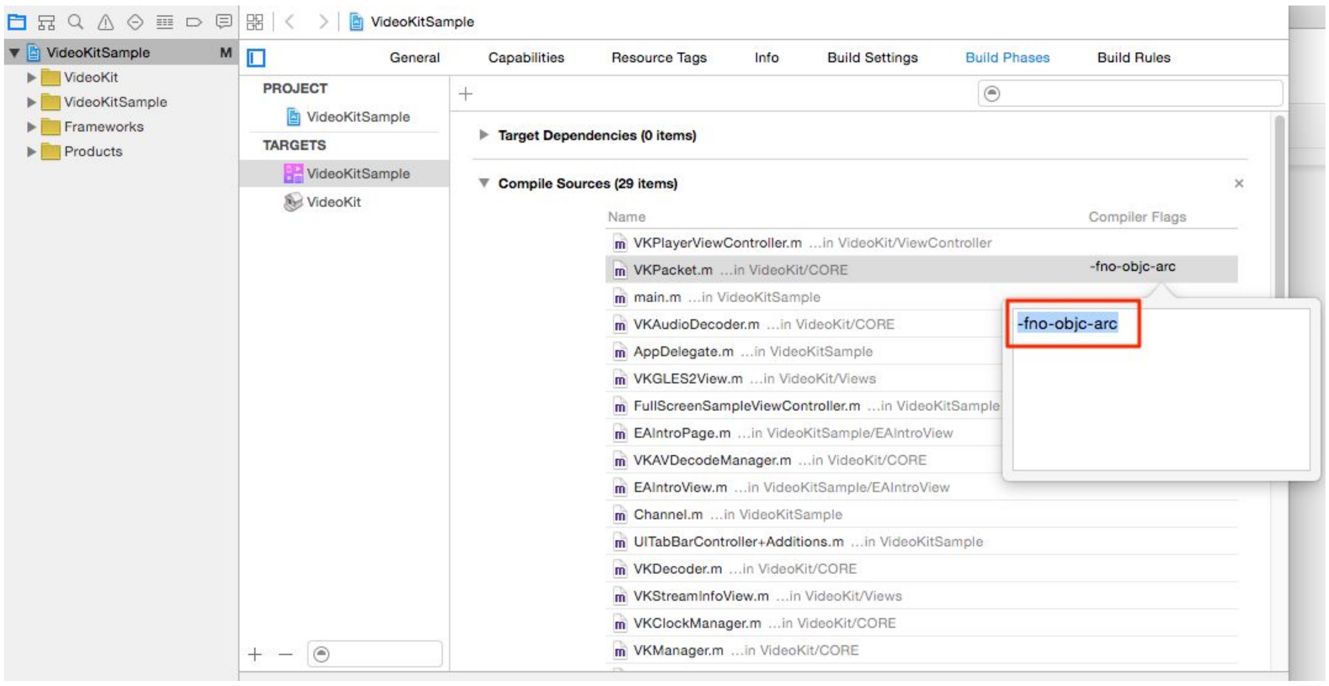
```
- (void)onPlayerViewControllerDidStartRecordingWithPath:(NSString *)recordPath {
    NSLog(@"Recording started with path = %@", recordPath);
}

- (void)onPlayerViewControllerDidStopRecordingWithPath:(NSString *)recordPath
error:(VKErrorRecorder)error {
    if (error == kVKErrorNone) {
        NSLog(@"Recording is ended with success");
    } else {
        NSLog(@"Recording is ended with error = %lu", error);
    }
}
```

TN6. ARC Supported Projects

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

Go to Targets -> Build Phases -> Compile Sources

double click on the right column of the row under Compiler Flags (It can also be added to multiple files by holding the cmd button to select the files and then pressing enter to bring up the flag edit box.)

