iOS 点播回放 Core SDK

- BJVideoPlayerCore SDK 支持百家云 **点播、回放** 的 **播放** 及 **下载** 功能，不包含 UI 资源，包含 UI 的 SDK 请参考 iOS 点播 UI SDK 及 iOS 回放 UI SDK。SDK 4.x 版本最低支持 iOS 11.0 及以上系统，旧版的 3.x 支持 iOS 10.0 以上系统。
- git 链接：**https://git2.baijiashilian.com/open-ios/BaijiaYun.git**
- App 下载：**https://itunes.apple.com/app/id1146697098?ls=1&mt=8**
- 旧版SDK：iOS 点播回放 Core SDK 3.0
- 变更记录：changelog

Demo

点播、回放的 UI SDK 是开源的，BJVideoPlayerCore 的集成可参考 BJVideoPlayerUI SDK demo中 Core SDK 的集成方式。

接入SDK

兼容设备

4.x SDK 支持 iOS 11.0 及以上的系统，iPhone、iPad 等设备，集成 2.0 或以上版本的 SDK 要求 Xcode 的版本至少为 9.0，**集成 2.2.0 或以上版本的 SDK 要求 Xcode 的版本至少为 11.0**，由于 SDK 依赖关系复杂、手动配置繁琐，建议使用 CocoaPods 方式引入。

存在 Cocoapods 导入失败的 spec 仓库时，通常可以查找镜像源，替换本地的 Cocoapods 中对应 spec 仓库的 source 的指向来解决。

Podfile 配置

- Podfile 中设置 source

```
source 'https://github.com/CocoaPods/Specs.git'
source 'https://git2.baijiashilian.com/open-ios/specs.git'
```

- `Podfile` 中引入 `BJVideoPlayerCore`：

```
pod 'BaijiaYun/BJVideoPlayerCore', '~>4.0'
```

- 执行 `pod install`

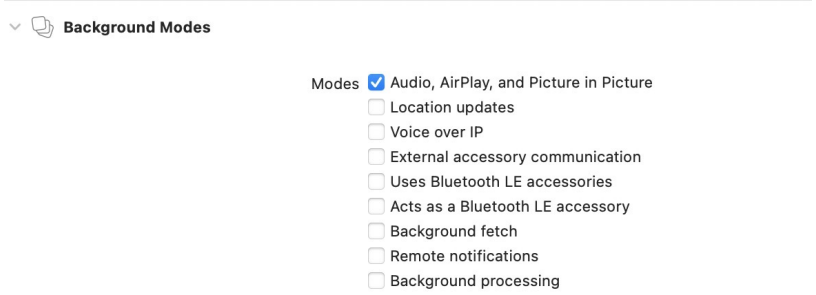
项目全局设置

1. 配置ATS

需要在 info.plist 里面增加 NSAllowsArbitraryLoads = true

2. （可选）开启后台音频

如果有后台播放的需求，则必须开启此项：在 `Project > Target > Signing & Capabilities` 中打开 `Background Modes` 开关，选中 `Audio, Airplay, and Picture in Picture` 。



主要功能

SDK 功能主要分为**点播**、**回放**及**下载**三个部分。

要点说明

1. 设置专属域名前缀

例如专属域名为 “demo123.at.baijiayun.com”，则前缀为 “demo123”，参考专属域名说明。

```
#import <BJVideoPlayerCore/BJVAppConfig.h>

...

/** 设置专属域名前缀 */
[[BJVAppConfig sharedInstance] setPrivateDomainPrefix:loginUser.privateDomainPre
```

2. `BJVTokenManager` 设置 `BJVRequestTokenDelegate`

点播、回放 SDK 的播放和下载功能都需要集成方传入视频 `token` 来向服务器请求数据，数据对应的 `token` 由集成方后台调用百家云 API 获取：点播 token、回放 token，然后通知给 SDK，SDK 使用 `token` 等参数调用 SDK 的相关 API 实现功能。

集成方可配置 `token` 的有效期，当 SDK 发现点播、回放、下载的链接过期或不可用时，会返回 `BJVErrorCode_invalidToken （errorCode = 1010）` 的错误码，同时会通过 `BJVRequestTokenDelegate` 的方法回调，要求给对应的数据传入一个可用的 `token` 。

集成方在 `BJVTokenManager` 的 `tokenDelegate` 的回调方法里提供一个有效的 `token` 返回给 SDK，SDK 就能继续正常工作，这样就实现了自动更新 token 的效果。

对于点播回放的视频播放功能，若集成方没有 token 过期和自动更新的需求，**可以不设置**此 `tokenDelegate` ，SDK 会在发现链接过期或不可用时尝试使用旧的 token 重新请求服务器，请求失败将抛出 `BJVErrorCode_invalidToken （errorCode = 1010）` 错误，APP 执行错误处理的业务逻辑。

对于点播回放的下载功能，则必须要为 `BJVTokenManager` 设置 `tokenDelegate` 。

```
/** 用于在 点播播放/下载 中，集成方需要调用 completion 提供 videoID 对应的 token */
- (void)requestTokenWithVideoID:(NSString *)videoID
                     completion:(void (^)(NSString * _Nullable token, NSError * _Nullable error))completion {
    // 示例：集成方获取点播 token 的方法
```

```
    [self getTokenWithVideoID:videoID
                    completion:^(NSString *token, NSError *error) {
                        completion(token, error);
                    }];
}

/** 用于在 回放播放/下载 中，客户需要调用 completion 提供 classID、sessionID 对应的 token */
- (void)requestTokenWithClassID:(NSString *)classID
                      sessionID:(nullable NSString *)sessionID
                     completion:(void (^)(NSString * _Nullable token, NSError * _Nullable error))completion {
    // 示例：集成方获取回放 token 的方法
    [self getTokenWithClassID:classID
                    sessionID:sessionID
                   completion:^(NSString *token, NSError *error) {
                        completion(token, error);
                    }];
}
```

3. 属性、方法监听方式：Block

demo 及示例代码中大量使用特定的 block 进行属性、方法监听，可参考 直播文档中的对监听 block 的相关介绍。

点播功能说明

建议： 自主集成点播功能之前，可参考 BJVideoPlayerUI SDK 已经开源的代码，该 UI SDK 提供了清晰的点播模块 API 调用方式。

1. 引入头文件

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

2. 设置专属域名

参考全局设置部分，点播、回放、下载通用，项目工程内**只需要设置一次**。

3. （可选）设置 `BJVRequestTokenDelegate`，管理 token

参考要点说明部分，点播、回放、下载通用，项目工程内**只需要设置一次**。

4. 初始化播放器管理类 BJVPlayerManager

```
/** 初始化 manager 实例 */
- (instancetype)initWithPlayerType:(BJVPlayerType)playerType;
```

4.1 指定播放器类型，创建点播 manager 实例

点播播放器分为 `AVPlayer` 与 `IJKPlayer` 两种，参考 `BJVPlayerType` 。

`AVPlayer` 是 iOS 系统的播放器，支持 m3u8 格式的视频，加载比较快，但不支持播放加密视频和flv视频。

`IJKPlayer` 是第三方开源播放器，基于 `FFmpeg` ，支持加密和 flv 格式的视频，不支持m3u8。

```
/**
 播放器类型：AVPlayer(不支持加密视频播放)、IJKPlayer
 #discussion 当播放器初始设置为 AVPlayer 时，但是播放的视频实际为加密格式，
 #discussion SDK 将会切换播放器的类型为 IJKPlayer 来尽量保证可以正常播放视频
 #discussion 因此，在获取播放器的属性时，不能作为一个属性存储，需要实时获取，
 #discussion 或者监听播放器类型的变化，及时获取到新的属性。
 */
@property (nonatomic, readonly) BJVPlayerType playerType;
```

