

IOS 播放器 sdk

概述

AlivcMediaPlayer 是一款基于 IOS 平台的多媒体视频播放 SDK。它为 IOS 的开发者提供了简单易用的接口，帮助开发者方便快捷、低门槛的实现多媒体播放功能的开发。它支持 HLS、RTMP、HTTP FLV、MP4 等多种流媒体播放格式，视频支持 h264 格式、音频支持 AAC 格式。另外，针对直播用户的需求，还增加了首帧秒开的功能；同时为了减少直播的延迟，增加了弱网条件下播放的跳帧功能。

版本和新增功能

功能	版本
支持 HLS、RTMP、HTTP FLV、mp4 等流格式	v1.0
支持 h264+aac	v1.0
支持 armv7、arm64	v2.0
支持直播首帧秒开	v2.1
支持弱网条件下的丢帧策略	v2.1
支持多实例、支持模拟器调试，支持 https	v2.2
支持带切边的视频渲染模式	v2.2

阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员以及对此感兴趣的用户，要求开发者对播放器的基本功能有一定的了解。

开发准备

设备和系统版本

ios8.0 及以上

iphone5 及以上

安装包下载及说明

安装包的下载地址为：[点击下载](#)

播放器 SDK 的完整下载包中包含 demo、doc、lib 等：

1. demo：主要存放了调用 SDK 的示例工程，可以帮助用户了解如何使用该 SDK。
2. lib：播放器 SDK 开发包，包含播放器 framework 文件，需要在您的工程中进行引用。其中，arm 目录下面的 framework 仅支持 armv7 和 arm64 平台，arm&simulator 目录下面的 framework 除了 armv7、arm64 外，还支持 x86、x86_64，可以用作模拟器调试。

快速开发

开发环境配置

1. 需要准备 iOS 的运行环境（XCode6.0 以上版本，iOS SDK8.0 以上版本），以及硬件 CPU 支持 ARMv7、ARMv7s 或 ARM64 的 iOS 设备。
2. 在阿里云官网上注册云帐号，并开通视频点播或视频直播服务。方法如下：

[视频点播服务开通](#)

[视频直播服务开通](#)

3. 通过访问控制服务创建播放器专用子帐号及其 AccessKey：

- a. 登陆[访问控制服务控制台](#)
- b. 在用户管理中新建用户：

访问控制RAM

概览

用户管理

群组管理

授权策略管理

角色管理

设置

用户管理

新建用户

刷新

登录名

请输入登录名进行模糊查询

搜索

登录名/显示名	备注	创建时间	操作
		22:09:00	管理 授权 删除 加入组
		21:34:19	管理 授权 删除 加入组

共有2条， 每页显示：20条

«

«

1

»

»

注意勾选为该用户自动生成 AccessKey 选项：

* 登录名:

player

长度1-64个字符，允许输入小写英文字母、数字、"@",".","_"或"-"

显示名:

长度1-12个字符或汉字，允许输入英文字母、数字、"@",".","_"或"-"

备注:

邮箱:

国家/地区:

中国大陆(+86)



电话:

☒ 为该用户自动生成AccessKey

确定

取消

创建子帐号成功，注意保存好该帐号的 AccessKey：

这是用户AccessKey可供下载的唯一机会，请及时保存！

✓

新建AccessKey成功！

AccessKey详情

▼

保存AK信息

- c. 为子帐号分配调用播放器权限：

点击授权链接：

用户管理

新建用户

刷新

登录名

请输入登录名进行模糊查询

搜索

登录名/显示名	备注	创建时间	操作	
player		2016-05-30 13:44:24	管理	授权 删除 加入组

在可选授权策略名称中搜索 mts，将 AliyunMTSPlayerAuth 授予此子帐号：

添加授权策略后，该账户即具有该条策略的权限，同一条授权策略不能被重复添加。

可选授权策略名称	类型	已选授权策略名称	类型
mts		AliyunMTSPlayerAuth	系统
AliyunMTSFullAccess	系统	使用媒体转码服务(MTS)播放器的权...	
管理媒体转码服务(MTS)的权限			

用户权限验证

用户需要用申请得到的 AccessKeyID 和 AccessKeySecret 进行权限验证，才能够使用播放器。权限验证需要通过实现 AliVcAccessKeyProtocol 协议来完成。

```
1. [AliVcMediaPlayer setAccessKeyDelegate:self];
```

