阿里巴巴集团

SDK 参考手册

iOS 媒体播放器 SDK 参考手册

文档修订历史

版本	作者	工作描述	修订历史	修改日期
1.0	石惊	iOS 媒体播放器 SDK 文档		2015-08-03
2.0	石惊	iOS 媒体播放器 SDK 文档		2016-05-24
2.1	石惊	播放器对首帧、卡顿、延迟等进行了优化，增加了相应的接口；		2016-07-11

参考文献

无

目录

文档说明

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。

1.简介

iOS媒体播放器SDK是在iOS平台上使用的软件开发工具包(Soft Development Kit)，为iOS开发者提供简单易用的接口，帮助开发者实现iPhone/iPad/iPod等平台上的媒体播放应用开发。

1.1 功能说明

1. 方便快捷、低门槛实现媒体播放功能。用户无须关心内部实现细节，只需要自定义界面既可以实现专业级的播放应用。
2. 流媒体格式支持HLS、RTMP、HTTP FLV。
3. 文件媒体格式支持mp4。
4. 视频压缩格式支持h264，音频压缩格式支持AAC。

1.2 安装包说明

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。
3. doc：存放SDK相关接口文档。

1.3 播放器性能

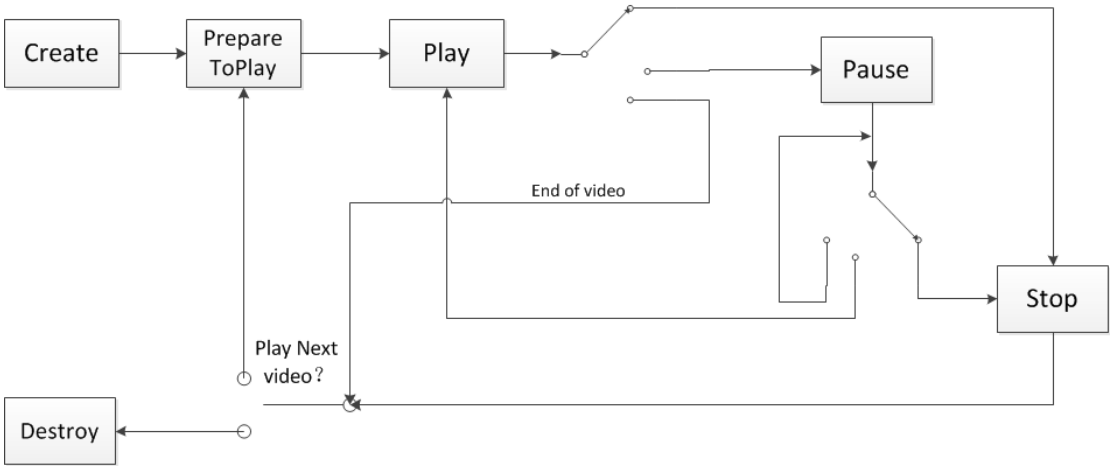
1. 目前播放器SDK对H.264、AAC的视频流采用硬件解码，播放1080P视频CPU占用在10-20%之间。
2. SDK的大小：目前SDK的大小为7.1M，包括armv7和arm64两种平台。

1.4 注意事项

1. 播放器SDK目前只支持单实例播放，不能够同时开2个或以上的播放器实例。需要另开实例的时候，必须关闭之前存在的实例。
2. 操作系统版本要求ios8.0以上。