示例代码：

```
// example：创建使用 AVPlayer 的 playerManager
self.playerManager = [[BJVPlayerManager alloc] initWithPlayerType:BJVPlayerType_AVPlayer];
```

4.2 初始化点播

- 初始化在线点播

```
/**
 初始化在线视频
 #param videoID 视频 ID
```

```
 #param token 需要集成方后端调用百家云后端的API获取，传给移动端
 #discussion 调用此方法初始化在线视频时，默认不加密、不使用集成方鉴权
 */
- (void)setupOnlineVideoWithID:(NSString *)videoID
                         token:(NSString *)token;

/**
 初始化在线视频，设置是否加密、集成方鉴权
 #param videoID 视频 ID
 #param token 需要集成方后端调用百家云后端的API获取，传给移动端
 #param encrypted 是否加密，「仅在使用 IJKPlayer 时有效」，参考 initWithPlayerType: 方法
 #param accessKey 集成方鉴权, 视频如果需要请求第三方服务器查看是否有权限, 可设置该参数。鉴权验证请求需要与百家云后台沟通。
 */
- (void)setupOnlineVideoWithID:(NSString *)videoID
                         token:(NSString *)token
                     encrypted:(BOOL)encrypted
                     accessKey:(nullable NSString *)accessKey;
```

- 初始化本地点播

```
/**
 初始化本地视频

 #param downloadItem  本地视频文件类型，通过下载模块获得
 */
```

```
[self.playerManager setupLocalVideoWithDownloadItem:downloadItem];
```

4.3 用户信息设置

```
/**
 第三方用户名和编号
 #discussion 需要在调用 setupOnlineVideoWithID: 之前设置
 */
@property (nonatomic, strong, nullable) NSString *userName;
@property (nonatomic, strong, nullable) NSString *userNumber;
```

4.4 视频播放设置

assign属性的信息需要在调用 `setupOnlineVideoWithID:` 之前设置。

```
/**
 视频的起始播放时间
 #discussion 需要在实例化 playerManager 对象后设置
 #discussion 设置为大于 0 的值时，记忆播放 playTimeRecordEnabled 无效
 #discussion !!!: 使用同一个 playerManager 播放不同视频时，该值不会自动重置，需要根据实际情况重新设置
 */
@property (nonatomic, assign) NSTimeInterval initialPlayTime;

/**
 播放倍速
```

```
 #discussion 有效区间：[0.5, 2.0]
 */
@property (nonatomic, assign) CGFloat rate;

/**
 偏好清晰度列表

 #discussion 优先使用此列表中的清晰度播放在线视频，优先级按数组元素顺序递减
 #discussion preferredDefinitionList 列表元素为清晰度的标识字符串，现有标识符：low（标清），high（高清），superHD（超清），720p，1080p，audio（纯音频），可根据实际情况动态扩展
 #discussion 此设置对播放本地视频无效
 */
@property (nonatomic, strong) NSArray<NSString *> *preferredDefinitionList;

/**
 进入后台时，是否继续播放音频，必须在工程的 background modes 中添加 audio 才会生效
 note: 当设置 backgroundAudioEnabled 为 YES 时，内部会把 pictureInPictureEnabled 置为 NO
 */
@property (nonatomic, assign) BOOL backgroundAudioEnabled;

/** 是否开启记忆播放 */
@property (nonatomic, assign) BOOL playTimeRecordEnabled;

/** 记忆播放时间，当播放器设置正确的播放数据源后才能返回正常数据 */
@property (nonatomic, readonly) NSTimeInterval playTimeRecord;

```

```
/** 清除播放记录 */
- (void)clearPlayTimeRecords;

/**
 开启记忆播放时，视频剩余时间不足 ignorableRemainingTimeInterval 秒时，
 视为播放结束，清除该视频的播放进度纪录，默认 5s，可根据需要自行设置
 */
@property (nonatomic, assign) NSTimeInterval ignorableRemainingTimeInterval;
```

初始化播放代码：

```
BJVPlayerManager *manager = [[BJVPlayerManager alloc] initWithPlayerType:self.videoOptions.playerType];

manager.backgroundAudioEnabled = backgroundAudioEnabled;
manager.remoteControlEnabled = remoteControlEnabled;
manager.remoteControlImage = remoteControlImage;
manager.pictureInPictureEnabled = pictureInPictureEnabled;
manager.preferredDefinitionList = preferredDefinitionList;
manager.playTimeRecordEnabled = playTimeRecordEnabled;
manager.initialPlayTime = initialPlayTime;
manager.userName = userName;
manager.userNumber = userNumber;
manager.rate = rate;

```

```
[self.playerManager setupOnlineVideoWithID:vid
                                     token:token
                                 encrypted:encryptEnabled
                                 accessKey:nil];
// 设置自动开始播放
[self.playerManager play]
```

5. 视频信息

播放器相关设置在 `BJVPlayProtocol.h` 文件中定义。

5.1 视频播放信息

```
/** 当前播放状态 */
@property (nonatomic, readonly) BJVPlayerStatus playStatus;

/** 视频的时长 */
@property (nonatomic, readonly) NSTimeInterval duration;

/** 视频缓存时长 */
@property (nonatomic, readonly) NSTimeInterval cachedDuration;

/** 正片的当前的播放时间，单位秒 */
@property (nonatomic, readonly) NSTimeInterval currentTime;

/** 视频信息 */
@property (nonatomic, readonly, nullable) BJVPlayInfo *playInfo;

/** 当前播放清晰度 */
```

```
@property (nonatomic, readonly, nullable) BJVDefinitionInfo *currDefinitionInfo;

/** 当前使用的字幕信息 */
@property (nonatomic, readonly, nullable) BJVSubtitleInfo *currentSubtitleInfo;

/** 当前时间点的字幕内容 */
@property (nonatomic, readonly, nullable) BJVSubtitle *currentSubtitle;

/** 进入后台或锁屏时，是否支持开启远程控制，仅在使用后台播放时有效，默认关闭 */
@property (nonatomic, assign) BOOL remoteControlEnabled;

/** 设置远程控制状态下的封面 */
@property (nonatomic, strong) UIImage *remoteControlImage;
```

`BJVPlayInfo` 视频信息：

```
@property (nonatomic, readonly) NSString *videoID;
@property (nonatomic, readonly) NSString *classID, *sessionID;
// 是否加密
@property (nonatomic, readonly) BOOL encrypted;

/**
 裁剪后的回放视频的version
 #discussion  -1: 裁剪后的主版本，0: 裁剪前的原始视频.
 */
```

```
@property (nonatomic, readonly) NSInteger clipedVersion;
@property (nonatomic, readonly) NSString *title;
@property (nonatomic, readonly) NSString *coverURL; // 视频封面图片
@property (nonatomic, readonly) NSArray<BJVDefinitionInfo *> *definitionList;
@property (nonatomic, nullable, readonly) NSArray<BJVSubtitleInfo *> *subtitleInfo;
@property (nonatomic, readonly) NSString *format; // 文件格式: mp4, ev1, mp3
@property (nonatomic, readonly) BJRecordType recordType; //录制方式
@property (nonatomic, readonly, nullable) BJVLamp *lamp; // 后台配置的跑马灯
// 是否展示用户列表, 聊天列表
@property (nonatomic, readonly) BOOL isShowUserList, isShowChatList;

// 是否开启禁止录屏功能
@property (nonatomic, readonly) BOOL enablePreventScreenCapture;

- (instancetype)initWithUserVideoInfo:(BJVUserVideo *)userVideoInfo;

// 房间界面布局类型
@property (nonatomic, readonly) BJVLayoutTemplate layoutTemplate;

// 点播三分屏布局模版信息
@property (nonatomic, readonly) BJVTripartiteLayoutInfo *tripartiteLayoutInfo;

// 是否支持评论
```

```
@property (nonatomic, readonly) BOOL enableVideoComment;
```