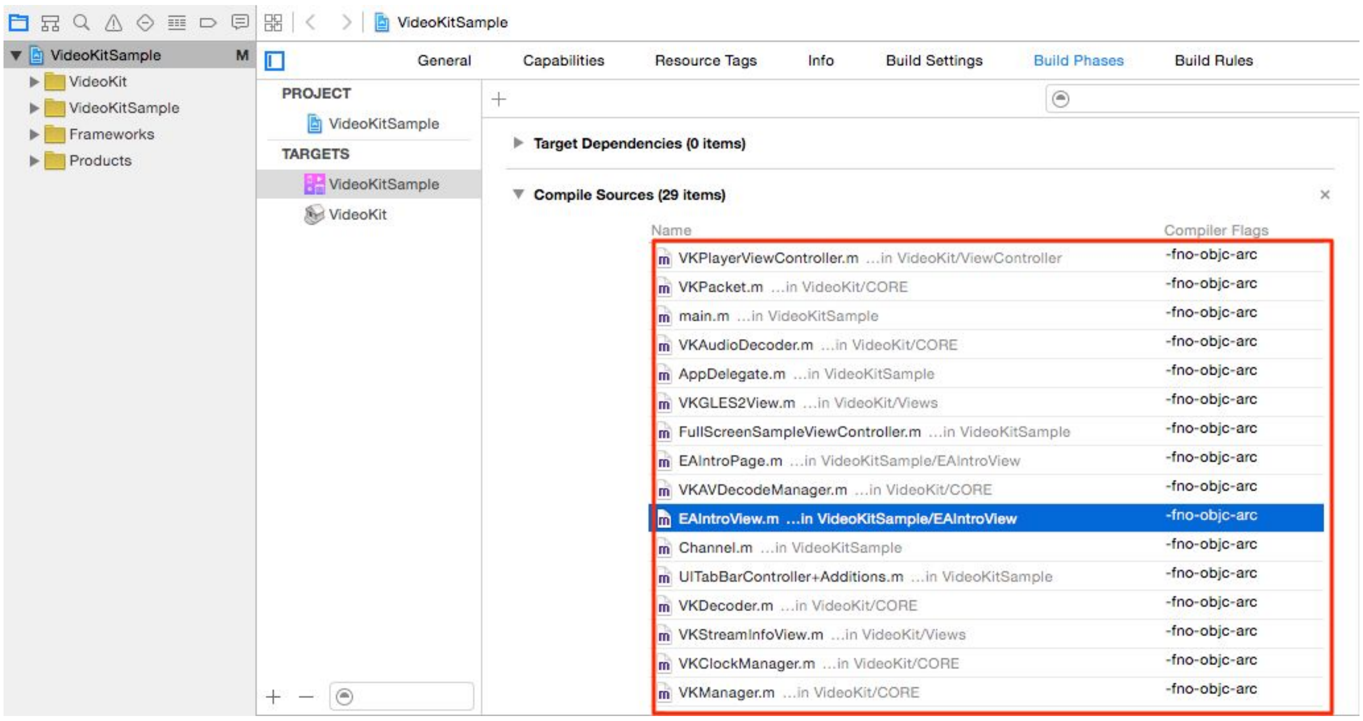


Name	Compiler Flags
VKPlayerViewController.m ...in VideoKit/ViewController	
VKPacket.m ...in VideoKit/CORE	-fno-objc-arc
main.m ...in VideoKitSample	
VKAudioDecoder.m ...in VideoKit/CORE	
AppDelegate.m ...in VideoKitSample	
VKGLES2View.m ...in VideoKit/Views	
FullScreenSampleViewController.m ...in VideoKitSample	
EAIntroPage.m ...in VideoKitSample/EAIntroView	
VKAVDecodeManager.m ...in VideoKit/CORE	
EAIntroView.m ...in VideoKitSample/EAIntroView	
Channel.m ...in VideoKitSample	
UITabBarController+Additions.m ...in VideoKitSample	
VKDecoder.m ...in VideoKit/CORE	
VKStreamInfoView.m ...in VideoKit/Views	
VKClockManager.m ...in VideoKit/CORE	
VKManager.m ...in VideoKit/CORE	

-fno-objc-arc

-fno-objc-arc

Write above compiler flag into the box



Name	Compiler Flags
VKPlayerViewController.m ...in VideoKit/ViewController	-fno-objc-arc
VKPacket.m ...in VideoKit/CORE	-fno-objc-arc
main.m ...in VideoKitSample	-fno-objc-arc
VKAudioDecoder.m ...in VideoKit/CORE	-fno-objc-arc
AppDelegate.m ...in VideoKitSample	-fno-objc-arc
VKGLES2View.m ...in VideoKit/Views	-fno-objc-arc
FullScreenSampleViewController.m ...in VideoKitSample	-fno-objc-arc
EAIntroPage.m ...in VideoKitSample/EAIntroView	-fno-objc-arc
VKAVDecodeManager.m ...in VideoKit/CORE	-fno-objc-arc
EAIntroView.m ...in VideoKitSample/EAIntroView	-fno-objc-arc
Channel.m ...in VideoKitSample	-fno-objc-arc
UITabBarController+Additions.m ...in VideoKitSample	-fno-objc-arc
VKDecoder.m ...in VideoKit/CORE	-fno-objc-arc
VKStreamInfoView.m ...in VideoKit/Views	-fno-objc-arc
VKClockManager.m ...in VideoKit/CORE	-fno-objc-arc
VKManager.m ...in VideoKit/CORE	-fno-objc-arc

Do this for all VideoKit source files.

