**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](https://help.aliyun.com/document_detail/29938.html)

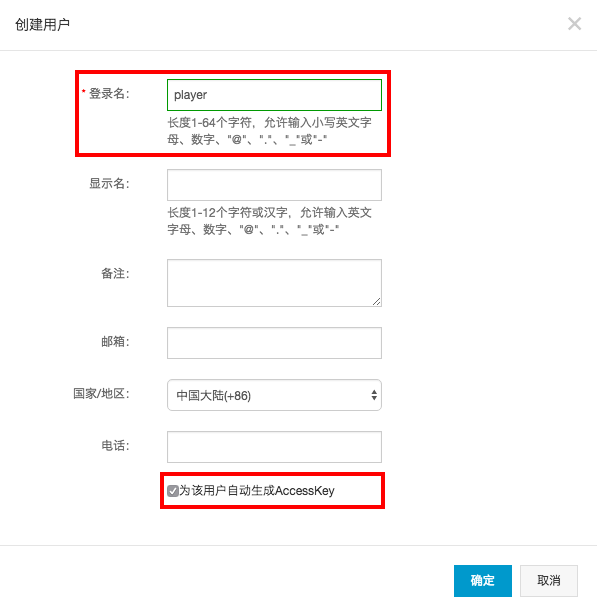
[视频直播服务开通](https://help.aliyun.com/document_detail/29955.html)

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](https://ram.console.aliyun.com/" \l "/overview)
* b. 在用户管理中新建用户：



注意勾选为该用户自动生成AccessKey 选项：



创建子帐号成功，注意保存好该帐号的AccessKey：



* c. 为子帐号分配调用播放器权限：

点击**授权**链接：



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



**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];

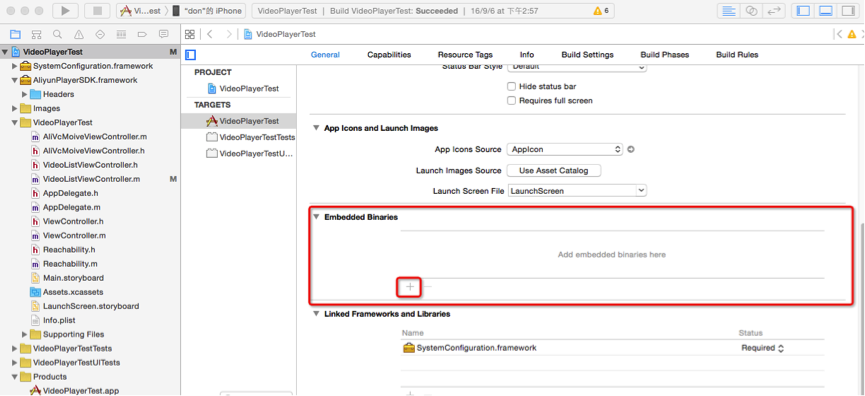
权限验证可以在播放器创建之前进行，且播放多个视频只需要验证一次即可。通过协议的getAccessKeyIDSecret接口，sdk可以获取用户的AccessKeyID和AccessKeySecret来完成验证。