示例监听代码：

```
// example：监听播放状态变化
bjl_weakify(self);
[self bjl_kvo:BJLMakeProperty(self.playerManager, playerStatus) filter:^BOOL(NSNumber * _Nullable now, NSNumber * _Nullable old, BJLPropertyChange * _Nullable change) {
    return now.integerValue != old.integerValue;
} observer:^BOOL(NSNumber * _Nullable now, NSNumber * _Nullable old, BJLPropertyChange * _Nullable change) {
    bjl_strongify(self);
    BJVPlayerStatus status = now.integerValue;
    // 播放状态变化的自定义处理
    [self playerStatusDidChangeTo:status];
    return YES;
}];

// example：播放时间更新
    // 播放时间
    [self bjl_kvoMerge:@[BJLMakeProperty(self.playerManag
 duration),
        BJLMakeProperty(self.playerManager, currentTime)]
              observer:^(id _Nullable value, id _Nullable oldValue, BJLPropertyChange *_Nullable change) {
                  bjl_strongify(self);
                  bjl_dispatch_async_main_queue(^{

```

```
                //通知主线程刷新 当前时间展示
                [self.mediaControlView updateProgressWithCurrentTime:self.playerManage
cacheDuration:self.playerManager.cachedDuration totalDuration:self.playerManager.duration];
            });
        }];
```

5.2 视频画面

```
/** 播放视图 */
@property (nonatomic, readonly, nullable) UIView *playerView;

/** 播放视图显示模式 */
@property (nonatomic) BJVPlayerViewScalingMode scalingMode;
```

示例代码：

```
UIView *playerView = self.playerManager.playerView;
[self.view addSubview:playerView];
//TODO设置布局约束
[playerView bjl_makeConstraints:^(BJLConstraintMaker *make) {
    make.edges.equalTo(self.view);
}]
```

```
bjl_weakify(self);
```

```
[self bjl_kvo:BJLMakeProperty(self.playerManager,
currDefinitionInfo)
    observer:^BOOL(id _Nullable now, id
_Nullable old, BJLPropertyChange * _Nullable
change) {
        bjl_strongify(self);
        // 根据当前清晰度获取视频宽高比
        BJVDefinitionInfo *currDefinitionInfo =
self.playerManager.currDefinitionInfo;
        CGFloat width = currDefinitionInfo.width;
        CGFloat height = currDefinitionInfo.height;
        CGFloat videoRatio = height > 0.0 ? width /
height : 0.0;

        // 更新播放视图布局
        [self.playerManager.playerView
bjl_remakeConstraints:^(BJLConstraintMaker
*make) {
            if (videoRatio > 0) {

make.edges.equalTo(superview).priorityHigh();
                make.center.equalTo(superview);

make.top.left.greaterThanOrEqualTo(superview);

make.bottom.right.lessThanOrEqualTo(superview);

make.width.equalTo(playerView.bjl_height).multiplie
            }
            else {
                make.edges.equalTo(superview);
            }
        }];

        return YES;
    }];
```

5.3 视频控制

```
- (void)play;
- (void)pause;
- (void)seek:(NSTimeInterval)time;
/**
 切换清晰度
 #param index 目标清晰度 在 BJVPlayInfo 的 definitionList 数组中的序号
 */
- (void)changeDefinitionWithIndex:(NSUInteger)index;
-
/** 切换字幕文件
 #param index 目标字幕 在 BJVPlayInfo 的 subtitleInfo 数组中的序号
 */
- (void)changeSubtitleWithIndex:(NSUInteger)index;

/**
 清理播放器，重置状态、清空数据
 */
- (void)reset;
-
 /**
 销毁播放器
 #discussion 调用后彻底销毁播放器，不可恢复
 #discussion 可在退出播放界面前调用
 */
- (void)destroy;
```

5.4 视频播放中的错误监听

```
/*
 视频播放出错
 @param playInfo 出错视频信息
 @param error 错误信息
 */
- (BJLObservable)video:(BJVPlayInfo *)playInfo playFailedWithError:(NSError *)error;

// example：监听视频播放出错
[self bjl_observe:BJLMakeMethod(self.playerManager, video:playFailedWithError:) observer:^BOOL(BJVPlayInfo *playInfo, NSError *error) {
    bjl_strongify(self);
    // 自定义播放错误处理方法
    [self video:playInfo playFailedWithError:error];
    return YES;
}];
```

- 错误码信息

```
typedef NS_ENUM(NSInteger, BJVErrorCode) {
    BJVErrorCode_unknown            = -999,    // 未知错误
    BJVErrorCode_requestFailed      = 1000,    // 网络请求失败
    BJVErrorCode_invalidToken       = 1010,    // token 参数错误
    BJVErrorCode_invalidAccessKey   = 1020,    // 鉴权验证失败
    BJVErrorCode_invalidPlayInfo    = 1030,    // 播放信息解析错误
    BJVErrorCode_invalidVideoURL    = 1040,    // 在线播放，视频地址失效
```

```
    BJVErrorCode_fileLost          = 1050,    // 本地播放，播放文件不存在
    BJVErrorCode_playFailed        = 1060     // 视频播放失败
};
```

5.5 画中画

画中画功能仅支持 `BJVPlayerType_AVPlayer` 类型的播放器。

```
/**
 开启画中画
 note: 当前的播放器类型为 BJVPlayerType_AVPlayer 且 backgroundAudioEnabled == NO 时, 设置 pictureInPictureEnabled 才有效
 note: 当设置 backgroundAudioEnabled 为 YES 时, 内部会把 pictureInPictureEnabled 置为 NO
 */
@property (nonatomic, assign) BOOL pictureInPictureEnabled;

/** 画中画视图控制器, 可能为空 */
@property (nonatomic, strong, nullable, readonly) AVPictureInPictureController *pipViewContrller;

/** 启动画中画, 仅 AVPlayer 有效  */
- (void)startPictureInPicture;

/** 关闭画中画, 仅 AVPlayer 有效 */
- (void)stopPictureInPicture;
```

如果UI上增加了画中画的按钮， 仅需在点击启动时调用：

```
[self.playerManager startPictureInPicture];
```

停止画中画：

```
[self.playerManager stopPictureInPicture];
```

5.6 点播打点

首先需要在百家云后台给对应视频打点，然后 `playerManager` 初始化成功之后调用打点请求，再根据返回数据匹配需要的UI。

```

/**
 点播：请求视频打点数据

 #param fids        视频fid
 #param completion  请求完成回调
 #return            请求的 task
 */
- (NSURLSessionTask *)getVideoCatalogueInfoWithVideoVID:(NSString *)vid
                                                  token:(NSString *)token
                                             completion:(nullable void (^)(NSURLSessionDataTask *_Nullable task,
                                                                           NSArray<BJVVideoCatalogueItem *> *_Nullable list,
                                                                           NSError *_Nullable error))completion;
```

5.7 点播评论

点播是否支持评论由 `playinfo.enableVideoComment` 控制。
整体功能由 `BJVCommentManager` 管理:

```
@interface BJVCommentManager: NSObject

@property (nonatomic, readonly, nullable) NSString *userToken;

+ (instancetype)sharedInstance;

- (void)updateUserToken:(NSString *)userToken;
```