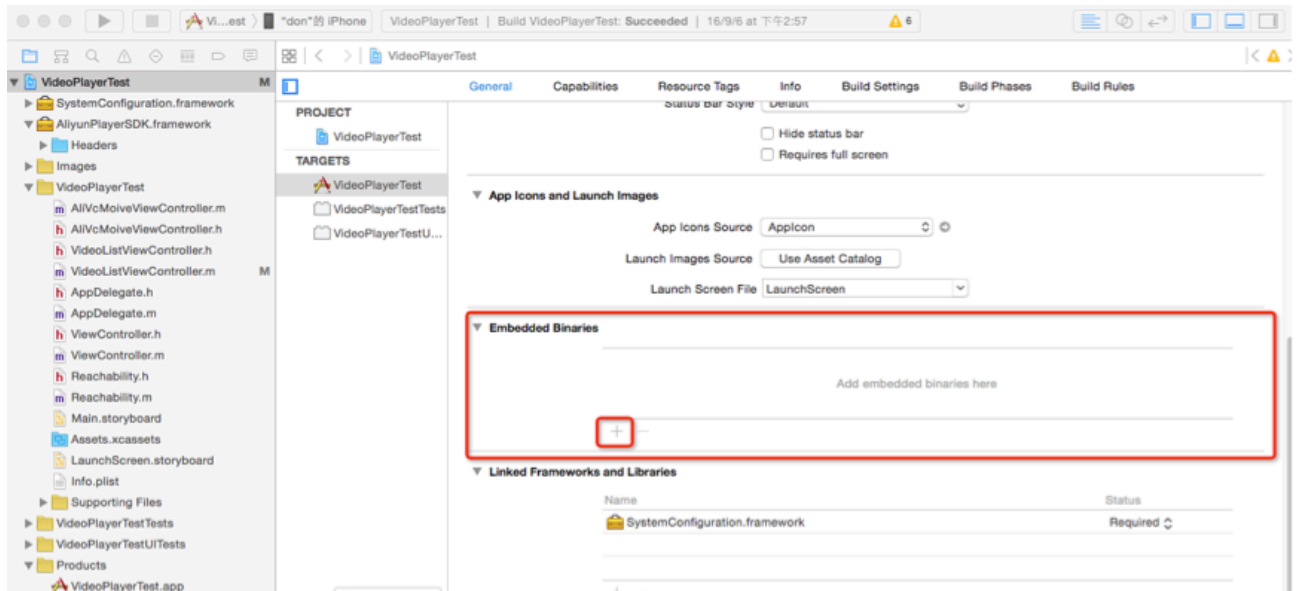
权限验证可以在播放器创建之前进行，且播放多个视频只需要验证一次即可。通过协议的 getAccessKeyIDSecret 接口，sdk 可以获取用户的 AccessKeyID 和 AccessKeySecret 来完成验证。

```
1. NSString* accessKeyID = @"QxJIheGFRL926hFX";
2. NSString* accessKeySecret = @"hipHJKpt0TdznQG2J4D0EVSavRH7mR";
3. -(AliVcAccesskey*)getAccessKeyIDSecret
4. {
5.     AliVcAccesskey* accessKey = [[AliVcAccesskey alloc] init];
6.     accessKey.accessKeyId = accessKeyID;
7.     accessKey.accessKeySecret = accessKeySecret;
8.     return accessKey;
9. }
```

