

乐视云播放SDK4.0-iOS端使用文档

乐视云播放SDK4.0-iOS端使用文档	1
文档版本说明	4
功能说明	4
1. 简介	4
2. 模块结构	4
2.1 类图	4
2.2 简述	5
3.有UI SDK	5
4.无UI SDK	5
3.1 直播/点播播放器	5
3.1.1 播放器生命周期	5
3.1.2 概述	6
3.1.3 打断/后台处理	6
3.1.4 播放结束/出错处理	6
接入说明	6
1.下载SDK	6
2.工程配置	6
2.1在工程中选择性引入静态库文件	6
2.2 设置静态库头文件路径以及静态库路径	7
2.3 在工程中引入需要的Bundle文件	8
2.4 设置Other Linker Flags	8
2.5 设置依赖的Frameworks	9
2.6 关闭BITCODE的开关	10
2.7 启动服务	10
2.8 模拟器调试存在的问题	11
2.9 申报AppStore广告IDFA相关问题	11
官网注册SDK相关说明	11
1.注册账号	11
2.完善配置	12
3.联系我们	12
iOS播放器SDK皮肤版本使用说明	12

1.版本信息	12
2.概述	12
3.类文件介绍	12
4.属性说明	13
5.API说明	13
6.代码示例	16
6.1工程配置	16
6.2主要代码	16
iOS播放器SDK无皮肤版本接口文档	17
1.版本信息	17
2.概述	17
3.基类播放器相关接口	18
3.1概述	18
3.2 LECPlayer	18
3.3 LECVODPlayer	23
3.4 LECLivingPlayer	25
3.5 LECActivityPlayer	26
3.6 LCVODDownloadManager	28

文档版本说明

版本号	版本说明	更新人	日期
1.0	SDK说明文档	郭子龙	2016.02.02

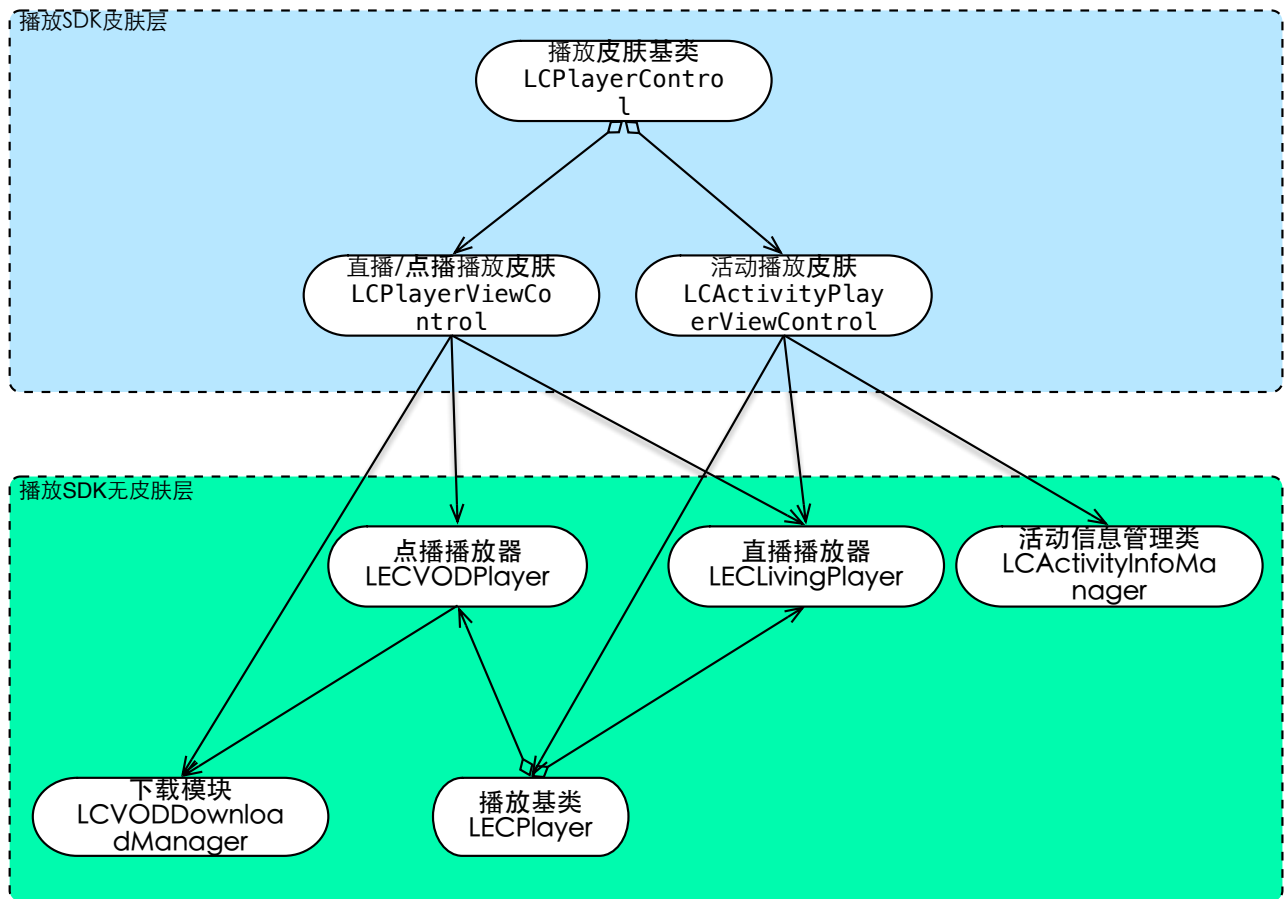
功能说明

1. 简介

乐视云计算播放SDK(以下简称播放SDK)为客户提供方便接入乐视云计算云直播、云点播等业务的客户端接口。对于需要快速接入的客户，可以使用播放SDK有皮肤的相关接口,只需要简单的几行代码即可完成客户端的集成工作。对于需要自定义播放器皮肤的客户,可以使用播放SDK无肤的相关接口。