2. 系统框架

2.1 系统框架图



2.2 系统框架图

无

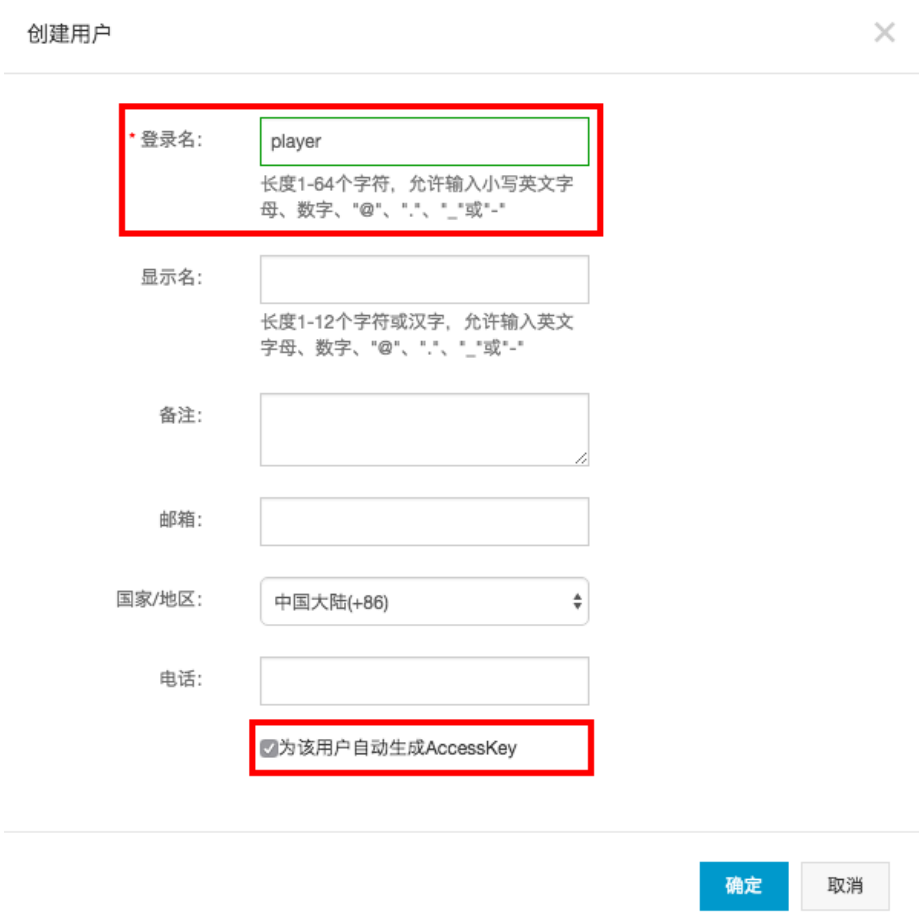
3. 使用说明

3.1 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。
   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：

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：

3.2 设置用户权限

将用户申请的AccessKeyID和AccessKeySecret进行设置，播放器才能够使用。用户需要实现以下AliVcAccessKeyProtocol协议来获取用户的AccessKeyID和AccessKeySecret。

[AliVcMediaPlayer setAccessKeyDelegate:self];

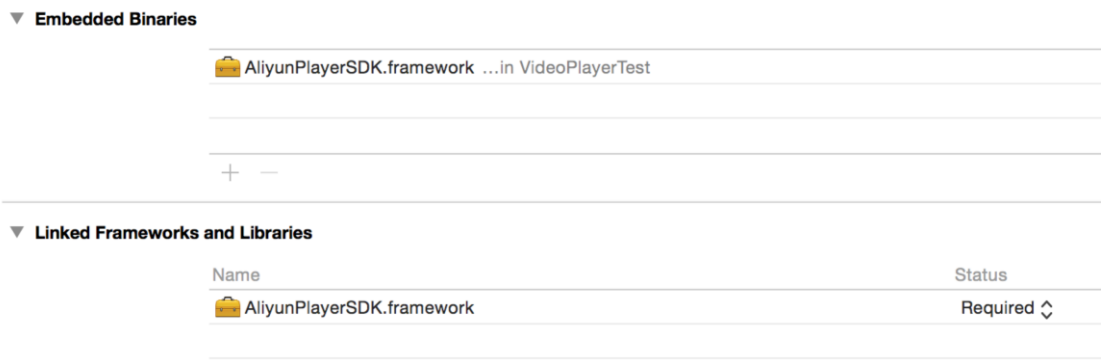
验证会每个小时进行一次验证，用户需要实现该协议的getAccessKeyIDSecret接口来获取用户的AccessKeyID和AccessKeySecret。

```
NSString* accessKeyID = @"QxJIheGFRL926hFX";
NSString* accessKeySecret = @"hipHJKpt0TdznQG2J4D0EVSavRH7mR";
-(AliVcAccesskey*)getAccessKeyIDSecret
{
    AliVcAccesskey* accessKey = [[AliVcAccesskey alloc] init];
    accessKey.accessKeyId = accessKeyID;
    accessKey.accessKeySecret = accessKeySecret;
    return accessKey;
}
```

3.3 SDK 包添加

请参考以下步骤，将播放器SDK添加到XCode工程中：

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



3. 将SDK中的头文件包含到工程中。

3.4 开发步骤

使用iOS媒体播放器SDK的调用顺序为：

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。

3.5 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。

下面给出了部分重要的 Demo 中调用 SDK 的代码：

```
-(void) playVideo
{
    //新建播放器
    player = [[AliVcMediaPlayer alloc] init];
    //创建播放器，传入显示窗口
    [player create:mShowView];
    //注册准备完成通知
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(OnVideoPrepared:)
name:AliVcMediaPlayerLoadDidPreparedNotification object:player];
    //注册错误通知
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(OnVideoError:)
name:AliVcMediaPlayerPlaybackErrorNotification object:player];
    //授权验证
    [player getUserPriority:@"WvYZ1RVw3EJm2bxj"
keySecret:@"TWDcZszk00iEbH0YHLVp7UtbIMKv2g&"];
    //传入播放地址，初始化视频，准备播放
    [player prepareToPlay:mUrl];
    //开始播放
    [player play];
```