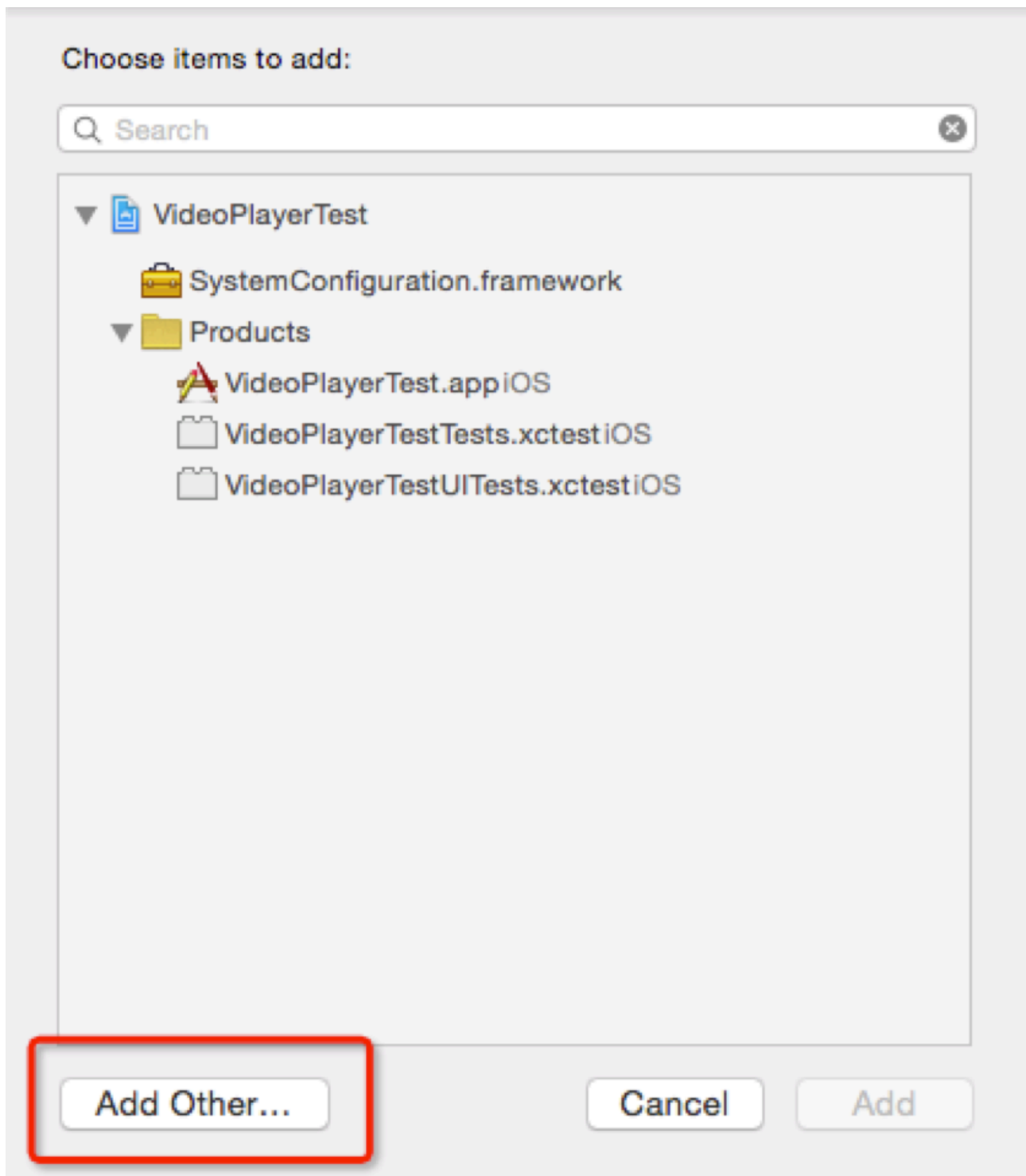
sdk 包添加

使用 xcode 创建工程的方法如下：

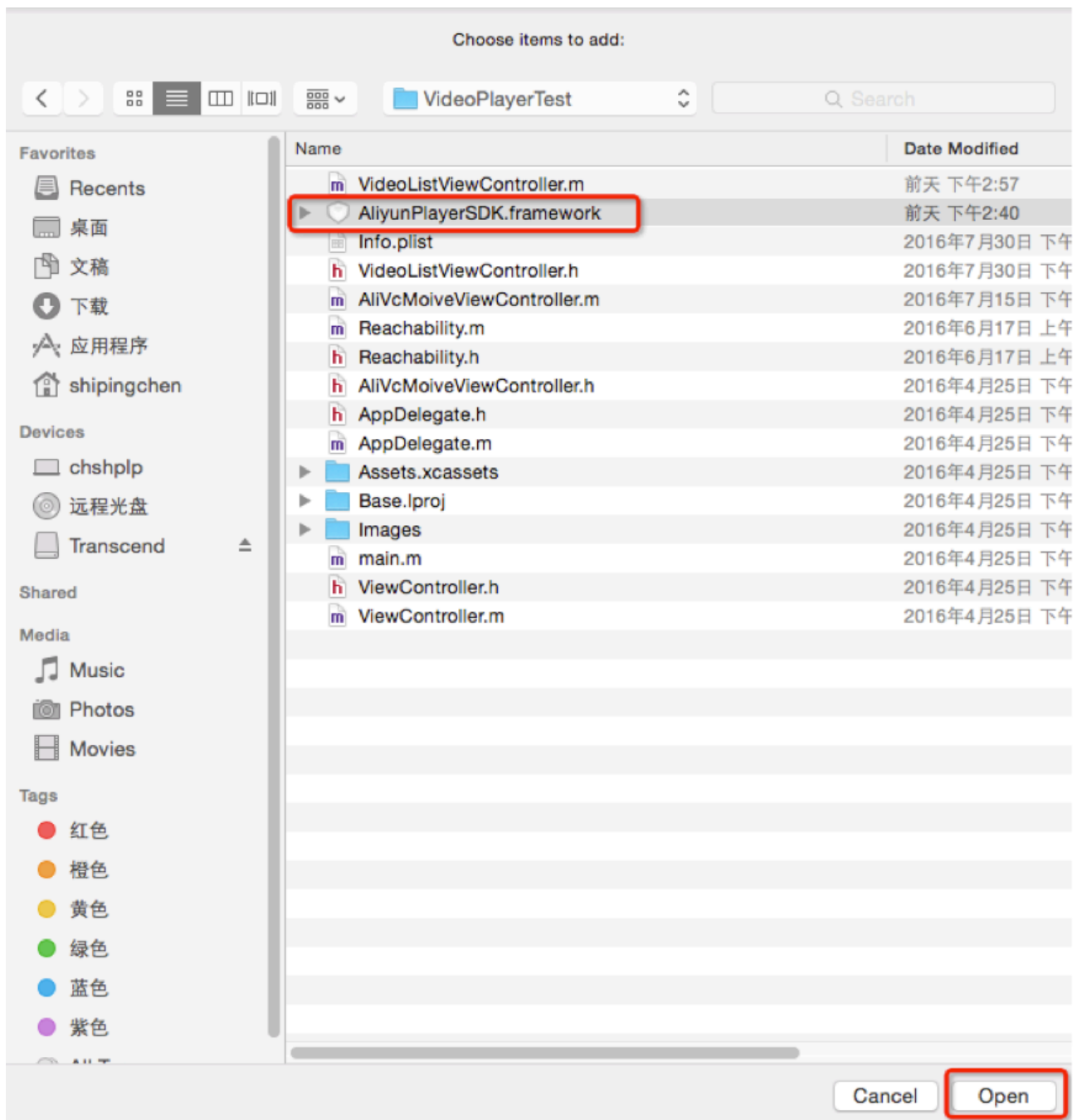
1. XCode 创建一个 iOS 应用工程。
2. 将 SDK 中的 framework 添加到工程中，如图：