TN7. How to use Airplay feature

AirPlay is the term Apple uses for the protocol that lets you stream your Apple devices to your Apple TV
But, Apple has, bafflingly, not given developers any way to add a Dual Screen AirPlay button inside their apps – users must (themselves) open the multitasking tray(<= iOS 6)/control center(iOS 7 & above), swipe to/find the AirPlay icon, select the Apple TV and turn on Mirroring.
(more info at apple site - iOS: How to use AirPlay Mirroring - http://support.apple.com/kb/HT5209)

After enabling mirroring, when the player starts playing, dual screen is automatically detected and video plays on Apple TV screen If allowAirPlay property of VKPlayerController/VKPlayerViewController class is set to YES. (default value is NO)

For tvOS version of VideoKit, there is no need to support Airplay as video already shows on television screen.

TN8. Hardware Acceleration Notes

A. Audio

Since beginning VideoKit 2.6, below audio codecs are supported with hardware acceleration using Apple native framework named AudioToolbox.framework

1. aac
2. ima
3. alac
4. amr
5. gsm
6. ilbc
7. mp1
8. mp3
9. **ac3 ***
10. **eac3 ***

ac3 and eac3 codecs are licensed by Dolby Digital and can not be used without a license. But, with the release of iOS 9.3, Apple has given support to Dolby Digital Plus (aka Enhanced AC-3) and Dolby Digital (AC-3) codecs and these codecs are natively supported data types in Core Audio, the foundational audio framework for all Apple platforms. However, **not all iOS devices** have hardware chipset to decode these codecs. The supported device list is below,
(Newer devices from the list also supports ac3 & eac3 audio hardware decoding)

Platform	Dolby Digital(ac3)	Dolby Digital Plus (eac3)
iPhone 5c and older	✗	✗
iPhone 5s and newer	✓	✓
iPhone 6 Series	✓	✓
iPhone SE	✓	✓
Original iPad Series	✗	✗
iPad mini 2/3/4	✓	✓
iPad Air Series	✓	✓
iPad Pro Series	✓	✓
iPod touch 6th Gen	✓	✓

B. Video

H264 codec

Hardware video acceleration for h264 codec is used with the help of Apple native framework named VideoToolBox.framework. This framework is available for devices that have iOS 8 & tvOS 10.2 and above. Although there seems all iOS devices support hardware video acceleration, there are some limitations for h264 profiles according to the device. As expected, older devices like iPhone 4s, iPad mini, etc can not decode video streams that have high resolution or high bitrate and newer devices have more powerful hardware decoder chipsets and can play very high resolution video streams. For tvOS, appleTV 4rd gen or above supports hardware h264 decoding but 4th gen may have limitations for high profile h264 videos.

H265 codec

Apple, in latest OS versions, let developers use VideoToolbox.framework to decode h265 videos for specific devices and OS versions (iOS11 or above). Basically, devices that have A9 or A10 Fusion chip or above can decode 8bit & 10bit h265 encoded videos, however since VideoKit doesn't support 10bit videos for now, We can say VideoKit supports only 8bit h265 encoded videos on those devices.

Since, there are different capabilities related to codecs like profiles for example, it's better to download the VideoKit sample project and try it on a device and see the result.In Xcode console, when the video starts playing, VideoKit drops a log message if the HW decoding is supported.

TN9. Handling M3U & PLS Playlist Files

VideoKit does not include a built in functionality that parses the playlist files like m3u and/or pls. But, We decided to make a basic demo to show how can be achieved even If VideoKit does not include this feature.

```
        let playlistManager = PlaylistManager(urlString: urlString)

        playlistManager.fetch(completionHandler: { channels in
            self.channelList = channels
            self.tableView.reloadData()
        }
```

With providing a m3u/pls playlist file url to the PlaylistManager object, it’s possible to get streaming urls in an array as channel objects.