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

4.接口说明

4.1 SDK 接口

SDK 中提供了类 AliVcMediaPlayer 来实现播放器各种功能接口，同时，我们对播放器的各种通知进行了定义，并且定义播放器的各种错误代码。

播放器功能接口	
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
view	设置播放显示窗口
videoHeight	获取视频高度
videoWidth	获取视频宽度
mediaType	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
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

播放器通知事件	
AliVcMediaPlayerLoadDidPreparedNotification	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

播放器错误代码 AliVcMovieErrorCode	
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	视频格式编解码不支持

ALIVC _ERR_NO_MEMORY	没有足够的内存

下面详细介绍一下各个成员函数的具体使用：

1. - (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。

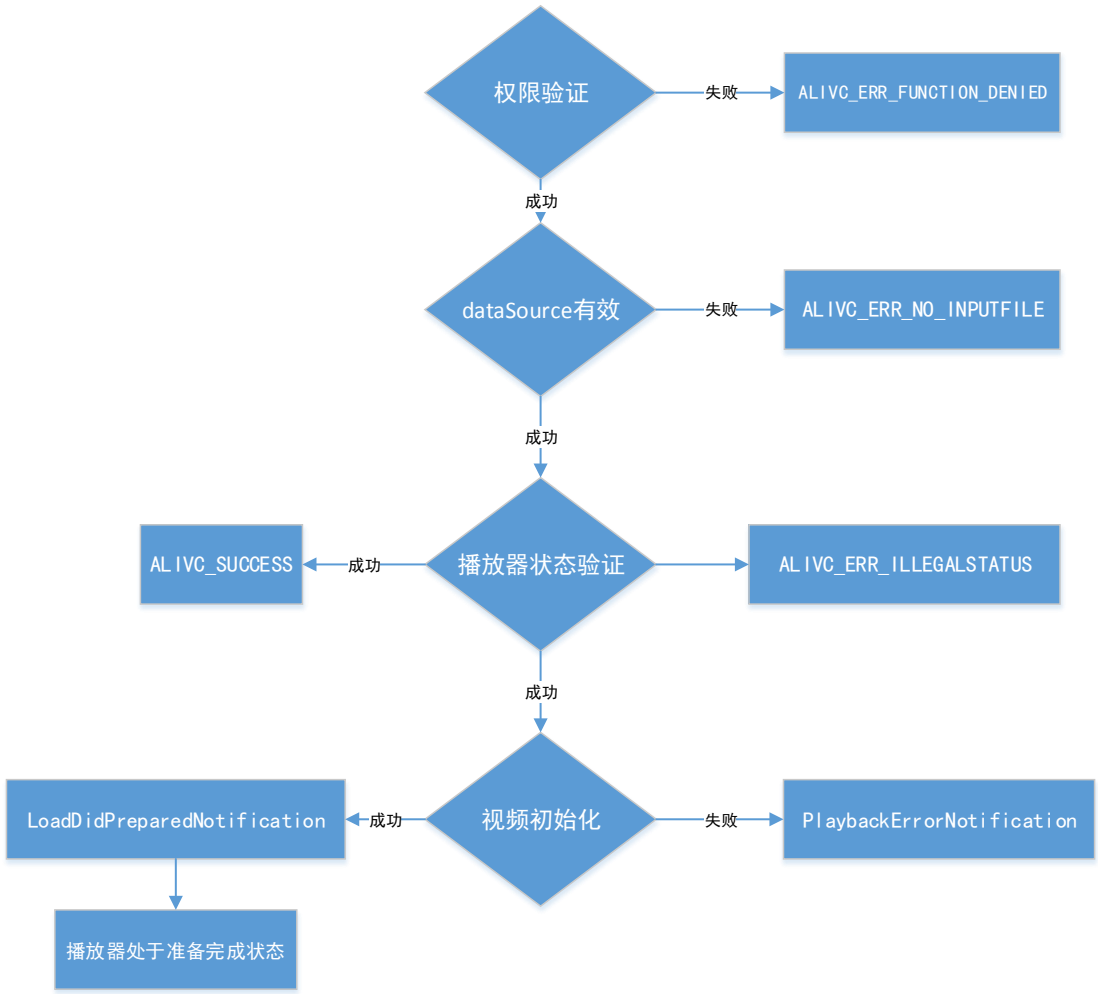
2. - (AliVcMovieErrorCode) prepareToPlay: (NSURL*)dataSource