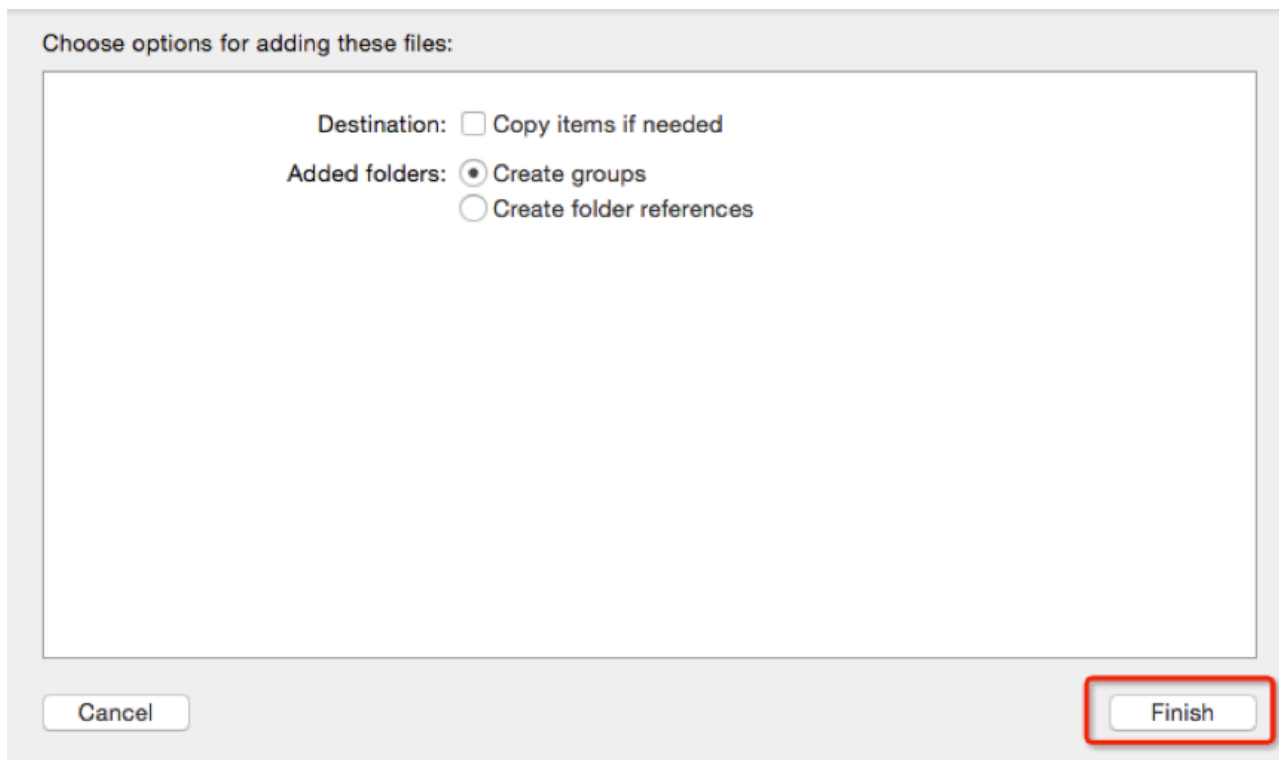
点击 “+” 号后出现：



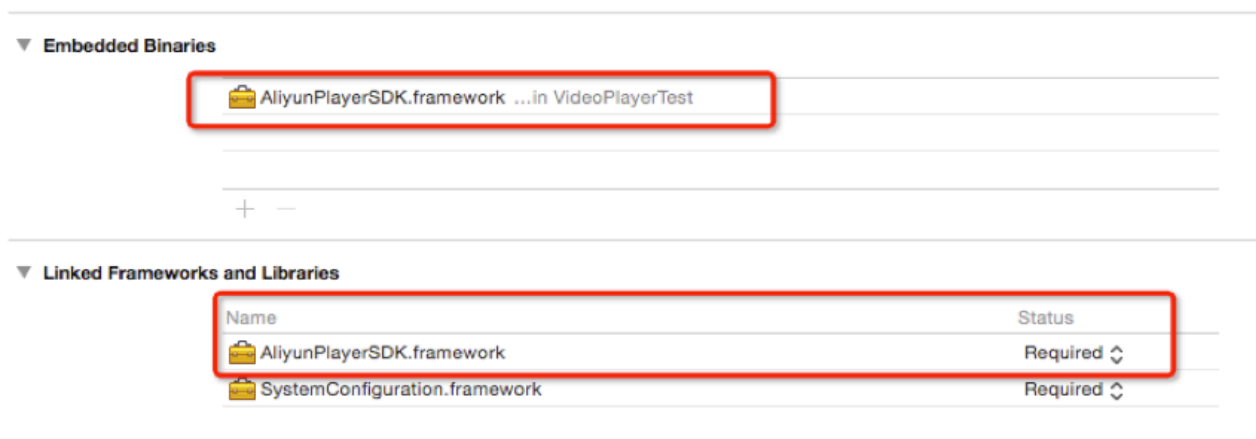
点击 “Add Other...” ，选中 framework.



点击 “Open” .



点击 “Finish”



完成 framework 的添加

1. 将 SDK 中的头文件 AliyunPlayerSDK.h 包含到工程中。

demo 示例

在 SDK 中提供了 Demo，此 Demo 是用播放器 SDK 开发了一个完整的视频播放器，用户可以参考 Demo 进行播放器的开发。

首先，我们来看一下组成播放器的基本模块以及播放器的工作流程，见下图：

使用 SDK 开发播放器时，基本的开发步骤为：

1. alloc 播放器后，调用 create 创建播放器，并传入 view 显示窗口
2. 注册通知响应函数。
3. 调用 prepareToPlay 准备开始播放，传入要播放的视频地址。
4. 调用 play 接口进行播放。

下面以 demo 为例来详细说明上面的这些步骤。

```
1.      //将需要播放的视频添加到视频列表中
2.      //如果需要播放本地视频，则将本地视频拷贝到应用程序的 Document 目录下即可
3.      -(void) addVideoToList
4.      {
5.          //按照如下格式进行添加， videoName 是在列表中显示的名字
6.          [videolists setObject:@"http://yourVideoAddress.m3u8" forKey:@"videoName"];
7.      }

1.
2.      -(void) playVideo
3.      {
4.          //新建播放器
5.          player = [[AliVcMediaPlayer alloc] init];
6.          //创建播放器，传入显示窗口
7.          [player create:mShowView];
8.          //注册准备完成通知
9.          [[NSNotificationCenter defaultCenter] addObserver:self
10.             selector:@selector(OnVideoPrepared:)
11.             name:AliVcMediaPlayerLoadDidPreparedNotification object:player];
12.          //注册错误通知
13.          [[NSNotificationCenter defaultCenter] addObserver:self
14.             selector:@selector(OnVideoError:) name:AliVcMediaPlayerPlaybackErrorNotification