2. 模块结构

2.1 类图



2.2 简述

类图中涉及类都为公共类，户可直接调，目前播放SDK分为有皮肤层和皮肤层，两层相关类的源码并不开源;肤层相关类暴露了一些自定义皮肤的方法，例如替换缓冲图片等。如果需要深度定制皮肤，或在播放器皮肤上增加功能则需要用户对播放SDK无皮肤层相关类，绘制上层皮肤。

3.有UI SDK

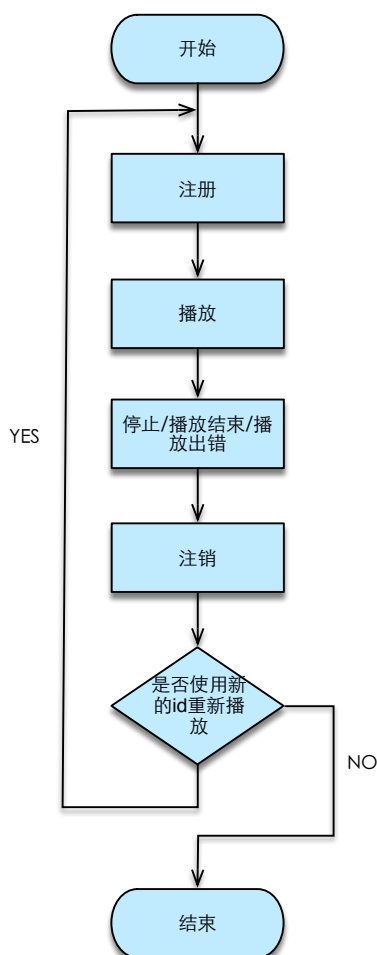
播放SDK有皮肤层相关类给用户暴露接口简单，便于用户接入。相对的用户对于皮肤与播放的控制权就比较少。如果用户需要自定义皮肤，可以参照皮肤层类的类关系(2.1类图)，仿照着DEMO完成 定义皮肤的开发。

对于需要对接直播和点播的用户，可以使皮肤层的LCPlayerViewController类。对于需要对接活动直播的用户(持多机位直播)，可以使皮肤层的LCActivityPlayerViewController类。关于后台以及打断事件，用户不需要做额外处理，SDK内部会恢复到被打断前的状态。

4.无UI SDK

3.1 直播/点播播放器

3.1.1 播放器生命周期



3.1.2 概述

UI播放SDK需要用户传入标识视频唯一性的ID作为注册参数，对于点播是uu、vu，对于直播是liveId或streamId。相关参数可以通过官网前端页面获取，也可以通过云直播/云点播API获取，具体请参照相关文档：<http://help.letvcloud.com/Wiki.jsp?page=DevelopmentGuide>

