1.播放器

播放器的对外核心接口为`PLVMediaPlayerCore`，该接口对外的实现类为`PLVVodMediaPlayer`，区别如下：

- PLVMediaPlayerCore：播放器核心类，提供纯播放器的完整功能，但不包含其他保利威的业务能力
- PLVVodMediaPlayer：是 PLVMediaPlayerCore 的子类，实现了画中画、记忆播放、广告等具有保利威业务能力

注意: 2.1.x版本以下播放器对外核心接口为`PLVPlayerCore`，请以实际版本情况调用。

以下内容将提供PLVVodMediaPlayer的介绍，

2. 初始化

首先，在页面上创建一个`PLVVodMediaPlayer`对象。代码示例如下：

```
#import <PolyvMediaPlayerSDK/PolyvMediaPlayerSDK.h>

@interface UIViewController ()
@property (nonatomic, strong) PLVVodMediaPlayer *player;
@end

@implementation UIViewController()<
PLVMediaPlayerCoreDelegate,
PLVVodMediaPlayerDelegate
>

- (void)viewDidLoad {
  [super viewDidLoad];

  PLVVodMediaPlayer *player = [[PLVVodMediaPlayer alloc] init];
  [player setupDisplaySuperview:self.view];
  self.player = player;
}

@end
```

另外，您也可以通过以下方式初始化并配置URL，默认会自动开始播放配置的URL:

```
/// 播放器初始化
- (instancetype)initWithContentURL:(NSURL *)contentURL;
```

3.设置数据源

通过调用接口`setVideo:`设置数据源

```
- (void)setVideo:(PLVVodMediaVideo *)video
```

在调用该接口后，默认会自动开始播放，您也可以通过播放参数配置来控制不自动起播

另外，您也可以通过以下方式播放相应的URL:

```
/// 设置URL 播放
- (void)setPlayerURL:(NSURL *)playerURL;
```

4.播放参数配置

提供了各种属性控制播放模式：

```
/// 播放模式 视频模式、音频模式
@property (nonatomic, assign) PLVVodMediaPlaybackMode playbackMode;

/// 路由线路，仅对加密视频有效，传入 POVVodVideo 对象中 availableRouteLines 数组的元素
@property (nonatomic, copy) NSString *routeLine;

/// 是否开启记忆播放位置，默认 NO， 开启后从前一播放位置续播
@property (nonatomic, assign) BOOL rememberLastPosition;

/// seek 播放定位类型设置。0 默认，1 精确模式
@property (nonatomic, assign) PLVVodMediaPlaySeekType seekType;

/// 音频播放 seek定位类型设置。0 默认，1 快速模式
@property (nonatomic, assign) PLVVodMediaPlayAudioSeekType audioSeekType;

/// 是否播放片头，默认 NO
@property (nonatomic, assign) BOOL enableTeaser;
```

`PLVMediaPlayerCore`父类中提供了一些常用的播放参数，例如自动播放：

```
@property (nonatomic, assign) BOOL autoPlay;
```

5.播放控制

播放器提供了一系列的播放控制接口，例如：

```
/// 暂停 主播放器 播放
- (void)pause;

/// 静音 主播放器
- (void)mute;

/// 取消静音 主播放器
- (void)cancelMute;

/// 开始 主播放器 播放
- (void)play;

/// 切换清晰度
- (void)setPlayQuality:(PLVVodQuality )quality;

/// seek 播放控制，跳到具体时间点开始播放
- (void)seekToTime:(NSTimeInterval)toTime;
```

更多控制操作可以参考SDK的`PLVVodMediaPlayer`以及它的父类`PLVMediaPlayerCore`

6.回调

使用PLVVodMediaPlayer播放器需要配置以下两种代理：

- PLVVodMediaPlayerDelegate：点播播放器特有回调代理
- PLVMediaPlayerCoreDelegate：基础播放器回调

```
- (PLVVodMediaPlayer *)player{
    if (!_player){
        _player = [[PLVVodMediaPlayer alloc] init];
        _player.coreDelegate = self;
        _player.delegateVodMediaPlayer = self;
        _player.autoPlay = YES;
        _player.rememberLastPosition = YES;
        _player.enablePIPInBackground = YES;
    }
    return _player;
}
```