15.             object:player];
16.          //传入播放地址，初始化视频，准备播放
17.          [player prepareToPlay:mUrl];
18.          //开始播放
19.          [player play];
20.      }
```

播放器可配置参数与可选功能

配置参数	用途描述
timeout	设置网络超时断开链接的时间
dropBufferDuration	设置直播过程中缓冲区视频丢帧的起始时间，若缓冲区中视频帧的时长超过这个值，则开始丢帧操作。设置这个参数可以控制直播延时的长度，参数值越小则直播的延迟越小。

seek 功能

接口名称	用途描述
seekTo	seek 到指定位置之前的最近的一个关键帧
seekToAccurate	精准跳转到指定位置

除了上述可配置的功能和参数，AlivcMediaPlayer 还定义了播放器的事件状态通知和错误代码，以方便开发者掌握播放器的运行状态。

```
1. -(void) OnVideoPrepared:(NSNotification *)notification
2. {
3.     //收到完成通知后，获取视频的相关信息，更新界面相关信息
4.     [self.playSlider setMinimumValue:0];
5.     [self.playSlider setMaximumValue:player.duration];
6. }
7.
8. -(void)OnVideoError:(NSNotification *)notification
9. {
10.     AliVcMovieErrorCode error_code = player.errorcode;
11. }
```

若需要了解上述功能和接口的详细用法，请参照下节的接口说明。

接口说明

SDK 中提供了类 AliVcMediaPlayer 来实现播放器各种功能接口，同时，我们还提供了播放器的各种状态通知，以及播放器的各种错误代码。

接口名称	功能描述
create	创建播放器
prepareToPlay	初始化视频，准备播放
play	开始播放视频
pause	暂停视频播放
stop	停止视频播放
reset	重置播放器
destroy	销毁播放器
seekTo	跳转到指定位置
seekToAccurate	精准跳转到指定位置
view	设置播放显示窗口
videoHeight	获取视频高度
videoWidth	获取视频宽度
livePlayer	设置播放器的直播点播模式
timeout	设置网络超时时间
dropBufferDuration	缓冲区丢帧的起始长度
duration	获取视频长度
currentPosition	获取当前视频播放位置
bufferingPosition	获取当前视频缓冲位置
muteMode	播放器静音模式
scalingMode	播放器渲染时的缩放模式
errorCode	播放器错误代码
getSDKVersion	获取播放器版本号