在使用过程中需遵循播放器的生命周期，不再需要播放器时需要调用SDK的注销方法。

3.1.3 打断/后台处理

与皮肤层相同，在被打断或进入后台时不需要用户做额外的处理，播放器会主动恢复到打断前的状态。

3.1.4 播放结束/出错处理

在播放结束(EOS)或播放出错(Error)的状态时，播放器内部会自行执行stop操作。对于上层可以再次调用play进行重试操作或者调用注销方法注销播放SDK。

接入说明

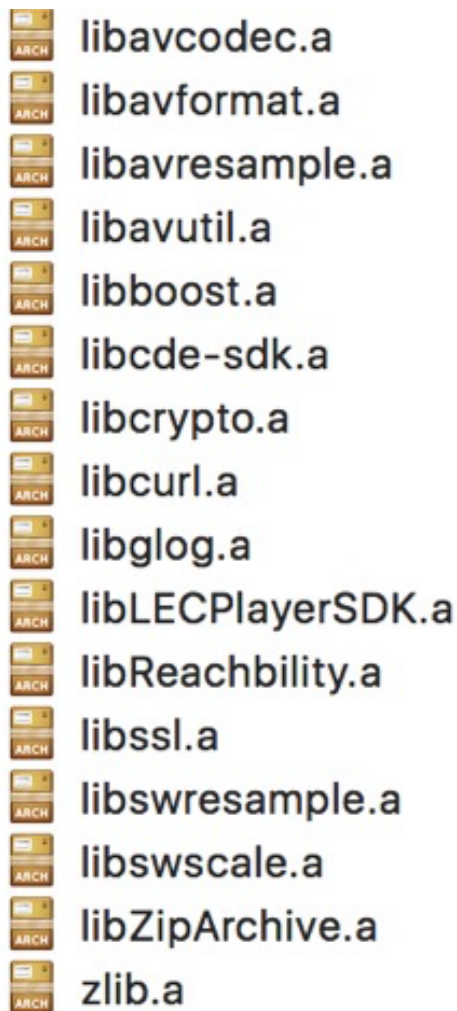
1. 下载SDK

前往[下载中心下载](#)最新版本播放SDK并解压缩

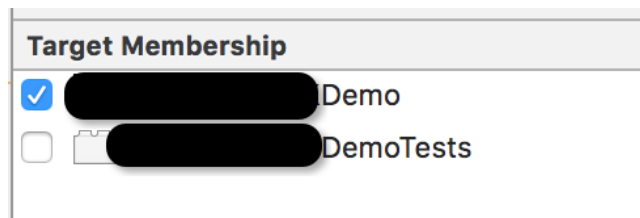
2. 工程配置

2.1 在工程中选择性引入静态库文件

在工程目录结构中，点击右键->Add Files to“工程名”。在打开的文件选择器中找到SDK中提供的.a文件并加入到工程。此处需要注意，SDK中引入了一些第三方静态库，如果这库和你的工程中的冲突可以不重复引用，如libZipArchive，SDK所包含的第三方依赖库见下图：

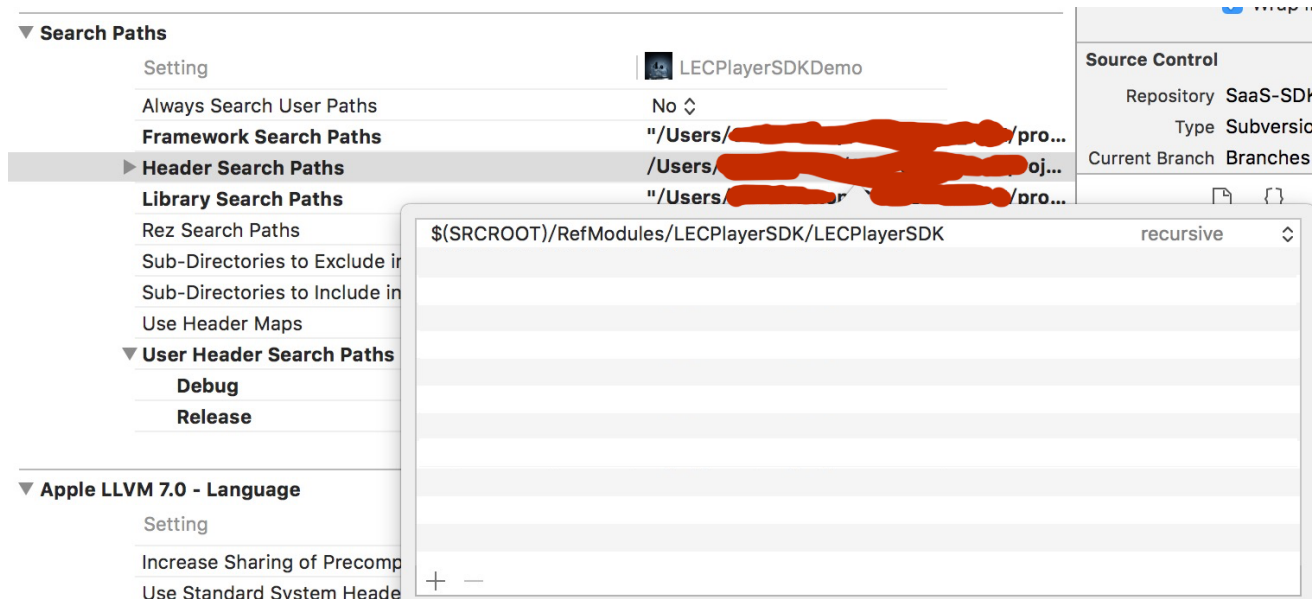


需要记得要确定下，每个需要的.a都加了当前的target，如图



2.2 设置静态库头文件路径以及静态库路径

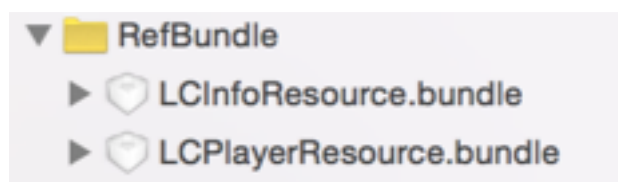
在TARGETS->Build Settings->Header Search Paths中加入SDK中提供的头文件所在路径。如下图:



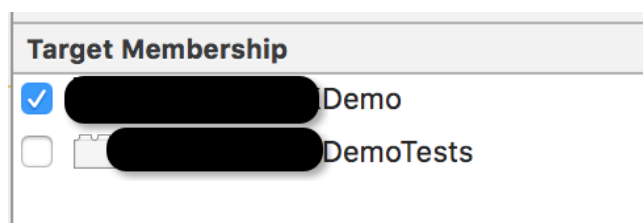
同样设置Lib的路径。

2.3 在工程中引入需要的Bundle文件

在工程目录结构中，点击右键->Add Files to “工程名”。在打开的文件选择器中找到SDK中提供的Bundle文件并加入到工程。如图：

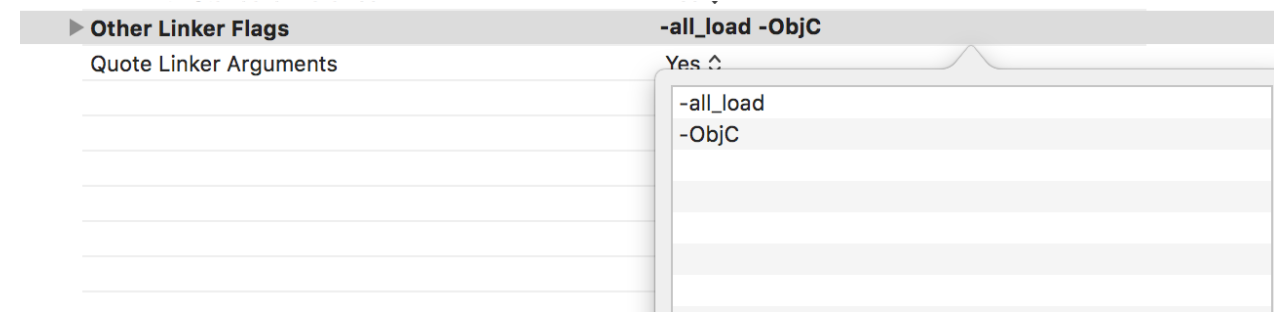


记得要确定下,每个Bundle都设置了当前项目的Target。如图：

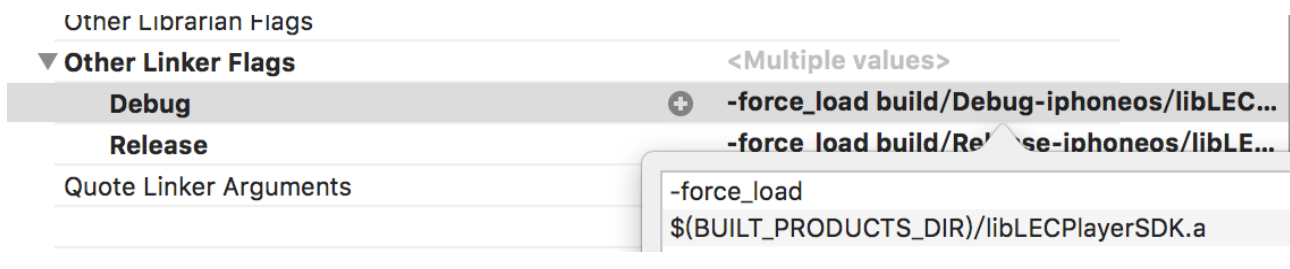


2.4 设置Other Linker Flags

在TARGETS->Build Settings->Other Linker Flags中加 -ObjC和-all_load。如图：



如果编译时存在Duplicated的问题，可以尝试删掉-ObjC和-all_load单独设置-force_load只加载使用到的类别的静态库。如图：




































2.5 设置依赖的Frameworks

在TARGETS->General->Linked Frameworks and Libraries中加 SDK 依赖库，如下图：

▼ Link Binary With Libraries (33 items)

×

Name	Status
 libstdc++.tbd	Required ⇅
 libconv.tbd	Required ⇅
 CoreLocation.framework	Required ⇅
 libz.1.2.5.tbd	Required ⇅
 libxml2.2.tbd	Required ⇅
 AudioToolbox.framework	Required ⇅
 VideoToolbox.framework	Required ⇅
 libstdc++.6.0.9.tbd	Required ⇅
 libz.tbd	Required ⇅
 JavaScriptCore.framework	Required ⇅
 CoreData.framework	Required ⇅
 CoreMedia.framework	Required ⇅
 MediaPlayer.framework	Required ⇅
 AdSupport.framework	Required ⇅
 SystemConfiguration.framework	Required ⇅
 CFNetwork.framework	Required ⇅
 CoreTelephony.framework	Required ⇅
 libswresample.a	Required ⇅
 libavutil.a	Required ⇅
 libcode-sdk.a	Required ⇅
 libcurl.a	Required ⇅
 libcrypto.a	Required ⇅
 libavformat.a	Required ⇅
 zlib.a	Required ⇅
 libavresample.a	Required ⇅
 libglog.a	Required ⇅
 libswscale.a	Required ⇅
 libLECPlayerSDK.a	Required ⇅
 libZipArchive.a	Required ⇅
 libssl.a	Required ⇅
 libReachability.a	Required ⇅
 libavcodec.a	Required ⇅
 libboost.a	Required ⇅

+ -

Drag to reorder frameworks

2.6 关闭BITCODE的开关

由于目前静态库不支持BITCODE,需要在工程设置中关闭BITCODE的开关以通过编译。在 TARGETS->Build Settings->Enable Bitcode,将值设置为NO。