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. }

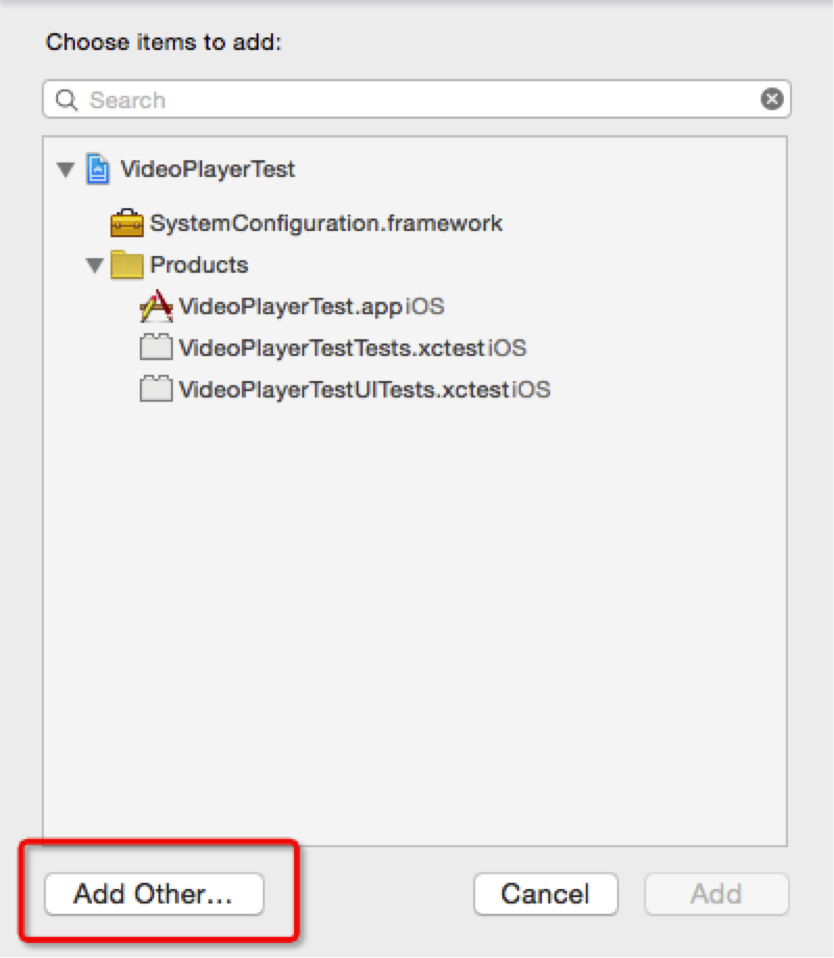
**sdk包添加**

使用xcode创建工程的方法如下：

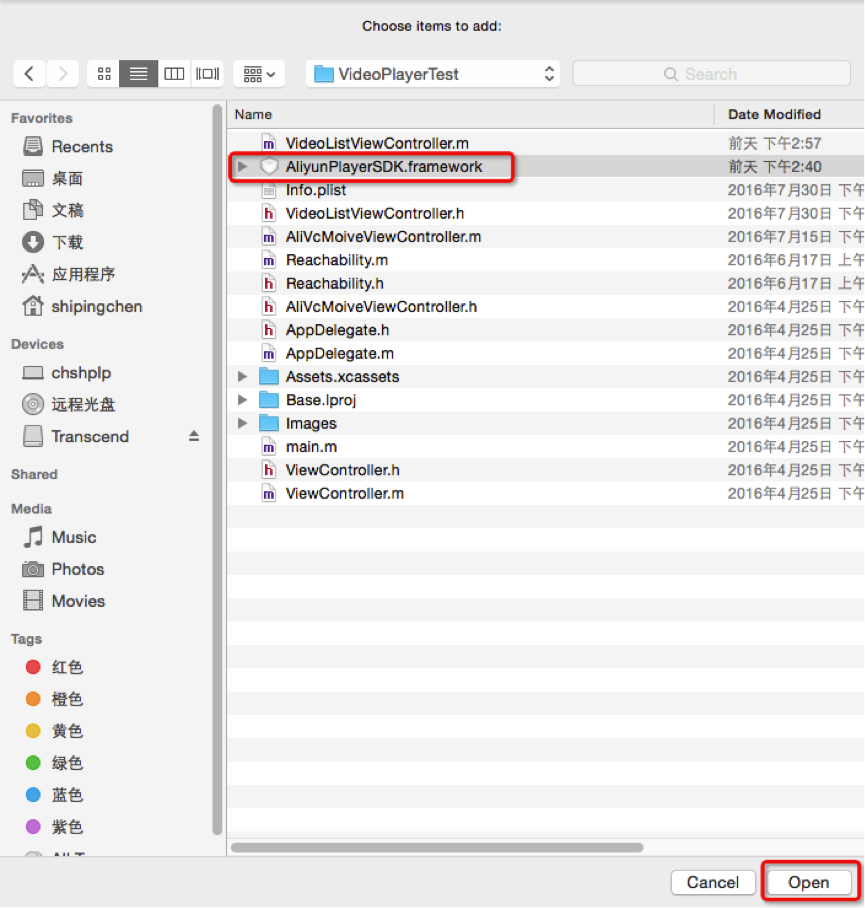
1. XCode创建一个iOS应用工程。
2. 将SDK中的framework添加到工程中，如图：