- 获取 `token`， 参考：获取点播评论token接口
- 设置 `userToken`

```
[[BJVCommentManager sharedInstance] updateUserToken:token];
```

- 获取评论数据

```
- (void)requestCommentWithVideoID:(NSString *)vid page:(NSInteger)page pageSize:(NSInteger)pageSize completedBlock:(void(^)(NSArray<BJVCommentItem *> * _Nullable data, NSError  * _Nullable error))completedBlock;

- (void)leaveCommentWithVideoID:(NSString *)vid comment:(NSString *)comment  avatar:(NSString *)avatar completedBlock:(void(^)(NSError  * _Nullable error))completedBlock;

- (void)likeCommentWithVideoID:(NSString *)vid commentID:(NSString *)commentID  like:
```

```
(BOOL)like completedBlock:(void(^)(NSError * _Nullable error))completedBlock;

- (void)deleteCommentWithVideoID:(NSString *)vid commentID:(NSString *)commentID completedBlock:(void(^)(NSError * _Nullable error))completedBlock;
```

在自定义UI界面通过调用 `requestCommentWithVideoID` 可以获取评论列表：

```
[[BJVCommentManager sharedInstance] requestCommentWithVideoID:videoID page:page pageSize:pageSize completedBlock:block];
```

6. 播放器清理、销毁

- 清理

调用点播初始化方法 setupVideo 时将自动调用，其它情况可根据实际需要调用。

```
// 清理播放器，重置状态、清空数据
[self.playerManager reset];
```

- 销毁

```
/**
 销毁播放器
 #discussion 调用后彻底销毁播放器，不可恢复
 #discussion 可在退出播放界面前调用
 */
[self.playerManager destroy];
```

7. 部分功能点说明

7.1 记忆播放

- 设置 `BJPlayerManager` 的 `playTimeRecordEnabled` 属性开关记忆播放。开启后记录 vid 对应的播放进度，每 5 秒【更新】一次播放进度，seek 时、停止播放时立即【更新】一次；
- 如果 视频总时长 减去 当前进度 小于 N 秒，则视为播放结束，只移除记录、不添加，N 默认 5 秒，可自定义，参考 `BJPlayerManager` 的 `ignorableRemainingTimeInterval` 属性；
- 调用 `clearPlayTimeRecords` 方法清除播放纪录；
- 最多存 100 条，超过 100 条时顶掉最早的；
- 设置视频初始播放时间为大于 0 的值时，记忆播放无效。

7.2 纯音频播放

- 点播需要在后台配置转码纯音频才有纯音频
- 播在线播放的默认清晰度可以在后台配置清晰度列表，SDK 根据配置的清晰度优先级列表来匹配第一个清晰度，如果后台配置了纯音频最高优先级，就可以进入回放和点播的时候直接播放纯音频，否则就通过改变清晰度的方式播放纯音频
- 视频清晰度 `PMVideoDefinitionType` 增加纯音频枚举
- 纯音频播放

```
NSArray *definitionList = [self.playerManager.playInfo.definitionList copy];
for (NSInteger index = 0; index < definitionList.count; index ++) {
    BJVDefinitionInfo *definitionInfo = [[definitionList bjl_objectAtIndex:index] bjl_as:[BJVDefinitionInfo class]];
    if ([definitionInfo.definitionKey isEqualToString:@"audio"]) {
```

```
        [self.playerManager
changeDefinitionWithIndex:index];
    }
}
```

7.3 视频清晰度标识

`BJVDefinitionInfo` 的 `definitionKey` 表示清晰度标识符，与 `definitionName` 一一对应。可以随视频的实际清晰度动态扩展。

definitionKey	definitionName
low	标清
high	高清
superHD	超清
720p	720p
1080p	1080p
audio	纯音频
...	...

7.4 视频切换功能

如果想要实现切换上一个或者下一个视频的功能，可以在UI上增加按钮并增加回调，调用以下代码：

```
[self.playerManager reset];
[self.playerManager setupOnlineVideoWithID:vid
                                     token:token
                                 encrypted:encryptEnabled
                                 accessKey:nil];
```

回放功能集成

建议：自主集成回放功能之前，可参考 BJPlaybackUI SDK 已经开源的代码，该 UI SDK 提供了清晰的回放模块 API 调用方式。

1. 引入头文件

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

2. 设置专属域名前缀

参考全局设置部分，点播、回放、下载通用，项目工程内**只需要设置一次**。

3. （可选）设置 `BJVRequestTokenDelegate`，管理 token

参考全局设置部分，点播、回放、下载通用，项目工程内**只需要设置一次**。

4. 创建、进入回放房间

创建、进入回放房间的整体流程:

- 按需要设置专属域名前缀
- 定义一个 `BJVRoom` 的属性 `room`，用于管理回放房间
- 使用回放相关信息将 `room` 属性实例化
- 为回放的进入，退出，聊天消息更新等事件添加监听
- 调用 `room` 的 `enter` 方法进入房间，监听到进入房间成功之后，即可开始观看回放

4.1 定义回放房间属性

```
@property (nonatomic) BJVRoom *room;
```

- 以下为 `BJVRoom` 详细属性说明

```
/** 是否是本地视频 */
@property (nonatomic, readonly) BOOL isLocalVideo;

/** 是否是小班课 */
@property (nonatomic, readonly) BOOL isInteractiveClass;

/** 是否是合并回放房间 */
@property (nonatomic, readonly) BOOL isMixPlaybackRoom;

/** 是否正在加载 */
@property (nonatomic, readonly) BOOL loading;

/** 回放房间信息 */
@property (nonatomic, readonly, nullable) BJVRoomVM *roomVM;

/** 聊天消息 */
@property (nonatomic, readonly, nullable) BJVMessageVM *messageVM;

/** 在线用户 */
@property (nonatomic, readonly, nullable) BJVOnlineUserVM *onlineUsersVM;

/** 播放控制器 */
@property (nonatomic, readonly, nullable) id<BJVPlayProtocol> playerManager;
```

```
24.
25.  /** 文档显示，尺寸、位置随意设定，需要设置为整数的 pt 值 */
26.  @property (nonatomic, readonly, nullable) UIViewController<BJLSlideshowUI> *slideshowViewController;
27.
28.  /**
29.   裁剪后的回放视频的version, 需要在调用enter之前设置
30.   #discussion  -1: 裁剪后的主版本，0: 裁剪前的原始视频.
31.   #discussion 默认 clipedVersion 为 -1
32.   */
33.  @property (nonatomic) NSInteger clipedVersion;
34.
35.  /**
36.   分组, 设置后, 返回对应分组的聊天信息, 需要在调用enter之前设置
37.   #discussion 默认 group 为 -1, 即返回所有分组的聊天信息
38.   #discussion group, 0: 大班主讲和助教, 当 group > 0 时, 也会返回大班主讲和助教的信息
39.   */
40.  @property (nonatomic) NSInteger groupID;
41.
42.  /** 是否禁用动态文档，默认启用动态文档，为 NO，但是在无网络状态下进入离线回放时，将会自动切换成静态文档来保证能够正常观看 */
43.  @property (nonatomic) BOOL disablePPTAnimation;
44.
45.  /** 是否存在 h5 文档（指除了 ppt, pptx 类型的动画文档
```

```
46.  如果存在，不能设置成禁用动画文档，并且建议联网观看，否则无网络的离线回放将无法加载 */
47.  @property (nonatomic, readonly) BOOL existWebDocument;
48.
49.  /** 回放房间的相关信息 */
50.  @property (nonatomic, readonly) BJVPlaybackInfo *playbackInfo;
51.
52.  /** 本地房间的相关信息 */
53.  @property (nonatomic, readonly) BJVDownloadItem *downloadItem;
```