接口名称	功能描述	
setUserID	设置用户 ID	
setBussinessID	设置业务 ID	
getPropertyDouble	获取 double 型性能参数	
getPropertyLong	获取 Long 型性能参数	
getPropertyString	获取 String 型性能参数	
状态事件通知名称		通知内容
AliVcMediaPlayerLoadDidPreparedNotification		播放准备完成通知
AliVcMediaPlayerPlaybackDidFinishNotification		播放结束通知
AliVcMediaPlayerStartCachingNotification		开始缓冲通知
AliVcMediaPlayerEndCachingNotification		结束缓冲通知
AliVcMediaPlayerPlaybackErrorNotification		播放错误通知
AliVcMediaPlayerSeekingDidFinishNotification		跳转结束通知
AliVcMediaPlayerFirstFrameNotification		首帧显示通知
错误代码 AliVcMovieErrorCode		错误说明
ALIVC_SUCCESS		无错误
ALIVC_ERR_ILLEGALSTATUS		非法的播放流程
ALIVC_ERR_NO_NETWORK		无网络下播放网络视频
ALIVC_ERR_FUNCTION_DENIED		权限验证失败
ALIVC_ERR_UNKOWN		未知错误
ALIVC_ERR_NO_INPUTFILE		无输入文件
ALIVC_ERR_NO_VIEW		没有设置显示窗口
ALIVC_ERR_INVALID_INPUTFILE		无效的输入文件
ALIVC_ERR_NO_SUPPORT_CODEC		视频压缩格式不支持

错误代码 AliVcMovieErrorCode	错误说明
ALIVC_ERR_NO_MEMORY	内存不足

下面详细介绍一下各个成员函数的具体使用：

create

```
- (AliVcMovieErrorCode) create: (UIView*)view
```

功能：创建播放器，并设置播放器显示窗口。播放器内部会新建各个播放器变量并初始化，并启动播放器内部流水线线程等。

参数：UIView* view，播放器显示窗口

备注：如果创建播放器的时候 view 没有，则可以传递 nil，可以在后续需要设置 view。

prepareToPlay

```
- (AliVcMovieErrorCode) prepareToPlay: (NSURL*)dataSource
```

功能：根据视频文件内容初始化播放器实例，包括读取视频头，解析视频和音频信息，并根据视频和音频信息初始化解码器，创建下载（或读取本地文件）、解码、渲染线程等。

参数：(NSURL*)dataSource，当前播放视频的文件名或 URL

返回值：若播放器初始化成功，返回 ALIVC_SUCCESS；否则返回失败。

prepareToPlay 的具体过程为：

1. 验证用户是否有权限调用该函数。
2. 验证参数 dataSource 是否为空。
3. 如果播放器是正在播放或者正在暂停状态，则不能够进行 prepare，此时返回非法的播放流程错误 ALIVC_ERR_ILLEGALSTATUS，如果播放器状态是已经是准备完成状态，则返回 ALIVC_SUCCESS。
4. 对视频进行初始化，如果成功，则会发送 LoadDidPreparedNotification 通知，表示视频初始化完成。如果失败则会发送 PlaybackErrorNotification 通知，在错误通知中可以获取到错误代码。

备注：该函数是异步函数，需要等待播放准备完成通知 AliVcMediaPlayerLoadDidPreparedNotification，收到该通知后代表视频初始化完成，同事还可以获取到视频的相关信息如：duration、videoWidth、videoHeight。

play

```
- (AliVcMovieErrorCode) play
```

功能：播放当前视频

返回值：当播放视频成功，返回 ALIVC_SUCCESS，否则返回失败。

play 的具体过程为：

1. 验证是否有权限调用该函数
2. 如果播放器是停止的状态，则直接返回 ALIVC_ERR_ILLEGALSTATUS 错误；如果此时播放器为播放的状态，直接返回 ALIVC_SUCCESS。
3. 如果播放器在暂停或者准备完成的状态，则直接启动视频播放。

备注：播放器调用 play 进行播放，必须在播放器状态为准备完成的状态或者暂停的状态才能进行播放，其他情况都不能够将视频播放起来。

pause

```
- (AliVcMovieErrorCode) pause
```

功能：暂停当前视频播放

返回值：暂停视频播放成功 ALIVC_SUCCESS。否则返回失败。

pause 的具体过程为：

1. 验证是否有权限调用该函数。
2. 如果此时播放器为暂停状态，直接返回 ALIVC_SUCCESS。
3. 如果播放器状态为停止或者准备完成状态则返回错误的播放器状态 ALIVC_ERR_ILLEGALSTATUS。
4. 其他情况则暂停视频播放，并将播放器状态设置为暂停状态。