   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、显示线程等。

   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。

   下图为函数的逻辑框图：



具体过程为：

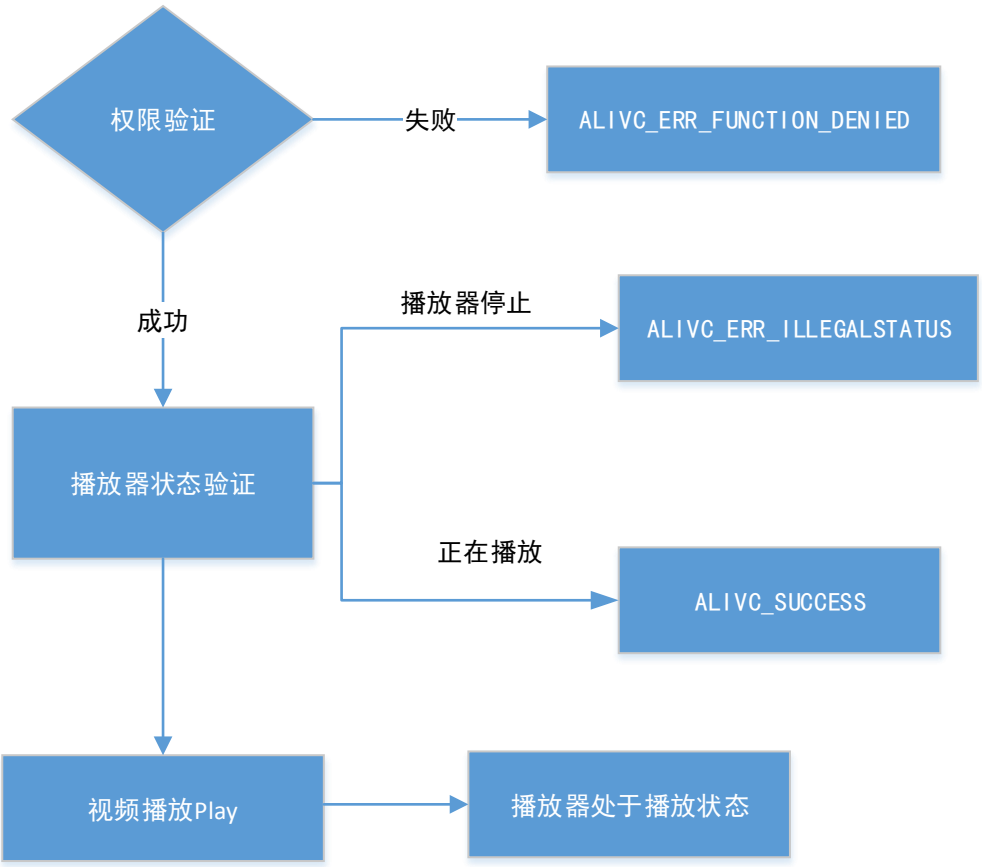
1. 验证用户是否有权限调用该函数。
2. 验证参数 dataSource 是否为空。
3. 如果播放器是正在播放或者正在暂停状态，则不能够进行 prepare，此时返回非法的播放流程错误 ALIVC_ERR_ILLEGALSTATUS，如果播放器状态是已经是准备完成状态，则返回 ALIVC_SUCCESS。
4. 对视频进行初始化，如果成功，则会发送 LoadDidPreparedNotification 通知，表示视频初始化完成。如果失败则会发送 PlaybackErrorNotification 通知，在错误通知中可以获取到错误代码。

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

3. - (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，具体失败代码见下述逻辑：



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。

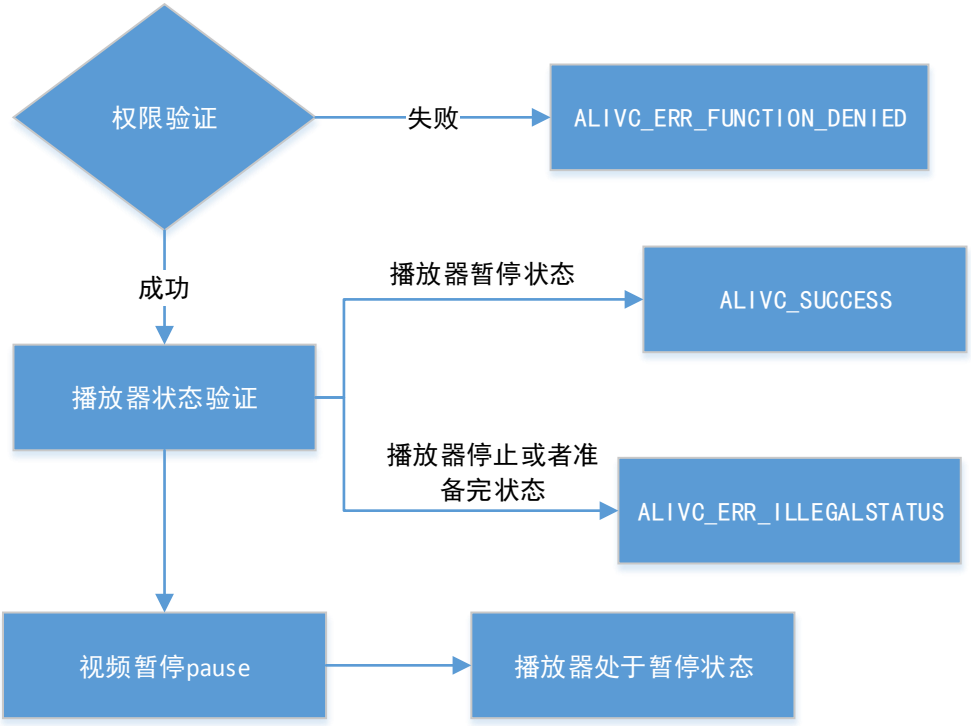
备注：播放器调用 play 进行播放，必须在播放器状态为准备完成的状态或者暂停的状态才能进行播放，其他情况都不能够将视频播放起来。