以下为回调提供的能力，可通过代理方法实现相应的产品逻辑:

```
@protocol PLVMediaPlayerCoreDelegate <NSObject>

@optional
/// 播放器加载前，回调options配置对象
///
/// @note 播放器在加载前，将触发此回调，并附带 PLVOptions，有自定义配置需求时，可对此对象进行参数设置。
///       当集成的是 PLVIJKPlayer 时，请按 PLVIJKFFOptions 来处理 options；
///       当集成的是 IJKMediaFramework 时，请按 IJKFFOptions 来处理 options；
///
/// @param player 播放器对象
/// @param options 播放配置对象
- (PLVOptions *)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerWillLoadWithOptions:(PLVOptions *)options;

/// 播放器 已准备好播放
///
/// @param player 播放器对象
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerIsPreparedToPlay:(BOOL)prepared;

/// 播放器 ’加载状态‘ 发生改变
///
/// @param player 播放器对象
/// @param loadState 播放器加载状态
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerLoadStateDidChange:(PLVPlayerLoadState)loadState;

/// 播放器 ’播放状态‘ 发生改变
///
/// @param player 播放器对象
/// @param playbackState 播放器播放状态
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerPlaybackStateDidChange:
(PLVPlaybackState)playbackState;

/// 播放器 播放结束
///
/// @param player 播放器对象
/// @param finishReson 播放结束原因
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerPlaybackDidFinish:
(PLVPlayerFinishReason)finishReson;

/// 播放器 已销毁
///
/// @param player 播放器对象
/// @param destroy 播放器销毁
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerDidDestroyed:(BOOL)destroy;

/// 主播放器 ‘SEI信息’ 发生改变
///
/// @param player 播放器对象
/// @param timeStamp 附带的时间戳信息
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player playerSeiDidChanged:(long)timeStamp;

/// 首帧渲染回调
- (void)plvMediaPlayerCore:(PLVMediaPlayerCore *)player firstFrameRendered:(BOOL)rendered;


@end
```

```
@protocol PLVVodMediaPlayerDelegate <NSObject>

@optional
/// 点播播放器 发生错误
///
/// @param vodMediaPlayer 点播播放器
/// @param error 错误信息对象 (可能为nil; error.code 可详见 PLVFPlayErrorCodeGenerator.h)
- (void)plvVodMediaPlayer:(PLVVodMediaPlayer *)vodMediaPlayer loadMainPlayerFailureWithError:(NSError *
_Nullable)error;

/// 点播播放器 定时返回当前播放进度
///
/// @param vodMediaPlayer 点播播放器
/// @param playedProgress 已播放进度 (0.0 ~ 1.0)
/// @param playedTimeString 当前播放时间点字符串 (示例 "01:23")
/// @param durationTimeString 总时长字符串 (示例 "01:23")
- (void)plvVodMediaPlayer:(PLVVodMediaPlayer *)vodMediaPlayer
               playedProgress:(CGFloat)playedProgress
             playedTimeString:(NSString *)playedTimeString
           durationTimeString:(NSString *)durationTimeString;


/// 画中画状态回调
/// @param vodMediaPlayer  点播播放器
/// @param pipState 画中画状态
- (void)plvVodMediaPlayer:(PLVVodMediaPlayer *)vodMediaPlayer pictureInPictureChangeState:
(PLVPictureInPictureState )pipState;

/// 画中画开启失败
/// @param vodMediaPlayer  点播播放器
/// @param error 错误信息
- (void)plvVodMediaPlayer:(PLVVodMediaPlayer *)vodMediaPlayer startPictureInPictureWithError:(NSError *)error;

/// 当前网络状态不佳 回调状态
/// @param poorState 网络不佳指示状态
- (void)plvVodMediaPlayer:(PLVVodMediaPlayer *)vodMediaPlayer poorNetworkState:(BOOL)poorState;
```

7.销毁

播放结束后不再使用播放器时，应销毁播放器：

```
[self.player clearPlayer];
```