This is the first part, in second part, the end of file notification must be handled to play the next channel as below,

```
    // MARK: - VKPlayerControllerDelegate
extension PlaylistSampleViewController:VKPlayerControllerDelegate {

    func player(_ player: VKPlayerControllerBase!, didChange state: VKDecoderState, errorCode errCode: 
VKError) {
        if (state == kVKDecoderStateConnecting) {
        } else if (state == kVKDecoderStateConnected) {
        } else if (state == kVKDecoderStateInitialLoading) {
        } else if (state == kVKDecoderStateReadyToPlay) {
        } else if (state == kVKDecoderStateBuffering) {
        } else if (state == kVKDecoderStatePlaying) {
            if (errCode == kVKErrorNoneReachedEndOfStream) {

                if (selectedIndex + 1) < channelList.count {
                    selectedIndex += 1
                    tableView.selectRow(at: IndexPath(row: selectedIndex, section: 0) , animated: true, scrollPositi
.top)

                    let channel = self.channelList[selectedIndex]

                    player?.contentURLString = channel.urlAddress
                    player?.decoderOptions = channel.options
                    player?.play()
                } else {
                    player?.stop()
                    tableView.deselectRow(at: IndexPath(row: selectedIndex, section: 0), animated: true)
                }

            } else {
            }
        } else if (state == kVKDecoderStatePaused) {
        } else if (state == kVKDecoderStateStoppedByUser) {
        } else if (state == kVKDecoderStateConnectionFailed) {
        } else if (state == kVKDecoderStateStoppedWithError) {
            if (errCode == kVKErrorStreamReadError) {
            }
        }
    }
}
```

By this way, as long as a new channel exists in the list, the new channel will start playing, and if there is no channel left, then the player will stop.

CHANGELOG

Version: 2.9 – Release Date: 01/03/2021

- Hardware Video acceleration for h265 codec
- Hardware Video acceleration for h264 codec for tvOS platform
- iOS14 & tvOS14 support
- openSSL is updated to stable 1.1.x version (openssl-1.1.1d)
- ffmpeg libraries are updated to 4.2.1 stable version
- VideoKit deployment target is set to 10.x
- Code is cleaned from old version SDK checks
- Sample project supports Swift 5

//Bug fixes & improvements
- kitkat.flv video has an A/V sync issue with ffmpeg 4.2.1 - FIXED
- filesharing was broken - FIXED
- channels are updated and a new webcam is added to the list.
- Added dark mode support to sample project
- Added logic to detect availability of video hardware acceleration for h264 and h265 codecs
- Fixed the bar height issue for devices having notch
- When recording MJPEG, the FPS(or PTS values) is not correct, so it causes fast playing - FIXED
- Hardware video accelerated decoding is enabled for tvOS
- iPhoneX iOS12 and iOS13, airplay fails with black screen, since renderView is zero size - FIXED
- showPicOnInitialBuffering feature is removed

Version: 2.8 – Release Date: 11/21/2017

- New shiny sample project written with Swift
- iOS11 support
- Sample project supports Swift 4
- Added new sample for playlist handling
- New shoutcast/icy metadata handling feature is added
- iPhoneX user interface support
- If video has multiple video streams, then now It's possible to switch

between video streams in realtime

//Bug fixes & improvements
• iOS11 autolayout crash - FIXED
• iOS11 warnings - FIXED
• Small ffmpeg memory leaks are FIXED
• Seeking in ts files is broken - FIXED
• When step method is called, a short stream's audio is heard - FIXED
• UIAlertView is depreciated - FIXED
• Airplay rotation issue on iOS11 - FIXED

Version: 2.7 – Release Date: 04/25/2017

• Custom I/O support (support playing audio/video stream in memory)
• BOB interlace opengles filter for deinterlacing interlaced videos for TV.
• Snapshots support in original video frame size.
• ffmpeg build script is updated for iOS10 SDK.
• gas-processor (improves video decoding performance) is updated to the latest version