2.7 启动服务

@implementation AppDelegate

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
launchOptions {
    [[LCPlayerService sharedService] startService];
}
```

2.8 模拟器调试存在的问题

前播放SDK在模拟器中运行会存在播放花屏的问题,如图:



在真机运行正常，该问题不影响App功能和申报，我们也正在努力修复中。

2.9 申报AppStore广告IDFA相关问题

由于播放SDK中采集了IDFA标示，所以在集成了播放SDK之后，需按照如下方式设置IDFA，如果申报过程中出现问题，请与我们技术支持联系。

广告标识符

此 App 是否使用广告标识符 (IDFA)?

☒ 是 ☐ 否

[广告标识符 \(IDFA\)](#) 是每台 iOS 设备的唯一 ID，是投放定向广告的唯一方法。用户可以选择在其 iOS 设备上限制广告定位。

如果您的 App 使用广告标识符，请在提交您的代码（包括任何第三方代码）之前进行检查，以确保您的 App 仅出于下面列出的目的使用广告标识符，并尊重“限制广告跟踪”设置。如果您在 App 中加入了第三方代码，则您将对此类代码的行为负责。因此，请务必与您的第三方提供商核实，确认此类代码是否遵循广告标识符和“限制广告跟踪”设置的使用限制。

此 App 使用广告标识符来实现以下目的（选择所有适用项）：

- ☐ 在 App 内投放广告
- ☐ 标明此 App 安装来自先前投放的特定广告
- ☒ 标明此 App 中发生的操作来自先前投放的广告

如果您认为自己还有其他可以接受的广告标识符使用方式，请[联系我们](#)。

iOS 中的“限制广告跟踪”设置

- ☒ 本人，zhouyinan，在此确认，此 App（以及与此 App 交互的任何第三方）使用广告标识符检查功能并尊重用户在 iOS 中的“限制广告跟踪”设置。当用户启用广告标识符后，此 App 不会用于 [iOS 开发人员计划许可协议](#) 中规定的“有限广告目的”之外的任何目的，以任何方式使用广告标识符，以及通过使用广告标识符获取的任何信息。

对于广告标识符的 (IDFA) 的使用，请务必选择正确的答案。如果您的 App 包含 IDFA 而您选择了“否”，此二进制文件将永久被拒绝，您必须提交另一个二进制文件。

官网注册SDK相关说明

1.注册账号

在使用播放SDK之前，先要前往[乐视云官网](http://www.letvcloud.com/)进行注册。整个过程操作很简单，只需一分钟就能搞定。

2.完善配置

账号注册完毕之后，还需要在用户中心进行一些简单的配置，具体请参照相关文档：

<http://help.letvcloud.com/Wiki.jsp?page=CloudDemand>

配置完成后，将标识视频唯一性的播放参数传递给播放SDK即可进行播放，相关参数可以通过官网前端页面获取，也可以通过云直播/云点播API获取，具体请参考相关文档：

<http://help.letvcloud.com/Wiki.jsp?page=DevelopmentGuide>

3.联系我们

如果存在任何疑问，请随时与我们联系。

QQ交流群: 138837683

iOS播放器SDK皮肤版本使用说明

1.版本信息

版本号	版本说明	更新人	日期
4.0	有皮肤版本播放器SDK	郭子龙	2015.12.3

2.概述

此SDK支持视频的直播和点播以及活动直播的功能，目前支持iOS7.0及以上系统，此SDK是包含播放器皮肤，用户集成之后自带UI界面，不支持用户自定义界面。

3.类文件介绍

- LCPlayerControl 播放器皮肤层封装基类,暴露公共的API以及属性
- LCPlayerViewController 实现点播和直播功能
- LCActivityPlayerControl 实现活动直播的功能

4.属性说明

- LCPlayerControl

属性名称	类型	描述
hiddenBackButton	BOOL	是否隐藏播放器自带的返回按钮
delegate	id <LCPlayerViewControlDelegate>	代理
enableGravitySensor	BOOL	是否支持重力感应
hiddenStatusBarWhenFullScreen	BOOL	全屏模式是否隐藏状态栏
autoPlay	BOOL	注册完成后是否自动播放
loadingLogoImage	UIImage	播放器启动加载Logo
hiddenMediaTitle	BOOL	是否隐藏播放器视频标题
playerVolume	float	播放器音量控制,0-1范围,默认1

- LCPlayerViewControl

属性名称	类型	描述
enableVODResumePlay	BOOL	是否启用续播功能,默认是NO
enableDownload	BOOL	是否启用下载功能,默认是YES,在注册播放器前设置

5.API说明

LCPlayerControlDelegate常用代理回调

/*

播放器页面全屏和半屏状态回调

*/

- (void)lcPlayerControl:(LCPlayerControl *)playerControl didChangePlayerFullScreenState:
(BOOL)fullScreen;

/*

点击播放器页面返回按钮回调

*/

- (void)lcPlayerControlDidClickBackBtn:(LCPlayerControl *)playerControl;

/*

返回播放器视频标题以及视频长度和当前播放时间进度信息

注意：仅仅点播会返回当前时间和总视频时间长度

*/

- (void)lcPlayerControl:(LCPlayerControl *)playerControl
mediaTitle:(NSString *)mediaTitle
currentPlayTime:(NSTimeInterval)currentPlayTimestamp
totalTime:(NSTimeInterval)totalTime;

/*

播放器常见错误事件回调

*/

- (void)lcPlayerControl:(LCPlayerControl *)playerControl
playerEvent:(LCPlayerControlEvent)event
error:(NSError *)error;

- LCPlayerControl

方法	方法描述	参数描述	返回值描述
- (void)play;	播放		
- (void)pause;	暂停		
- (void)destroyPlayer;	销毁播放器		
- (BOOL)statusBarHiddenState;	获取状态栏状态		返回BOOL标识状态栏隐藏状态，如果设置hiddenStatusBarWhenFullScreen=YES需要在UIViewController的-(BOOL)prefersStatusBarHidden方法中return此方法;

- LCPlayerViewController

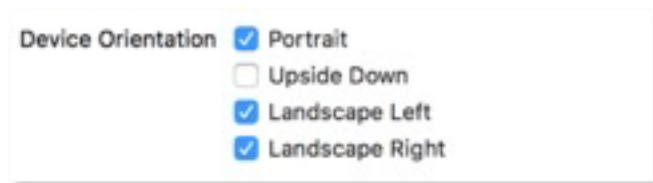
方法	方法描述	参数描述	返回值描述
- (UIView *)createPlayerWithOwner:(id)owner frame:(CGRect)frame;	播放器视图初始化创建	owner: 加载视图的所有者, 传self frame: 播放器视图Frame	返回UIView为播放器UI界面视图View, 上层需要保存并自行加载到Controller的self.view上
- (BOOL)registerLivePlayerWithId:(NSString *)pId mediaType:(LCPlayerMediaType)mediaType;	注册直播播放器	pId:直播ID mediaType:直播类型	返回BOOL标识是否可以执行此操作
- (BOOL)registerVodPlayerWithUU:(NSString *)uu vu:(NSString *)vu;	注册点播播放器	uu:用户标识 vu:点播视频标识	返回BOOL标识是否可以执行此操作
- (BOOL)registerLivePlayerWithStreamId:(NSString *)pId mediaType:(LCPlayerMediaType)mediaType;	播放直播StreamID	pId:直播StreamID mediaType:直播类型	返回BOOL标识是否可以执行此操作

- LCActivityPlayerControl

方法	方法描述	参数描述	返回值描述
- (UIView *)createPlayerWithOwner:(id)owner frame:(CGRect)frame;	播放器视图初始化创建	owner: 加载视图的所有者, 传self frame: 播放器视图Frame	返回UIView为播放器UI界面视图View, 上层需要保存并自行加载到Controller的self.view上
- (BOOL)registerActivityLivePlayerWithId:(NSString *)pId;	注册活动直播播放器	pId:直播ID	返回BOOL标识是否可以执行此操作

6.代码示例

6.1工程配置



6.2主要代码

1.在AppDelegate中初始化注册Player的配置信息数据