4.2 创建回放房间

- 创建在线回放房间

```
1. /**
2.  创建在线回放房间，配置项采用默认值
3.
4.  #discussion 具体参考 `onlinePlaybackRoomWithClassID:sessionID:token:e 方法
5.  #discussion 视频不加密
6.  #discussion 使用 AVPlayer
7.
8.  @param classID     课程 ID
9.  @param sessionID   课节 ID
10.  @param token      需要集成方后端调用百家云后端的API获取，传给移动端
11.  @return          回放房间实例
12.  */
13. self.room = [BJVRoom onlinePlaybackRoomWithClassID:classID
14.                                          sessionID:sessionID
```

```
                                token:token];
```

```
/**
 创建在线回放房间，支持多项配置

 #param classID     课程 ID
 #param sessionID   课节 ID
 #param token       需要集成方后端调用百家云后端的API获取，传给移动端
 #param encrypted   是否加密，「仅在使用IJKPlayer 时有效」，playerType 参数传BJVPlayerType_IJKPlayer
 #param accessKey   集成方鉴权，回放如果需要请求第三方服务器查看是否有权限，可设置该参数。鉴权验证请求需要与百家云后台沟通。
 #param playerType  播放器类型：AVPlayer、IJKPlayer
 #return            回放房间实例
 */
self.room = [BJVRoom onlinePlaybackRoomWithClassID:classID
                                         sessionID:sessionID
                                             token:token

encrypted:encryptEnabled
                                         accessKey:accessKey
                                        playerType:playerType];
```

- 创建本地回放房间

创建本地回放的 `downloadItem` 参数通过下载功能获得

```
/**
 创建本地回放房间

```

```
 @param downloadItem    本地回放文件类型，通过下载模块获得
 @param playerType      播放器类型：AVPlayer、IJKPlayer
 @return                回放房间实例
 */
self.room = [BJVRoom localPlaybackRoomWithDownloadItem:downloadIte

playerType:BJVPlayerType_AVPlayer];
```

`init` 方法初始化示例代码：

```
    BJVRoom *room = [BJVRoom onlinePlaybackRoomWithClassID:classID sessionID:sessionID token:token];
    // 设置groupID 可以实现聊天分组的功能，默认-1，返回所有聊天信息
    room.groupID = options.groupID;

// 设置playerManager的属性
    BJVPlayerManager *playerManager = bjl_as(self.room.playerManager, BJVPlayerManager);
    playerManager.userName = userName;
    playerManager.userNumber = userNumber;
    playerManager.remoteControlEnabled = remoteControlEnabled;
    playerManager.remoteControlImage = remoteControlImage;
    playerManager.backgroundAudioEnabled = backgroundAudioEnabled;
    playerManager.preferredDefinitionList = preferredDefinitionList;
```

```
    playerManager.playTimeRecordEnabled = playTimeRecordEnabled;
    playerManager.initialPlayTime = initialPlayTime;
    ...
```

`viewDidLoad` 方法示例：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // UI布局
    // 关于room的一些状态监听
    // ppt文档属性设置

    // 以上准备完成之后调用 enter
   [self.room enter];
}
```

4.3 回放房间状态监听

在创建回放房间之后、进入回放房间（调用 `BJVRoom` 的 `enter` 方法）之前设置。

可参考 BJPlaybackUI SDK 中 `BJPRoomViewController+observer.m` 文件的代码。

- 监听进入房间

```
/**
 进入房间
 #param error 错误信息，成功进入时返回空值，错误码参考 BJVErrorCode
 */
- (BJLObservable)roomDidEnterWithError:(nullable NSError *)error;
```

```
// 示例代码
bjl_weakify(self);
[self bjl_observe:BJLMakeMethod(self.room, roomDidEnterWithError:)
        observer:^BOOL(BJLError *error) {
            // 此处建议在监听到进入房间成功之后，再对 BJVRoom 的 view models进行监听
            bjl_strongify(self);
            // example: 添加对 BJVRoom 的各个 VM 的监听
            [self addObserversForPlaybackViewModels];
            // example: 添加对 BJVRoom 的播放器的相关监听
            [self addObserversForVideoPlayer];
            return YES;
        }];
```

- 监听退出房间

```
/**
 退出房间
 #param error 错误信息，成功退出时返回空值，错误码参考 BJVErrorCode
 */
- (BJLObservable)roomDidExitWithError:(NSError *)error;

// 示例代码
bjl_weakify(self);
[self bjl_observe:BJLMakeMethod(self.room, roomDidExitWithError:)
        observer:^BOOL(BJLError *error) {
            bjl_strongify(self);
```

```
            // 退出房间之后的自定义操作
            [self playbackEndWithError:error];
            return YES;
        }];
```

- 监听加载状态

```
/**
 是否正在加载
 */
@property (nonatomic, readonly) BOOL loading;

// 示例代码
bjl_weakify(self);
[self bjl_kvo:BJLMakeProperty(self.room, loading)
    observer:^BOOL(NSNumber * _Nullable value, NSNumber * _Nullable oldValue, BJLPropertyChange * _Nullable change) {
        bjl_strongify(self);
        // example: 根据 loading 值显示/隐藏加载动画
        if (value.boolValue) {
            [self showLoading];
        }
        else {
            [self stopLoading];
        }
        return YES;
    }];
```

- 错误码信息

```
typedef NS_ENUM(NSInteger, BJVErrorCode) {
    BJVErrorCode_unknown            = -999,     // 未知错误
```

```
    BJVErrorCode_requestFailed      = 1000,    // 网络请求失败
    BJVErrorCode_invalidToken       = 1010,    // token 参数错误
    BJVErrorCode_invalidAccessKey   = 1020,    // 鉴权验证失败
    BJVErrorCode_invalidPlayInfo    = 1030,    // 播放信息解析错误
    BJVErrorCode_invalidVideoURL    = 1040,    // 在线播放，视频地址失效
    BJVErrorCode_fileLost           = 1050,    // 本地播放，播放文件不存在
    BJVErrorCode_playFailed         = 1060     // 视频播放失败
};
```

4.4 合并回放房间

- 创建在线合并回放房间：

```
/**
 创建在线合并回放房间，配置项采用默认值

 #discussion 具体参考 `onlinePlaybackRoomWithClassID:sessionID:token:e 方法
 #discussion 视频不加密

 #param mixID       课程 ID
 #param mixToken    需要集成方后端调用百家云后端的API获取，传给移动端
 #param playerType  播放器类型：AVPlayer、IJKPlayer
 #return           回放房间实例
 */
```

```
self.room = [BJVRoom BJVRoom onlineMixPlaybackRoomWithMixID:mixID

mixToken:mixToken

playerType:playerType];
```

- 合并回放房间的监听

```
/**
 合并回放房间即将播放新片段

 #discussion 在获取到必要元数据后，会更新room内的playback info数据再调用该回调
 可监听此方法来更新和特定playback info相关的UI组件
 */
- (BJLObservable)mixPlaybackRoomWillLoadSlice:(BJVPlaybackInfo*)slice;

/**
 合并回放房间的新片段加载结束

 #discussion 如果加载失败，可以调用 `reload` 方法来重试
 可监听此方法来更新和特定playback info相关的UI组件

 #param error 为nil表示加载成功。否则 error 表示具体原因, 错误码参考 BJVErrorCode
 */
- (BJLObservable)mixPlaybackRoomSlice:(BJVPlaybackInfo*)slice didLoadWithError:(nullable NSError *)error;
```

5. 进入、退出、刷新回放房间