4. - (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。具体逻

辑如下：



具体过程为：

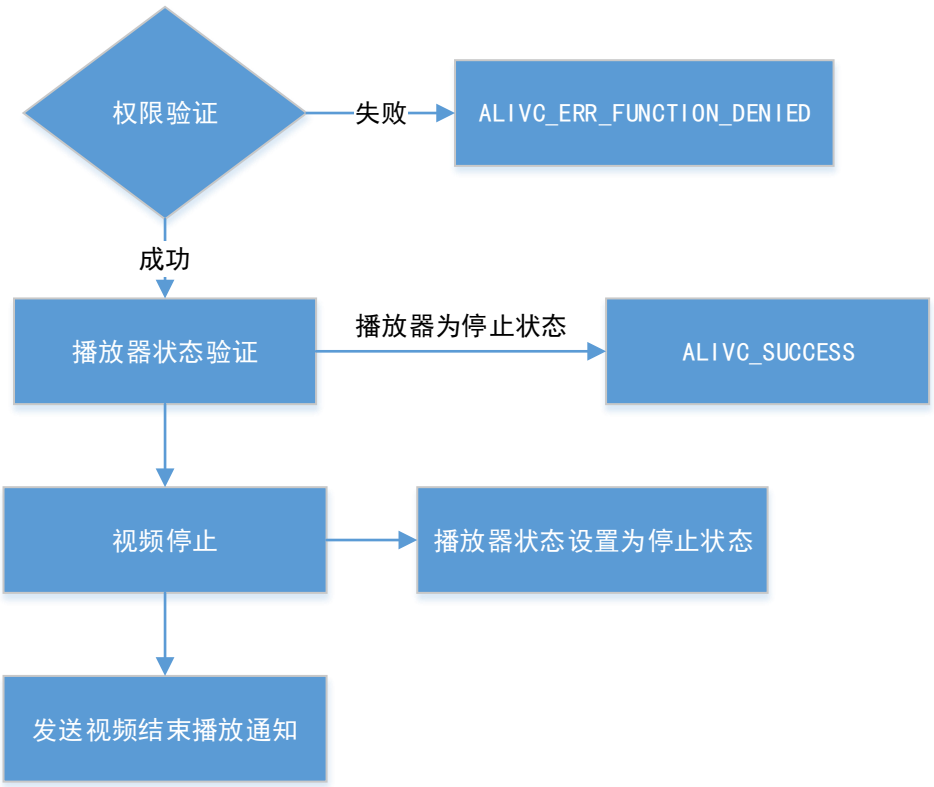
1. 验证是否有权限调用该函数。
2. 如果此时播放器为暂停状态，直接返回 ALIVC_SUCCESS。
3. 如果播放器状态为停止或者准备完成状态则返回错误的播放器状态 ALIVC_ERR_ILLEGALSTATUS。
4. 其他情况则暂停视频播放，并将播放器状态设置为暂停状态。