```
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [[LCPlayerService sharedService] startService]; //预加载配置文件

    return YES;
}
```

2.在UIViewController中#import所需的类头文件, 实现对应的代理事件, 声明播放器Control属性以及播放View的属性。

```
#import "LCPlayerViewController.h"
@interface LCvodPlayerViewController ()
@property (nonatomic, strong) LCPlayerViewController * control;
@property (nonatomic, strong) UIView * lcPlayerView;
@end
```


3.初始化播放control以及播放视图需要加载到controller的self.view上（需要加载到self.view上）

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _control = [[LCPlayerViewControl alloc] init];
    _control.hiddenStatusBarWhenFullScreen = YES;
    _control.enableGravitySensor = YES;
    _control.hiddenBackButton = NO;
    _control.delegate = self;
    _lcPlayerView = [_control createPlayerWithOwner:self frame:CGRectMake(0, 20,
self.view.frame.size.width, 250)];
    [_control registerVodPlayerWithUU:@"点播所需UU" vu:@"点播所需VU"];
    [self.view addSubview:_lcPlayerView];
}
```

4.重写UIViewController的方法，实现转屏事件(若实现转屏必须重写以下方法，在相应的RootViewController实现)

```
- (BOOL)prefersStatusBarHidden
{
    return [_control statusBarHiddenState];
}

- (BOOL)shouldAutorotate
{
    return NO;
}
```

iOS播放器SDK无皮肤版本接口文档

1.版本信息

版本号	版本说明	更新人	日期
4.0	皮肤版本播放器SDK	侯迪	2015.12.7

2.概述

此SDK支持iOS7.0及以上系统，主要由点播播放器、直播播放器、活动播放器、点播下载模块组成。该SDK适合需要自定义播放器皮肤的客户使用。

3.基类播放器相关接口

3.1概述

LECPlayer为直播、点播以及活动直播Player的基类。目前播控接口、回调状态等接口在LECPlayer中都有定义，与业务相关的功能接口在子类中定义。播放器在使用前需要先进行注册，不再使用的时候要调用unregister进行销毁。

3.2 LECPlayer

• 属性

属性名	类型	读写属性	简介
videoView	UIView	只读	承载视频的view，可通过contentMode设置视频拉伸方式
volume	float	读写	播放器音量
actualVideoWidth	float	只读	在收到LECPlayerPlayEventGetVideoSize回调后，该值表示视频实际宽度
actualVideoHeight	float	只读	在收到LECPlayerPlayEventGetVideoSize回调后，该值表示视频实际高度
streamRatesList	NSArray	只读	NSArray中item类型为LCStreamRateItem;在注册成功后，该值有效；注意在设置选中item时，要判断isEnabled属性