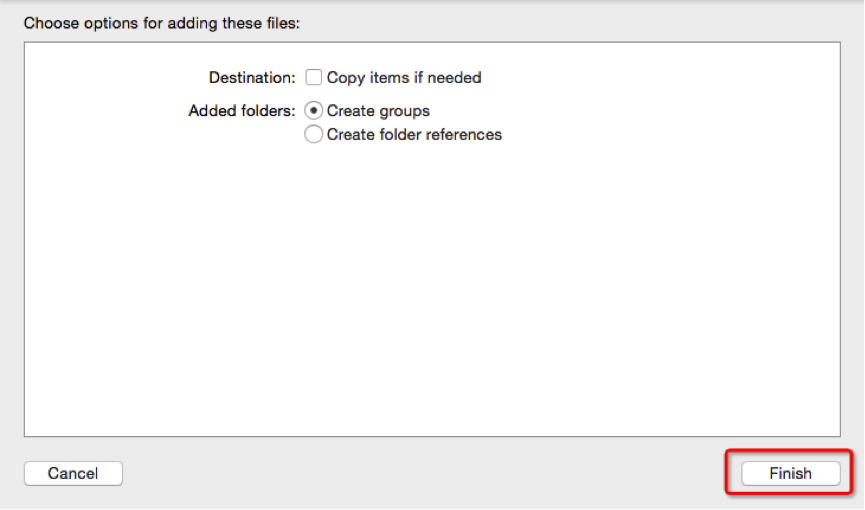
点击“+”号后出现：



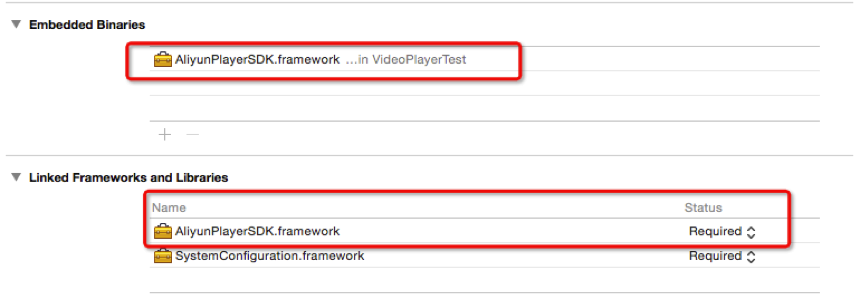
点击“Add Other…”，选中framework.



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. }
8. -(void) playVideo
9. {
10. //新建播放器
11. player = [[AliVcMediaPlayer alloc] init];
12. //创建播放器，传入显示窗口
13. [player create:mShowView];
14. //注册准备完成通知
15. [[NSNotificationCenter defaultCenter] addObserver:self
16. selector:@selector(OnVideoPrepared:) name:AliVcMediaPlayerLoadDidPreparedNotification object:player];
17. //注册错误通知
18. [[NSNotificationCenter defaultCenter] addObserver:self
19. selector:@selector(OnVideoError:) name:AliVcMediaPlayerPlaybackErrorNotification object:player];
20. //传入播放地址，初始化视频，准备播放
21. [player prepareToPlay:mUrl];
22. //开始播放
23. [player play];
24. }

**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. -(void)OnVideoError:(NSNotification \*)notification
8. {
9. AliVcMovieErrorCode error\_code = player.errorcode;
10. }

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 |
| 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 |
| 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 |
| FFP*PROP* DOUBLE*1st* ADECODE\_TIME | 18009 |
| FFP*PROP* DOUBLE\_ DECODE\_TYPE | 18010 |
| FFP*PROP* DOUBLE*LIVE* DISCARD\_DURATION | 18011 |
| FFP*PROP* DOUBLE*LIVE* DISCARD\_CNT | 18012 |
| FFP*PROP* DOUBLE*DISCARD* VFRAME\_CNT | 18013 |
| FFP*PROP* DOUBLE*RTMP* OPEN\_DURATION | 18040 |
| FFP*PROP* DOUBLE*RTMP* OPEN\_RTYCNT | 18041 |
| FFP*PROP* DOUBLE*RTMP* NEGOTIATION\_DURATION | 18042 |
| FFP*PROP* DOUBLE*HTTP* OPEN\_DURATION | 18060 |
| FFP*PROP* DOUBLE*HTTP* OPEN\_RTYCNT | 18061 |
| FFP*PROP* DOUBLE*HTTP* REDIRECT\_CNT | 18062 |
| FFP*PROP* DOUBLE*TCP* CONNECT\_TIME | 18080 |
| FFP*PROP* DOUBLE*TCP* DNS\_TIME | 18081 |
| FFP*PROP* INT64*VIDEO* CACHED\_DURATION | 20005 |
| FFP*PROP* INT64*AUDIO* CACHED\_DURATION | 20006 |
| FFP*PROP* INT64*VIDEO* CACHED\_BYTES | 20007 |
| FFP*PROP* INT64*AUDIO* CACHED\_BYTES | 20008 |
| FFP*PROP* INT64*VIDEO* CACHED\_PACKETS | 20009 |
| FFP*PROP* INT64*AUDIO* CACHED\_PACKETS | 20010 |
| FFP*PROP* INT64*SELECTED* VIDEO\_STREAM | 20001 |
| FFP*PROP* INT64*SELECTED* 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