```
/**  进入房间 */
- (void)enter;

/** 退出房间 */
- (void)exit;

/** 重新加载房间内容
 #discussion 用于回放过程中出错后重试。
 */
- (void)reload;
```

6. 回放视频播放管理

与点播的 `BJVPlayerManager` 类似，回放房间 `BJVRoom` 的 `playerManager` 遵循 `BJVPlayProtocol` 协议，因此回放的视频播放管理可以参考点播的播放设置、添加播放视图、视频播放控制、视频播放信息、视频播放中的错误监听 等部分，使用 `self.room.playerManager` 调用相关方法。

6.1 回放视频信息

具体的回放视频信息，可参考 `room.playbackInfo` 。

`BJVPlaybackInfo` 继承于 `BJVPlayInfo` ，
在 `BJVPlayInfo` 基础上增加了回放的信息：

```
@interface BJVPlaybackInfo: BJVPlayInfo
<BJLYYModel>

// all.json信令文件 URL
@property (nonatomic, readonly) NSString
*signalFileURL;

```

```objc
// 回放是否要支持答题器和小测
@property (nonatomic, readonly) BOOL enableQuizAndAnswer;

// 是否显示问答
@property (nonatomic, readonly) BOOL enableQuestion;

// 学生视频列表
@property (nonatomic, readonly) NSArray<BJVUserVideo *> *userVideoList;

// 回放的班型
@property (nonatomic, readonly) BJVRoomType roomType;

// 小班课 1v1 是不是信令录制
@property (nonatomic, readonly) BOOL isInteractiveClass1v1SignalingRecord;

// 小班课 1v1 黑板数量
@property (nonatomic, readonly) NSInteger interactiveClass1v1BlackboardPages;

// 白板背景图片 url
@property (nonatomic, readonly) NSString *whiteboardURL;

// 是否开启隐藏学生消息中的手机号功能
@property (nonatomic, readonly) BOOL enableHideStudentPhoneNumber;

// 三分屏主屏内容
@property (nonatomic, readonly) BJVMajorPlayType majorPlayType;
```

6.2 视频窗口占位图更新

当前回放如果播放时未推送视频流，可以选择监听主讲人变化，然后选择展示或者隐藏一张占位图：

```
    // 主讲
    [self bjl_kvo:BJLMakeProperty(self.room.onlineUsersVM, currentPresenter)
        observer:^BOOL(id _Nullable now, id _Nullable old, BJLPropertyChange *_Nullable change) {
            bjl_strongify(self);
            BJVMediaUser *presenter = self.room.onlineUsersVM.currentPresenter;
            // 更新视频占位图 显示/隐藏 状态
            [self updateAudioOnlyImageViewHidden];
            return YES;
        }];
```

1. 当前播放纯音频流，可考虑隐藏播放画面；
2. 当前视频为合流录制，会一直有播放画面；
3. 播放视频时，可参考主讲人的音频是否打开，当前是否在推媒体流，当前是否在屏幕共享，当前小班课是否为信令录制。

```
- (void)updateAudioOnlyImageViewHidden {
    if (self.room.playerManager.currDefinitionInfo.isAudio {
        // 播放纯音频时，显示占位图
        self.audioOnlyImageView.hidden = NO;
        return;
    }

```

```
    if (self.room.playerManager.playInfo.recordType == BJRecordType_Mixed
        || self.room.playerManager.playInfo.recordType == BJRecordType_CompositeVideo) {
        // 合流视频 和 录制视频 一直不显示占位图
        self.audioOnlyImageView.hidden = YES;
        return;
    }

    // 播放视频时，根据老师是否打开摄像头来显示占位图
    self.audioOnlyImageView.hidden = self.room.onlineUsersVM.currentPresenter.videoOn || self.room.roomVM.isMediaPlaying || self.room.roomVM.isDesktopSharing || self.room.playbackInfo.isInteractiveClass1v1Signali
}
```

注意：调用 `self.room.playerManager` 的 `destroy` 方法将导致回放房间不可用，且不可恢复。

7. 回放文档管理

回放房间文档的白板、PPT、画笔的展示都来自 `BJVRoom` 的 `slideshowViewController` 。用户根据需要将 `slideshowViewController.view` 添加到需要展示的 父 view 上即可。**文档的尺寸和位置必须设置为整数的 pt 值。**

7.1 文档设置

- 交互设置：为保证文档内容与视频同步，需要设置回放的文档不支持交互

```
self.room.slideshowViewController.view.userInterac
= NO;
```

- 显示设置：通过 `BJLSlideshowUI` 中的相关属性设置，回放的文档均为静态文档

```
/** 静态文档显示模式
 只支持 BJLContentMode_scaleAspectFit、BJLContentMode_scaleAspectFill
 只对静态文档生效，参考 `BJLRoom` 的 `disablePPTAnimation`
 */
@property (nonatomic) BJLContentMode contentMode;

/** 静态文档尺寸
 加载文档图片时对图片做等比缩放，长边小于/等于 `imageSize`，放大时加载 1.5 倍尺寸的图片
 单位为像素，默认初始加载 720、放大加载 1080，取值在 `BJLAliIMGMinSize` 到 `BJLAliIMGMaxSize` 之间 (1 ~ 4096)
 不建议进回放房间成功后设置此参数，因为会导致已经加载过的图片缓存失效
 只对静态文档生效，参考 `BJLRoom` 的 `disablePPTAnimation`
 */
@property (nonatomic) NSInteger imageSize;

/** 静态文档占位图
 只对静态文档生效，参考 `BJLRoom` 的 `disablePPTAnimation`
 */
@property (nonatomic) UIImage *placeholderImage;
```

- 文档动效设置（仅支持在线回放）
  - `BJVRoom` 的 `disablePPTAnimation` 属性表示是否禁止文档动效，在线回放默认禁止，可设置为 `NO` 以开启动效；
  - 本地回放不支持文档动效。

7.2 文档显示

```
// example：添加文档视图到当前的视图控制器
UIViewController *pptViewController = self.room.slideshowViewController;

[self addChildViewController:pptViewController];
[self.view addSubview:pptViewController.view];
[pptViewController didMoveToParentViewController:self];
```

7.3 文档大纲

大纲数据参考文档大纲属性。

8. 回放房间信息

回放房间信息由 `BJVRoomVM` 管理，在 `BJVRoom` 的实例创建成功之后，可以对 `self.room.roomVM` 添加监听。

```
/** 当前是否正在播放媒体文件 */
@property (nonatomic, readonly) BOOL isMediaPlaying;

/** 当前是否正在共享屏幕 */
@property (nonatomic, readonly) BOOL isDesktopSharing;

```

```
/** 用户是否在录制屏幕，iOS11以上生效，iOS11以下一直是NO */
@property(nonatomic, readonly) BOOL screenCaptured;
```

8.1 公告

```
/** 回放房间公告 */
@property (nonatomic, readonly) BJVNotice *notice;
```

8.2 跑马灯

```
/** 回放房间跑马灯 */
@property (nonatomic, readonly, nullable) BJVLamp *lamp;
```

8.3 文档大纲

```
/** 当前回放房间文档的大纲信息 */
@property (nonatomic, readonly, nullable) NSArray <BJPDocumentCatalogueModel *> *documentCatalogueList;

/** 当前回放房间内实时的最新一条翻页数据 */
@property (nonatomic, readonly, nullable) BJPPageChangeModel *lastPageChangeSignal;
```

8.4 定制广播回调