属性名	类型	读写属性	简介
selectedStreamRateItem	LCStreamRateItem	只读	该变量标示当前使用码率，该变量只读，需要切换码率请调用switchSelectStreamRateItem方法实现
errorCode	NSString	只读	当播放失败时，可通过该属性获得错误码
errorDescription	NSString	只读	当播放失败时，可通过该属性获得错误描述
delegate	id	读写	播放器代理
datasource	id	读写	播放器数据源
position	int64_t	只读	播放器当前播放时长
duration	int64_t	只读	视频总时长，对于直播来说是0
playStatus	LECPlayerPlayStatus	只读	播放器状态
contentType	LECPlayerContentType	只读	播放器内容类型，目前只有广告和正片两种类型

• API

方法名	返回值类型	返回值描述	参数以及说明	概述
registerWithURLString	BOOL	返回该方法是否执行成功	//需要注册的播放url (NSString *) urlString	通过播放url注册播放器，注册成功会可执行相关播控方法

方法名	返回值类型	返回值描述	参数以及说明	概述
registerWithURLString	BOOL	返回该方法是否执行成功	//需要注册的播放url (NSString *) urlString //注册结束后会调用该block, 可以通过result判断注册是否成功, 通过player的error信息获得具体失败原因 (^block) completion	通过播放url注册播放器, 注册成功会可执行相关播控方法
unregister	BOOL	返回该方法是否执行成功		销毁播放器前, 或重新注册前需要调用该方法
prepare	BOOL	返回该方法是否执行成功		在play前可先调用此方法开始缓冲, 减少play时的起播时间; 也可以直接play方法开始播放
prepareWithCompletion	BOOL	返回该方法是否执行成功	//注册结束后会调用该block, 可以通过result判断注册是否成功, 通过player的error信息获得具体失败原因 (^block) completion	准备方法, 增加准备操作完成回调
play	BOOL	返回该方法是否执行成功		开始播放接口

方法名	返回值类型	返回值描述	参数以及说明	概述
playWithCompletion	BOOL	返回该方法是否执行成功	//注册结束后会调用该block，可以通过result判断注册是否成功，通过player的error信息获得具体失败原因 (^block) completion	开始播放接口，增加操作完成回调
pause	BOOL	返回该方法是否执行成功		暂停播放接口
resume	BOOL	返回该方法是否执行成功		回复播放接口
stop	BOOL	返回该方法是否执行成功		停止播放接口
stopWithCompletion	BOOL	返回该方法是否执行成功	//注册结束后会调用该block，可以通过result判断注册是否成功，通过player的error信息获得具体失败原因 (^block) completion	停止播放接口，增加操作完成回调
seekToPosition	BOOL	返回该方法是否执行成功	//seek时间，单位为秒 (NSInteger) position	seek到视频相应位置

方法名	返回值类型	返回值描述	参数以及说明	概述
seekToPosition	BOOL	返回该方法是否执行成功	//seek时间, 单位为秒 (NSInteger) position //注册结束后会调用该block, 可以通过result判断注册是否成功, 通过player的error信息获得具体失败原因 (^block) completion	seek到视频相应位置,增加操作完成回调
switchSelectStreamRateItem	BOOL	返回该方法是否执行成功	//目标切换码率 (LCStreamRateItem *) selectStreamRateItem	切换码率接口, selectStreamRateItem可从streamRatesList中选择isEnabled的item
switchSelectStreamRateItem	BOOL	返回该方法是否执行成功	//目标切换码率 (LCStreamRateItem *) selectStreamRateItem //completion代表操作结束, 可继续别的操作, 该返回并不代表操作结果成功, 播放状态需要根据回调决 (^block) completion	切换码率接口, 增加操作完成回调
canDoOperation	BOOL	返回该操作是否可以执行	//操作类型 (LECPlayerPlayOperation) operation	检测某操作是否可以执行

- 回调

回调名称	参数以及说明	概述
contentTypeChanged	//播放内容类型 (LECPlayerContentType) contentType	当播放内容发生变化时，该方法被调用
lecPlayer:playerEvent:	//播放器事件 (LECPlayerPlayEvent) event	播放器事件产生时，会调用该方法
lecPlayer:position:cacheDuration:duration:	//当前播放时间 (int64_t) position //从当前播放时间起的缓冲总时间 (int64_t) cacheDuration //视频总时长 (int64_t) duration	视频播放时长发生变化时的回调
lecPlayer:playFailed:	//出错类型 (LECPlayerEvent) event	播放出错时，会调用该回调

3.3 LECVODPlayer

• 属性

属性名	类型	读写属性	概述
uu	NSString	只读	userId，调用注册方法后该值可获取
vu	NSString	只读	videoUniqueId，调用注册方法后该值可获取
payCheckCode	NSString	只读	调用注册方法后该值可获取
payUserName	NSString	只读	调用注册方法后该值可获取
videoTitle	NSString	只读	视频标题，注册完成后，该值可获取
allowDownload	BOOL	只读	标志视频是否允许下载，注册完成后，该值可获取