备注：调用 pause 函数将暂停视频播放。一般在视频正在播放的情况下调用此函数。

stop

```
- (AliVcMovieErrorCode) stop
```

功能：停止当前视频播放，调用此函数则是结束视频播放，视频显示为黑屏，并回到视频播放起始点。

返回值：停止视频播放成功 ALIVC_SUCCESS。否则返回失败。

stop 的具体过程为：

1. 验证是否有权限调用该函数。
2. 如果此时播放器的状态为停止状态，直接返回 ALIVC_SUCCESS。
3. 其他情况则停止视频播放，并将播放器状态设置为停止状态，视频停止后会发送视频结束通知。

备注：调用该函数会释放音视频解码、渲染线程。如果需要重新进行播放，则需要再调用 `prepareToPlay` 重新对视频进行初始化。

seekTo

```
(AliVcMovieErrorCode) seekTo: (NSTimeInterval) newPos
```

功能：跳转指定的播放位置附近

参数：newPos，单位为毫秒

返回值：跳转成功 `ALIVC_SUCCESS`。否则返回失败。

seekTo 的具体过程为：

1. 验证是否有权限调用该函数。
2. 如果此时播放器的状态为停止或准备完成状态，返回错误的播放器状态 `ALIVC_ERR_ILLEGALSTATUS`。
3. 其他情况则进行视频跳转，跳转完成后会发送视频跳转结束通知 `AliVcMediaPlayerSeekingDidFinishNotification`。

备注：该函数仅允许在点播或播放本地视频过程中调用（直播禁用）。调用后视频会跳转到指定位置前最近的一个关键帧。参数的范围为 `[0,duration]`（duration 为视频的时长）。如果传入的参数小于 0，则播放器会自动将该参数修正到 0；如果传入参数大于 duration，则修正到 duration。

seekToAccurate

```
- (AliVcMovieErrorCode) seekToAccurate: (NSTimeInterval) newPos
```

功能：视频跳转到指定的播放位置。此函数能够精确的跳转到指定的位置，而 seekTo 只能跳转到附近的关键帧。

参数：newPos，单位为毫秒

备注：相比于 seekTo 函数，该函数跳转的位置更精准，但是函数执行的时间也更长。

destroy

```
(AliVcMovieErrorCode) destroy
```

功能：销毁播放器，该函数用来释放播放器的所有资源。

备注：在退出播放器的时候必须调用该函数，用来进行内存释放，否则会存在内存泄露。

reset

```
(AliVcMovieErrorCode) reset
```

功能：重置播放器。当播放的过程中调用该函数，会先停止当前的播放行为，销毁当前的播放器，然后创建一个新的播放器。

返回值：如果权限验证通过，则会返回 ALIVC_SUCCESS。

view

```
UIView *view
```

功能：设置包含视频内容的 view。

备注：需要在调用 prepareToPlay 之前设置 view，当前 view 只有视频帧图像，没有相关控制组件，相当于 iOS 系统播放器 MPMoviePlayerController 的控制方式 controlStyle 为 MPMovieControlStyleNone 的效果。

currentPosition

```
NSTimeInterval currentPosition
```

功能：获取当前视频播放位置，只读属性，获取单位为毫秒。

备注：当播放器状态为正在播放或暂停的状态，能够获取到有效值，否则获取值为无效的 0。

videoWidth

```
int videoWidth
```

功能：获取视频宽度，只读属性。

备注：当调用了 prepareToPlay 后，并不能立即获得 videoWidth 的值，只有当播放器发送了 prepared 通知后，videoWidth 的值才有效，否则为默认值 0。

videoHieght

```
int videoHeight
```

功能：获取视频高度，只读属性。

备注：当调用了 prepareToPlay 后，并不能立即获得 videoHeight 的值，只有当播放器发送了 prepared 通知后，videoHeight 的值才有效，否则为默认值 0。

livePlayer

```
int livePlayer
```

功能：设置播放器是直播还是点播模式，设置为 1 为直播，设置为 0 为点播.默认值为-1 不进行设置，由播放器自动判决。

备注：播放器自动判决会根据是否存在视频长度来判定是直播还是点播，此时会存在一定的偏差。如果用户知道自己的应用场景，应该设置此值使得播放器行为表现更加精确。

timeout

```
int timeout
```

功能：设置播放器网络超时时间，默认为 15000 毫秒。

备注：当播放网络视频时候，如果网络较差或者无网络的情况下，播放器此时会等待 timeout 的时间才会抛出网络异常的通知，用户收到此通知后进行处理，否则会一直等待。

dropBufferDuration

```
int dropBufferDuration
```

功能：当播放器处理直播模式时，设置缓冲区开始丢帧的起始长度，默认为 8000 毫秒。

备注：当缓冲区的长度大于 dropBufferDuration 时候，播放器开始启动丢帧策略，此时必须要保证缓冲区至少有一个 I 帧，否则会引起花屏，所以此值必须大于一个 GOP 的长度。