//Bug fixes & improvements
• If probesize is not enough to determine the colorspace, then default value causes the crash - FIXED
• End of file detection feature is broken - FIXED
• Recording can be restart rapidly (It's useful to change recording file for file size issues.)
• Although the license check mechanism is disabled in Enterprise license, license check warning is logged - FIXED
• App crashes when trying to update screen with opengles while App is in background - FIXED
• AudioToolbox can not decode MP2, pcm_alaw and pcm_ulaw, so they are disabled in audio hw decoder check
• disableInnerBuffer property is not effective since the ffmpeg flag is passed to avformat after connection - FIXED
• Two new decoder options are added to disable hardware acceleration for specific audio and video codecs
• New player property to manage mic recording for users using broadcasting sdk's in their projects
• After player is destroyed, the CVtextureCache objects are not removed

and caused to leak - FIXED
• timerPanelHidden is retained but never released and caused to memory leak - FIXED
• NSTimer type timerPanelHidden has memory leak - FIXED

Version: 2.6 – Release Date: 09/29/2016

• Hardware Video acceleration for h264 codec (iOS platform only)
• Hardware Audio acceleration for ac3, aac, eac3, amr, and mp3 codecs
• Portrait videos are supported
• iOS10 & Xcode 8 support
• cache protocol support (youtube like caching for http file streams)
• ffmpeg libraries are updated to latest 3.1.3 stable version
• Build scripts are updated for Xcode 8 and iOS10 (bitcode support)
• VideoKit deployment target is set to 8.x
• Code is cleaned from iOS 7 SDK checks
• Some UI related bugs are fixed

Version: 2.5 – Release Date: 06/18/2016

• Added zoom to anywhere functionality with pinch in/out gesture on video as in whatsapp
• FTP protocol support is added
• Both embed & fullscreen iOS player views' are adapted to autolayout
• Both embed & fullscreen tvOS player views' are adapted to autolayout
• New fancy transition animation from embed to fullscreen and vs
• Depreciated methods & classes are removed and new APIs are used (NSURLConnection, NSString, etc ...)
• Added panning functionality for mono audio streams
• ffmpeg logs can be disable
• openSSL is updated to a newer version (1.0.1i -> 1.0.2f)

//Bug fixes
• Can not get snapshots of multiple players playing at the same time - FIXED
• Parsing stream url address fails when passthrough is set & there is no any avoption for the url - FIXED
• RGB - YUV color space conversion causes frames seen faded - FIXED
• Some streaming servers start sending A/V packets with big time gaps that causes slowdown on displaying frames for a seconds - FIXED

• Recording fails for some live streams because of order error during muxing - FIXED
• VKManager.a & VideoKitCore.a are not bitcode enabled - FIXED

Version: 2.4 – Release Date: 11/30/2015

• Added tvOS support to VideoKit, applying new concepts for tvOS such as Focus Management and controlling all features from the remote controller

- play/pause video,
- start/stop recording,
- zoom in/out,
- mute/unmute,
- show/hide information view
- enter/exit fullscreen mode in embedded view.

• Added 'aac_adtstoasc' bitstream filter support to be able to record supported codecs by iOS into mp4 container
• Added AUGraph support
• Added Audio panning feature to route audio left or right channel
• rtmpt & rtsps protocols are added to supported protocol list
• ffmpeg libraries are updated to latest 2.8.1 stable version
• Build scripts are updated for Xcode 7 and iOS9 (bitcode support)
• Updated gas-preprocessor.pl file
• VideoKit deployment target is set to 7.x
• Code is cleaned from iOS 6 SDK checks
• Suppress all warnings in cmdutils.c file
• Streams that has more than 2 audio channels are now downgraded to 2 channels
• Player crashes when double tap is received after embed to full screen transition - FIXED
• Non-monotonous DTS in output stream error got during recording - FIXED
• If one player is stopped when multiple players are playing, all playing players are muted - FIXED
• player view is rotated left and back to normal again in iOS 7 when player is transiting from embed to fullscreen - FIXED
• Crash on stopping player because of the slider - FIXED
• Audio Video synchronization is lost when network is not reachable for a time - FIXED