属性名	类型	读写属性	概述
resumeFromLastPlayPosition	BOOL	只读	调用注册方法后该值可获取

• 接口

方法名	返回值类型	返回值描述	参数	参数描述
registerWith...	BOOL	返回该方法是否执行成功	uu	userUniqueId
			vu	videoUnique
			payCheckCode	payCheckCode
			payUserName	payUserName
			resumeFromLastPlayPosition	是否从上一次播放恢复，该属性为YES时，播放过程会从上一次播放的结束点开始
			resumeFromLastRateType	是否使用上一次播放的码率
			completion	注册结束后会调用该block，可以通过result判断注册是否成功，通过player的error信息获得具体失败原因

LECPlayerDatasource

/*是否需要自动锁屏*/

- (BOOL) needDisableIdleTimeWhenFinishPlayerWithPlayer:(LECPlayer *) player;

3.4 LECLivingPlayer

方法名	返回值类型	返回值描述	参数	参数描述
registerWith...	BOOL	返回该方法是否执行成功	liveId/streamId	直播id/活动id
			mediaType	直播流类型，有RTMP和HLS两个选择
			isLetvMedia	是否使用乐视媒资，默认为NO
			resumeFromLastRateType	是否使用上一次播放的码率
			completion	注册结束后会调用该block，可以通过result判断注册是否成功，通过player的error信息获得具体失败原因

3.5 LECActivityPlayer

方法名	返回值类型	返回值描述	参数	参数描述
registerWithLiveId...	BOOL	返回该方法是否执行成功	liveId	直播id
			isLetvMedia	是否使用乐视媒资，默认为NO
			completion	注册结束后会调用该block，可以通过result判断注册是否成功，通过player的error信息获得具体失败原因
registerWithStreamId...	BOOL	返回该方法是否执行成功	streamId	活动id
			isLetvMedia	是否使用乐视媒资，默认为NO

方法名	返回值类型	返回值描述	参数	参数描述
			completion	注册结束后会调用该block，可以通过result判断注册是否成功，通过player的error信息获得具体失败原因

• 属性

属性名	类型	读写状态	概述
currentPlayTimestamp	NSInteger	只读	当前播放的时间
serverRealTimestamp	NSInteger	只读	服务器真实时间
streamStartTimestamp	NSInteger	只读	直播流开始的时间
streamEndTimestamp	NSInteger	只读	直播流结束的时间
supportSeekOperation	BOOL	只读	是否支持时移操作
useVODMode	BOOL	只读	当前是否是点播模式
loadingIconUrl	NSString	只读	启动加载的图标url

3.6 LCVODDownloadManager

• 属性

属性名	类型	读写状态	概述
vodItemsList	NSSArray	只读	获取下载队列中的 LECVODDownloadItem
delegate	id< LCVODDownloadManagerDelegate >	weak	点播下载代理
defaultCodeSelectType	LCVODDownloadManagerDefaultCodeSelectType	assign	没有传入下载码率的下载任务会根据该值选择一个默认码率进行下载；默认选择的码率如果已经存在会直接报错，不会引起恢复的操作

• 接口

```

//单例
+ (id) sharedManager;

//创建下载Item
- (LECVODDownloadItem *) createVODDownloadItemWithUu:(NSString *)
uu
    withVu:(NSString *) vu
    userInfo:(NSDictionary *) dict
    withExpectVideoCodeType:(NSString *) videoCodeType;
//如果传入nil, 则会根据defaultCodeSelectType选择默认码率

//创建下载Item
- (LECVODDownloadItem *) createVODDownloadItemWithUu:(NSString *)
uu
    withVu:(NSString *) vu
    userInfo:(NSDictionary *) dict
    withExpectVideoCodeType:(NSString *) videoCodeType
    withPayCheckCode:(NSString *) payCheckCode
    withPayUserName:(NSString *) payUserName;

//开始下载
- (BOOL) startDownloadWithVODItem:(LECVODDownloadItem *)
downloadItem; //如果存在相同码率, 则会返回错误

//暂停下载
- (void) pauseDownloadWithVODItem:(LECVODDownloadItem *)
downloadItem;

//清空已经存在的下载
- (void) cleanDownloadWithVODItem:(LECVODDownloadItem *)
downloadItem;

//根据点播uu和vu获取多种码率下载LECVODDownloadItem
- (NSArray *) vodItemsListWithUu:(NSString *) uu vu:(NSString *) vu;

```

- 代理回调

```
//开始下载事件
- (void) vodDownloadManager:(LCVODDownloadManager *)
downloadManager didBeginDownloadVODDownloadItem:
(LECVODDownloadItem *) vodDownloadItem;
//下载中的下载状态返回
- (void) vodDownloadManager:(LCVODDownloadManager *)
downloadManager downloadingVODDownloadItem:
(LECVODDownloadItem *) vodDownloadItem downloadedBytes:(long
long) downloadedBytes totalBytes:(long long) totalBytes
speed:(float) speed;
//完成下载事件
- (void) vodDownloadManager:(LCVODDownloadManager *)
downloadManager didFinishDownloadVODDownloadItem:
(LECVODDownloadItem *) vodDownloadItem;
//下载出错
- (void) vodDownloadManager:(LCVODDownloadManager *)
downloadManager didFailDownloadVODDownloadItem:
(LECVODDownloadItem *) vodDownloadItem withErrorCode:
(NSString *) errorCode withErrorDesc:(NSString *) errorDesc;
```