duration

```
NSTimeInterval duration
```

功能：获取视频时长，只读属性，单位为毫秒。

备注：当调用了 prepareToPlay 后，并不能立即获得 duration 的值，只有当播放器发送了 prepared 通知后，duration 的值才有效，否则为默认值 0。

bufferingPosition

```
NSTimeInterval bufferingPostion
```

功能：获取当前播放器已经缓冲好的位置。

备注：开始播放后，可以获取此值，用来获取已经下载好的位置。

muteMode

```
BOOL muteMode
```

功能：设置播放器是否静音，YES 位静音。

备注：静音指播放器的静音，并不会影响系统音量。

scalingMode

```
ScalingMode scalingMode
```

功能：设置播放器渲染时的缩放模式，目前有两种模式，scalingModeAspectFit：等比例缩放显示，如果视频长宽比和屏幕长宽比不一致时，会存在黑边；scalingModeAspectFitWithCropping：带裁边的等比例缩放，如果视频长宽比和屏幕长宽比不一致时，会进行裁边处理以保持全屏显示。

备注：默认为 `scalingModeAspectFitWithCropping` 模式，可以动态改变。

errorCode

```
AliVcMovieErrorCode errorCode
```

功能：播放器错误代码，只读属性。

备注：当播放器出现错误时，可以通过获取这个属性来获取到错误的代码，通过错误代码可以了解到具体的错误原因。

getSDKVersion

```
-(NSString *) getSDKVersion
```

功能：获取 SDK 版本号。

返回值：返回 NSString 类型的版本号。

备注：当开发者反馈问题时，请使用该方法获得 SDK 版本号并随同问题一起反馈。

getPropertyDouble

```
-(double) getPropertyDouble:(int)property defaultValue:(double)defaultValue
```

功能：获取一些性能参数。

返回值：返回 double 型的参数。

备注：同类型的函数还有 `getPropertyLong`，`getPropertyString` 等。这些键值如下：

标识符	数值
FFP PROP DOUBLE OPEN FORMAT_TIME	18001
FFP PROP DOUBLE FIND STREAM_TIME	18002
FFP PROP DOUBLE OPEN STREAM_TIME	18003
FFP PROP DOUBLE 1st VFRAME_ SHOW_TIME	18004
FFP PROP DOUBLE 1st AFRAME_ SHOW_TIME	18005
FFP PROP DOUBLE 1st VPKT_ GET_TIME	18006
FFP PROP DOUBLE 1st APKT_ GET_TIME	18007
FFP PROP DOUBLE 1st VDECODE_TIME	18008

标识符	数值
FFP <i>PROP</i> DOUBLE <i>1st</i> ADECODE_TIME	18009
FFP <i>PROP</i> DOUBLE_ DECODE_TYPE	18010
FFP <i>PROP</i> DOUBLE <i>LIVE</i> DISCARD_DURATION	18011
FFP <i>PROP</i> DOUBLE <i>LIVE</i> DISCARD_CNT	18012
FFP <i>PROP</i> DOUBLE <i>DISCARD</i> VFRAME_CNT	18013
FFP <i>PROP</i> DOUBLE <i>RTMP</i> OPEN_DURATION	18040
FFP <i>PROP</i> DOUBLE <i>RTMP</i> OPEN_RTYCNT	18041
FFP <i>PROP</i> DOUBLE <i>RTMP</i> NEGOTIATION_DURATION	18042
FFP <i>PROP</i> DOUBLE <i>HTTP</i> OPEN_DURATION	18060
FFP <i>PROP</i> DOUBLE <i>HTTP</i> OPEN_RTYCNT	18061
FFP <i>PROP</i> DOUBLE <i>HTTP</i> REDIRECT_CNT	18062
FFP <i>PROP</i> DOUBLE <i>TCP</i> CONNECT_TIME	18080
FFP <i>PROP</i> DOUBLE <i>TCP</i> DNS_TIME	18081
FFP <i>PROP</i> INT64 <i>VIDEO</i> CACHED_DURATION	20005
FFP <i>PROP</i> INT64 <i>AUDIO</i> CACHED_DURATION	20006
FFP <i>PROP</i> INT64 <i>VIDEO</i> CACHED_BYTES	20007
FFP <i>PROP</i> INT64 <i>AUDIO</i> CACHED_BYTES	20008
FFP <i>PROP</i> INT64 <i>VIDEO</i> CACHED_PACKETS	20009
FFP <i>PROP</i> INT64 <i>AUDIO</i> CACHED_PACKETS	20010
FFP <i>PROP</i> INT64 <i>SELECTED</i> VIDEO_STREAM	20001
FFP <i>PROP</i> INT64 <i>SELECTED</i> AUDIO_STREAM	20002

版本更新说明

1.0 原始版本，调用硬件的 mediaPlayer 接口实现

2.0 改用硬件的 mediaCodec 接口实现

2.1 增加直播秒开功能、缓冲区丢帧策略等。

2.2 支持模拟器、支持多实例、支持 https、增加静音功能、添加视频渲染时的缩放模式