```
/**
 收到定制广播信令
 #param key     信令类型
```

```
 #param value  信令内容，类型可能是字符串或者字典等 JSON 数据类型
 #param isCache 是否为缓存
 */
-(BJLObservable)didReceiveCustomizedBroadcast:(NSString *)key value:(nullable id)value isCache:(BOOL)isCache;
```

8.5 测验

```
/**
 用户: 收到新题目
 #param survey 题目
 */
- (BJLObservable)didReceiveSurvey:(BJVSurvey *)survey;

/** 用户: 收到答题结束 */
- (BJLObservable)didFinishSurvey;
```

8.6 答题器

```
/** 收到答题器开始信息 */
-(BJLObservable)didReceiveQuestionAnswerSheet:(BJVAnswerSheet *)answerSheet;

/**
 收到答题结束信息
 #param endTime 答题结束时间戳
 */
-(BJLObservable)didReceiveEndQuestionAnswerWith
```

```
(NSTimeInterval)endTime;

/**
 收到答题器撤销信息
 #param endTime 答题结束时间戳
 */
-
(BJLObservable)didReceiveRevokeQuestionAnswerW
(NSTimeInterval)endTime;
```

8.7 问答

```
/** 清空问答 */
- (BJLObservable)didResetQuestion;

/**
 问答发布成功
 #discussion 只有在发布没有回答的问答时才会回调此方法
 #param question 问答全部内容
 */
- (BJLObservable)didPublishQuestion:
(BJVQuestion *)question;

/**
 收到问答回复
 #discussion 问答如果已经有了回复，取消发布之后重新发布的情况，只回调此方法，不回调发布成功
 #param question 问答全部内容
 */
- (BJLObservable)didReplyQuestion:(BJVQuestion
*)question;

/**
 取消发布问答成功
```

```
20. #discussion 任何有回答或者没有回答的问题取消发布成功都回调此方法
21. #param questionID 问答 ID
22. */
23. - (BJLObservable)didUnpublishQuestionWithQuestionID:(NSString *)questionID;
```

9. 回放聊天消息

9.1 聊天消息监听

回放聊天消息由 `BJVMessageVM` 管理，在 `BJVRoom` 的实例创建成功之后，可以对 `self.room.messageVM` 添加监听。

Core SDK 不维护聊天消息的列表，而是通过 消息覆盖更新、增量更新 的回调通知 UI 进行维护。

- 消息覆盖更新，重置列表

```
1. // 消息覆盖更新
2. - (BJLObservable)receivedMessagesDidOverwrite:(nullable NSArray<BJLMessage *> *)messages;
3.
4. // example：监听消息覆盖更新
5. bjl_weakify(self);
6. [self bjl_observe:BJLMakeMethod(room.messageVM, receivedMessagesDidOverwrite:)
7.         observer:^BOOL(NSArray<BJLMessage *> *messageArray){
8.             bjl_strongify(self);
9.             // !!!: 回放消息数有限，且 key 是唯一的，重置列表时不需要清除高度缓存
```

```
            self.allMessages = [messageArray mutableCopy];
            [self.tableView reloadData];
            return YES;
        }];
```

- 消息增量更新，扩展列表

```
// 消息增量更新
- (BJLObservable)didReceiveMessages:(nullable NSArray<BJLMessage *> *)messages;

// example：监听消息增量更新
bjl_weakify(self);
[self bjl_observe:BJLMakeMethod(room.messageVM, didReceiveMessages:)
            observer:^BOOL(NSArray<BJLMessage *> *messageArray){
                bjl_strongify(self);
                if (!messageArray.count) {
                    return YES;
                }
                [self.allMessages addObjectsFromArray:messageArray];
                [self.tableView reloadData];
                return YES;
            }];
```

9.2 （重要）聊天列表界面性能优化

显示聊天消息的 UI 性能是需要重点优化的，大幅跳转回放视频进度会导致聊天数据的大量更新，此时回放聊天界面性能消耗过高会导致 UI 卡顿、音视频和文档不同步，严重时会卡死主线程。

优化方式主要有：使用高度缓存、控制列表刷新频率、分页加载等等。具体实现可以参考 BJPlaybackUI SDK 的 BJPChatMessageViewController.m 文件。如果成功优化之后依旧存在卡顿，可以使用 `profile` 调试工具针对性能消耗较大的功能进行针对性的优化。

10. 用户列表

回放用户列表由 `BJVOnlineUserVM` 管理，在 `BJVRoom` 的实例创建成功之后，可以对 `self.room.onlineUsersVM` 添加监听。

- 房间用户列表

注意：长期课程的回放，除第1节之外的后续课节，房间用户列表信息不准确

```
/** 在线人数 */
@property (nonatomic, readonly) NSInteger onlineUsersTotalCount;

/** 在线用户 */
@property (nonatomic, readonly, nullable, copy) NSArray<BJVUser *> *onlineUsers;
```

```
// example：监听房间用户列表的变化
bjl_weakify(self);
[self bjl_kvo:BJLMakeProperty(self.room.onlineUsersVM, onlineUsers)
    observer:^BOOL(id _Nullable now, id _Nullable old, BJLPropertyChange * _Nullable change) {
        bjl_strongify(self);
```

```
        self.userList = [self.room.onlineUsersVM.onlineUsers copy];
        [self.tableView reloadData];
        return YES;
    }];
```

- 用户进出房间

```
/** 在线用户列表覆盖更新
 同时更新 `onlineUsers` */
- (BJLObservable)onlineUsersDidOverwrite:(NSArray<BJVUser *> *)users;

/** 有用户进入房间
 同时更新 `onlineUsers` */
- (BJLObservable)onlineUsersDidEnter:(NSArray<BJVUser *> *)users;

/** 有用户退出房间
 同时更新 `onlineUsers` */
- (BJLObservable)onlineUsersDidExit:(NSArray<BJVUser *> *)users;
```

- 当前主讲人

```
/** 当前主讲 */
@property (nonatomic, readonly, nullable, copy) BJVUser *currentPresenter;
```

```
// example：监听房间主讲人变化
bjl_weakify(self);
[self bjl_kvo:BJLMakeProperty(self.room.onlineUsersVM, currentPresenter)
```

```
    observer:^BOOL(id _Nullable now, id _Nullable old, BJLPropertyChange * _Nullable change) {
        bjl_strongify(self);
        BJVUser *presenter = self.room.onlineUsersVM.currentPresenter;
        NSLog(@"当前主讲为:%@", presenter.name);
        return YES;
    }];
```

- 发言用户列表变化

```
/**
 音视频用户列表变更
 @param mediaUsers 音视频用户列表
 */
- (BJLObservable)mediaUsersDidUpdate:(NSArray<BJVUser *> *)mediaUsers;
```