备注：播放器调用 pause 暂停视频播放，一般是在视频播放的情况下调用此函数。

5. - (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。具体逻辑如下：



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。

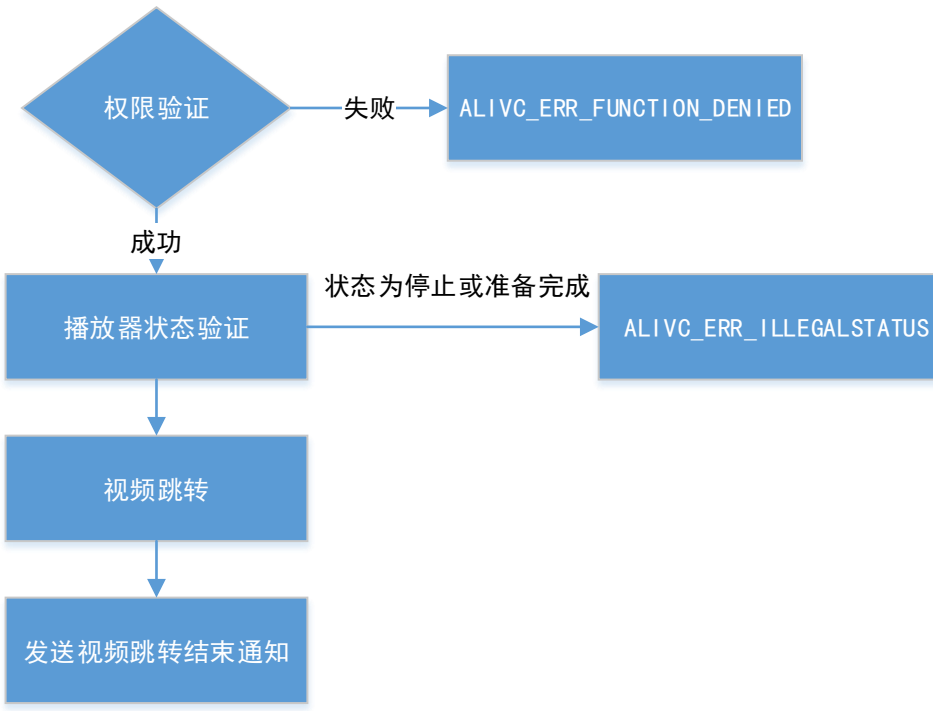
备注：该函数会停止掉内部音视频解码线程，如果需要重新进行播放，则需要再调用 prepareToPlay 进行重新对视频进行初始化。

6. - (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。具体逻辑如下：



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：该函数仅在点播或本地视频中允许调用。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]。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。

7. - (AliVcMovieErrorCode) reset

功能：重置播放器，其目的是将播放器所有状态恢复到最初位置，当播放器内部出现错误或其它问题时，调用此函数来进行恢复。

返回值：如果权限验证通过，则会返回 ALIVC_SUCCESS。

备注：该函数和 stop 函数的区别是，reset 也能够实现 stop 的功能，但是 reset 会去销毁播放器内部的各种变量，并且调用 create 重新初始化播放器内部各个变量。

8. - (AliVcMovieErrorCode) destroy

功能：销毁播放器，该函数用来释放播放器内部的所有变量，退出所有

线程。

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。

9. UIView *view

功能：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。

10. 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 0。

11. 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。

12. 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。

13. MediaType medaType

功能：设置播放器是直播还是点播模式，设置为 MediaType_LIVE 为直播，设置为 MediaType_VOD 为点播，默认为 MediaType_AUTO 不进行设置，由播放器自动判决。

备注：建议在可以清晰分辨视频类型的情况下尽量调用该函数。如果不调用，则播放器会自动根据视频的 duration 来判断媒体类型。Duration 为 0

且格式为 HLS、rtmp、http flv 的为直播类视频，其他为点播类视频。

14. 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。

15. 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 。

备注：该函数仅对直播场景有效，主要用于缩短主播与观众之间的时间延迟。当缓冲区中的视频时长超过设置的 duration 时，播放器会自动丢弃部分音视频数据，以减少延迟。系统默认 rtmp、http flv 直播时 duration 为 8 秒，HLS 直播时 duration 为 40 秒。建议 rtmp、http flv 直播时，duration 的值超过 GOP 时长（即两个关键帧之间的时间长度）的 2 倍；HLS 直播时，duration 的值超过 m3u8 文件中所有 ts 分片的总时长。这样可以避免出现经常性的视频丢帧。