Version: 2.3 – Release Date: 05/06/2015

• Added disableInnerBuffer setting to have more control on the management of buffers (useful feature for fixing latency issues)
• If the license-form.plist file is not reachable in a way in the project, then username and secret properties of player classes can be set to fill license-form as well.
• The frame rate of video stream can be decreased - A useful feature to play multiple local videos more than 4
• Playable streams are now including language information
• Commenting out TRIAL definition does not remove developer you are using .. warning message in single binary version - FIXED
• If multiple players are working in the same time, play/stop actions cause a crash - FIXED
• If there needs extra space to decode packet, then av_decode_video2 crashes - FIXED
• If a network stream is stopped in early states of playing, and then If the play action immediately fired, the action is not taken - FIXED
• If audioDecoder is not ready then changing volume is not effective - FIXED
• FIX_FOR_RTSP_TEARDOWN_MESSAGE define check is removed from core because of lack of ability to activate for binary license holders
• Status bar is covering the toolbar if it's not hidden - FIXED
• Done button on toolbar looks very tiny - FIXED
• Volume controls (Mute & volumeLevel) can not be set before decodemanager is initialized - FIXED
• Buffering event is not fired the exact time that the buffering begins - FIXED
• When the video stream is paused, the zoom action is not taken till replay - FIXED

Version: 2.2 – Release Date: 12/04/2014

• If the stream includes data-type stream, VideoKit crashes during recording - FIXED
• If the stream includes multiple audio streams, VideoKit fails on recording this stream - FIXED
• FFmpeg does not send RTSP-TEARDOWN message to server when the connection is closed - FIXED
• After the seeking slider is set to a position, it jumps back to old position for a short moment - FIXED
• Quotes and '$' character in arguments are not handled when VKDECODER_OPT_KEY_PASS_THROUGH is used - FIXED
• Embedded player's title can not be set with a custom text - FIXED
• Multiple Players can not be started at the same time - FIXED
• SetStatusBarHidden for Container of VKPlayerController is not updated for iOS 7 & above - FIXED
• If player view is added to a subview not directly to viewcontroller's view, then grow/shrink animation doesn't work well - FIXED
• License checking code has a bug that logs a message even though the credentials are valid - FIXED
• VideoKit Sample project supports iPhone 6 & iPhone 6 plus native resolutions now.
• Connection errors now give more details about the problem
• Added missed methods declerations in VKPlayerController class

Version: 2.1 – Release Date: 10/03/2014

• Recording functionality is added (No transcoding, record as it is)
• Player UI is updated for iOS 7 and above (iOS 6 UI is already supported)
• ffmpeg & openssl scripts are updated for XCode 6.0 cmdline tools
• Code is cleaned from iOS 5 SDK checks
• VideoKit deployment target is set to 6.x
• ffmpeg libraries are updated to latest 2.4.2 stable version

• Some APIs are depreciated, those are replaced with new ones in 2.4.2

Version: 2.0 – Release Date: 08/23/2014

• Added support for secure streams with openSSL (now https, rtmps, rtmpe, rtmpte, rtmpte & tls protocols are supported)
• Added new feature: VideoKit shows the first picture immediately without waiting to fetch all frames to start
• TCP & UDP protocols are supported (tcp://xxx & udp://xxx streams can be playable)
• Seek in RTMP file streams can be possible
• ffmpeg libraries are updated to latest 2.x stable version
• Build script is updated for building & supporting openSSL & ffmpeg version 2.x
• Some APIs are depreciated, those are replaced with new ones in 2.x
• When using multiple players, the owner of the post notification can not be determined in delegate method – FIXED
• 1 KB is not 1024 Bytes in file system – FIXED
• Some log types are changed in play and stop methods
• In some local files, restarting from the beginning of the movie causes no sound, FIXED
• In some videos, frame timer is updated so late, this causes black screen on the start of video, FIXED
• NEON support is broken in Xcode 5.x, new scripts are added for Xcode 4.6.3 to build with neon & asm support
• Remote license activating & controlling mechanism

Version: 1.11 – Release Date: 02/20/2014

• VideoKit crashes when try reaching not thread safe ffmpeg methods by multiple players at the same time, FIXED

Version: 1.10 – Release Date: 01/31/2014

• RTP protocol is supported
• Color formats other than YUV are supported (like RGB, BGR ...)
• Animated GIF files are supported with full transparency
• New decoder parameter, which is VKDECODER_OPT_KEY_PASS_THROUGH, is added to support passing different server parameters to ffmpeg.
• ffmpeg build script is updated for supporting arm64 arch

Version: 1.02 – Release Date: 01/05/2014

• Removed unnecessary header files

Version: 1.01 – Release Date: 12/10/2013

• VideoKit crashes on iOS 5.x devices on runtime, FIXED
• Removed Right Swipe Gesture recognizer which is unused
• Detecting file stream algorithm is broken, FIXED

Version: 1.0 – Release Date: 12/01/2013

• First initial release