16. 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。

17. NSTimeInterval bufferingPostion

功能：获取当前播放器已经缓冲好的位置，只读属性，单位为毫秒。

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。

18. 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。

19. - (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。

20. -(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。这些键值是：

FFP_PROP_DOUBLE_OPEN_FORMAT_TIME

FFP_PROP_DOUBLE_FIND_STREAM_TIME

FFP_PROP_DOUBLE_OPEN_STREAM_TIME

FFP_PROP_DOUBLE_1st_VFRAME_SHOW_TIME

FFP_PROP_DOUBLE_1st_AFRAME_SHOW_TIME

FFP_PROP_DOUBLE_1st_VPKT_GET_TIME

FFP_PROP_DOUBLE_1st_APKT_GET_TIME

FFP_PROP_DOUBLE_1st_VDECODE_TIME

FFP_PROP_DOUBLE_1st_ADECODE_TIME

FFP_PROP_DOUBLE_DECODE_TYPE

FFP_PROP_DOUBLE_LIVE_DISCARD_DURATION

FFP_PROP_DOUBLE_LIVE_DISCARD_CNT

FFP_PROP_DOUBLE_DISCARD_VFRAME_CNT

FFP_PROP_DOUBLE_RTMP_OPEN_DURATION

FFP_PROP_DOUBLE_RTMP_OPEN_RTYCNT

FFP_PROP_DOUBLE_RTMP_NEGOTIATION_DURATION

FFP_PROP_DOUBLE_HTTP_OPEN_DURATION

FFP_PROP_DOUBLE_HTTP_OPEN_RTYCNT

FFP_PROP_DOUBLE_HTTP_REDIRECT_CNT

FFP_PROP_DOUBLE_TCP_CONNECT_TIME

FFP_PROP_DOUBLE_TCP_DNS_TIME

FFP_PROP_INT64_VIDEO_CACHED_DURATION

FFP_PROP_INT64_AUDIO_CACHED_DURATION

FFP_PROP_INT64_VIDEO_CACHED_BYTES

FFP_PROP_INT64_AUDIO_CACHED_BYTES

FFP_PROP_INT64_VIDEO_CACHED_PACKETS

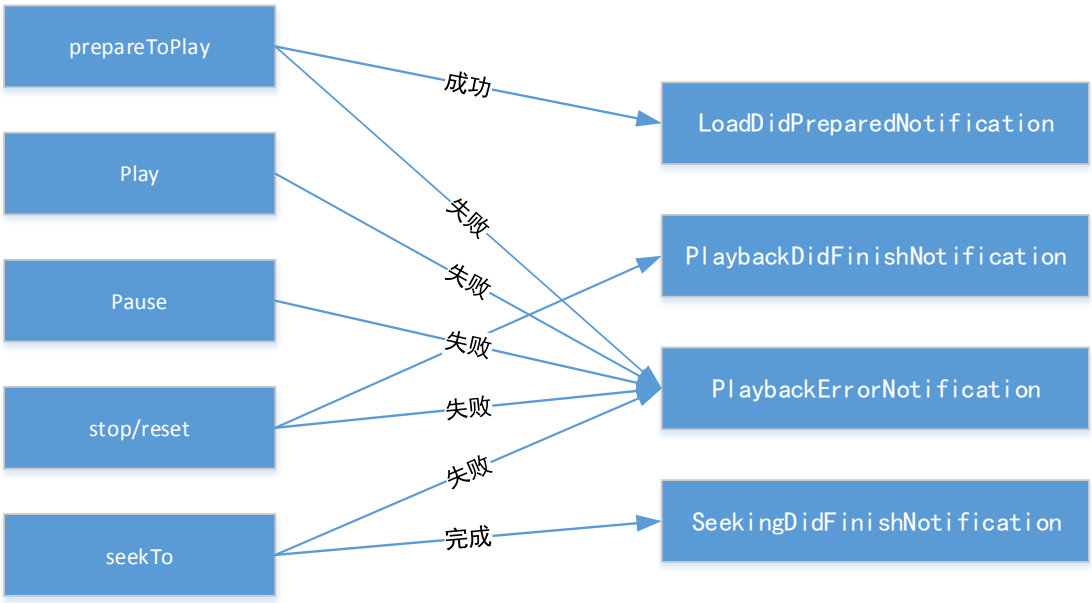
FFP_PROP_INT64_AUDIO_CACHED_PACKETS

FFP_PROP_INT64_SELECTED_VIDEO_STREAM

FFP_PROP_INT64_SELECTED_AUDIO_STREAM

4.2 播放器通知发送逻辑

下图是播放器发送通知逻辑：



具体逻辑：

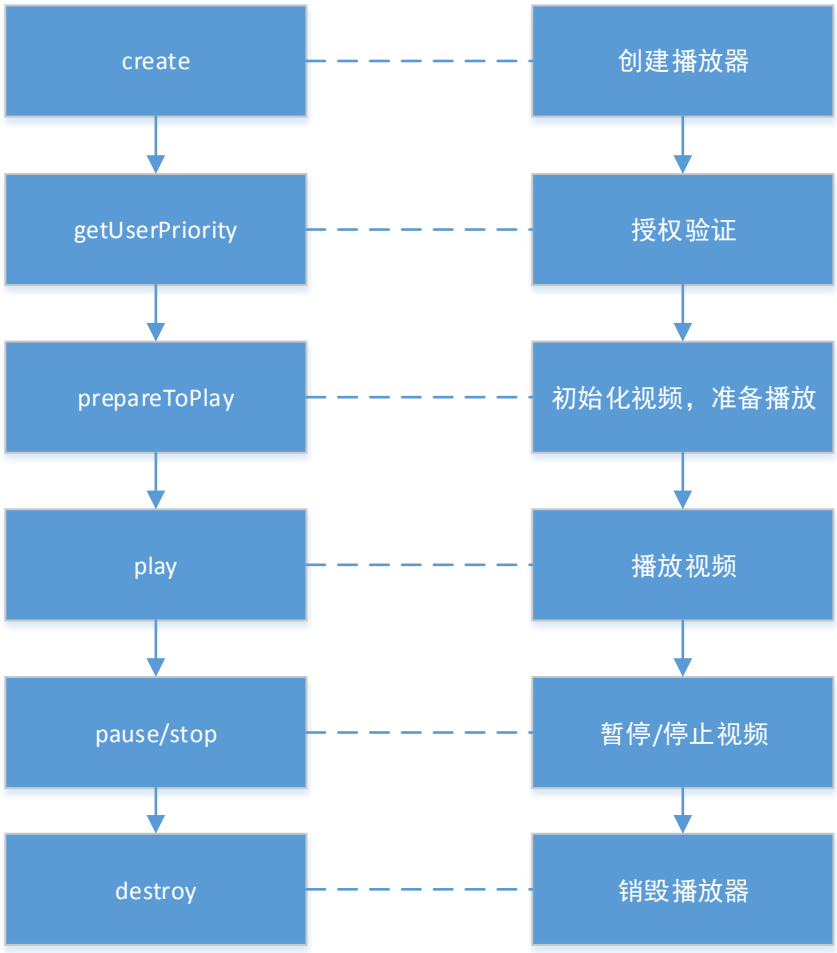
1. 调用 prepareToPlay 成功后发送 LoadDidPreparedNotification 通知，失败则会发送 PlaybackErrorNotification。
2. 调用 play、pause、stop、reset、seekTo 和 prepareToPlay 失败后发送 PlaybackErrorNotification 通知。
3. 调用 stop/reset 成功后播放视频结束发送 PlaybackDidFinishNotification 通知。
4. 调用 seekTo 成功后发送 SeekingDidFinishNotification 通知。

备注：同时还有 StartCachingNotification 和 EndCachingNotification 通知，这

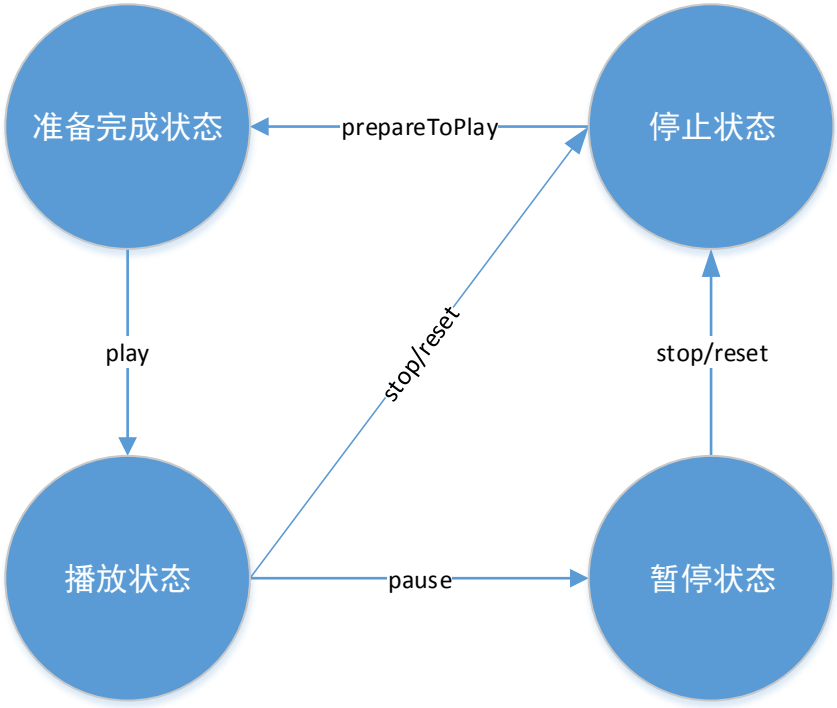
个是在网络视频缓冲数据不足以够播放后会发送此通知，一般网络视频在调用 seekTo 后会发送此通知。

另外，当收到播放器错误通知后，可以获取到错误代码，我们将错误代码分成了两个级别，一种错误时播放器产生的严重错误，需要用户来调用 stop 或 reset 来进行重置播放器。根据 ALIVC_ERR_RESET_LINE 来区分，如果大于 ALIVC_ERR_RESET_LINE 则播放器需要重置。

4.3 播放器接口调用流程



上图是播放器播放视频的流程，播放器内部会记录播放器的状态，状态的转换逻辑如下图：



5. 注意事项

无

6. 版权声明

版权所有，切勿盗版