```
// example：监听房间发言用户列表的变化
bjl_weakify(self);
[self bjl_observe:BJLMakeMethod(self.room.onlineUsersV
 mediaUsersDidUpdate:)
        observer:^BOOL(NSArray<BJVUser *> *mediaUsers) {
            bjl_strongify(self);
            self.mediaUsers = [mediaUsers copy];
            [self.tableView reloadData];
            return YES;
        }];
```

下载功能集成

建议： 自主集成下载功能之前，可参考 BJVideoPlayerUI SDK demo 的代码，demo 提供了清晰的下载模块 API 调用方式。

1. 功能点说明

1.1 后台下载

1. 2.x 的下载通过 `NSURLSession` 实现，支持后台下载 —— App 被杀死之后在电量充足、接入 Wi-Fi 的情况下仍可继续完成下载，参考苹果官方关于后台下载的文档；

2. 由于 NSURLSession 限制，不建议做下载队列管理，等待中的队列实际上是被暂停，后台下载会失效，建议使用 `priority` 属性设置下载任务的优先级；

3. 对于 NSURLSession 的设置，例如是否支持 4G 下载等，需要在 `application:will/didFinishLaunchingWithOptions:` 中**第一时间** 设置 `BJLDownloadManager` 的 `classDelegate` 属性，并实现 `BJLDownloadManagerClassDelegate` 中定义的方法；

4. 如果 AppDelegate 中实现了这个方法 `-handleEventsForBackgroundURLSession:completionHandler:`，需要在方法的实现中调用 BJLDownloadManager 的 `+handleEventsForBackgroundURLSession:completionHandler:`，否则后台下载任务无法正常恢复；

2. 引入头文件

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

3. 设置专属域名

参考全局设置部分，点播、回放、下载通用，项目工程内**只需要设置一次**。

4. 设置 `BJVRequestTokenDelegate`，管理 token

参考全局设置部分，点播、回放、下载通用，项目工程内**只需要设置一次**。

5. 初始化下载管理类 BJVDownloadManager

`BJVDownloadManager` 的具体用法可以参考 BJVideoPlayerUI SDK demo 中 `BJVDownloadViewController.m` 文件。

5.1 定义下载 manager 属性

```
@property (nonatomic) BJVDownloadManager *downloadManager;
```

5.2 创建 manager 实例

5.2.1 初始化

创建 `BJVDownloadManager` 支持设置多个参数，`identifier` 是每个实例的唯一标识，可传入集成方的用户账号，用来**区分不同账号的下载数据**。`inCaches` 可以让下载文件保存在 caches 目录（系统可能会自动清理下载文件，但会保留下载记录，支持重新下载文件），参数默认为 NO。

```
// 指定标识，不保存在 cache 目录
```

```
+ (instancetype)downloadManagerWithIdentifier:(NSString *)identifier;

// 指定标识、配置是否保存在 cache 目录
+ (instancetype)downloadManagerWithIdentifier:(NSString *)identifier inCaches:(BOOL)inCaches;

// example
self.manager = [BJVDownloadManager downloadManagerWithIdentifier:@"user.identifier" inCaches:NO];
```

5.2.2 下载 `NSURLSessionConfiguration` 配置

对于 NSURLSession 的设置，例如是否支持 4G 下载等，需要：

- 在 `application:will/didFinishLaunchingWithOptions:` 中 **第一时间** 设置 `BJLDownloadManager` 的 `classDelegate` 属性

```
// 设置 Class 代理
+ (nullable id<BJLDownloadManagerClassDelegate>)classDele
+ (void)setClassDelegate:(nullable id<BJLDownloadManagerClassDelegate>)classDele

// example
[BJVDownloadManger setClassDelegate:self];
```

- 实现 `BJLDownloadManagerClassDelegate` 协议中的 `downloadManager:URLSessionConfiguration:` 方法，在该方法中更改 `NSURLSessionConfiguration` 属性

```
//  Class 代理方法示例
```

```
- (void)downloadManager:(BJLDownloadManager *)downloadManager URLSessionConfiguration:(NSURLSessionConfiguration *)configuration {
    // 在这里根据需要修改 configuration 的配置项
    // configuration.allowsCellularAccess = YES; // The default value is YES.
    // configuration.HTTPMaximumConnectionsPerHost = <#MAX#>; // The default value is 6 in macOS, or 4 in iOS.
}
```

程序运行过程中更改 NSURLSession 的设置，需要：

- 调用 `BJLDownloadManager` 的 `refreshURLSessionWithCompletion:` 方法，**注意** 在 `refreshURLSessionWithCompletion:` 方法回调 `completion` 之前不要对下载任务进行 `add`、`remove`、`pause`、`resume` 等操作

```
[self openLoadingView];
// 开始刷新 NSURLSession
[self.downloadManager refreshURLSessionWithCompletion:^(BJLDownloadM * _Nonnull downloadManager) {
    // 刷新 NSURLSession 完成
    [self closeLoadingView]; // userInteractionEnabled: NO
}];
```

- 然后在 `downloadManager:URLSessionConfiguration:` 方法中给 `NSURLSessionConfiguration` 的属性设置不同的值

6. 下载任务管理

下载任务由 `BJVDownloadManager` 定义的相关方法管理。需要注意的是，合并回放不支持下载。

6.1 校验下载任务是否可添加

注意:视频下载之前都需要先调用 `validateItemWithVideoID:` 或者 `validateItemWithClassID:sessionID:` 校验点播/回放是否可以下载。

```
/**
 校验下载任务对象是否可以添加
  #discussion 合并回放SDK内部无法判断，需要外部对接端判断，合并回放不支持下载

 #return 是否可添加
 #discussion 已经添加过/已下载完成 的任务，方法返回 NO
 */
- (BOOL)validateItemWithVideoID:(NSString *)videoID;
- (BOOL)validateItemWithClassID:(NSString *)classID sessionID:(nullable NSString *)sessionID;

// example: 判断点播视频是否可下载
BOOL validate = [self.downloadManager validateItemWithVideoID:videoID];

// example: 判断回放是否可下载
BOOL validate = [self.downloadManager validateItemWithClassID:classID sessionID:sessionID];
```

6.2 添加下载任务

- 添加点播下载任务

```
/**
 点播
 #param videoID      视频 ID
 #param encrypted    是否加密
 #param preferredDefinitionList 下载清晰度列表，按顺序匹配、没有匹配将导致下载失败，传 nil 使用默认清晰度
 #discussion preferredDefinitionList 列表元素为清晰度的标识字符串，现有标识符：low（标清），high（高清），superHD（超清），720p，1080p，audio（纯音频），可根据实际情况动态扩展
 #param setting      添加成功后用于设置 item 属性的回调
 #return 添加成功返回 BJVDownloadItem 实例，videoID 已存在会返回 nil
 */
- (nullable BJVDownloadItem *)addDownloadItemWithVideoID:(NSString *)videoID
                                               encrypted:(BOOL)encrypted
                                   preferredDefinitionList:(nullable NSArray<NSString *> *)preferredDefinitionList;
- (nullable BJVDownloadItem *)addDownloadItemWithVideoID:(NSString *)videoID
                                               encrypted:(BOOL)encrypted
                                   preferredDefinitionList:(nullable NSArray<NSString *> *)preferredDefinitionList
```

```
                                    setting:(nullable
void (^)(BJVDownloadItem *item))setting;
```

```
// 示例代码
if (![self.downloadManager
validateItemWithVideoID:videoID]) {
    NSLog(@"video existed: videoID %@",
videoID);
    return;
}
BJVDownloadItem *downloadItem =
[self.downloadManager
addDownloadItemWithVideoID:videoID
                                encrypted:encrypted

preferredDefinitionList:preferredDefinitionList

setting:^(BJVDownloadItem * _Nonnull item) {
                                    // 集成方鉴权，根据需
要设置
                                    item.accessKey =
accessKey;
                                    // 集成方自定义信息
                                    item.userInfo =
userInfo;
                                }];
```

- 添加回放下载任务

```
/**
 回放
 #param classID      课程 ID
 #param sessionID    课节 I
 #param encrypted    是否加密
```

```
 #param preferredDefinitionList  下载清晰度列表，列表元素为 NSNumber<BJVDefinitionType> *，按顺序匹配、没有匹配将导致下载失败，传 nil 使用默认清晰度
 #discussion preferredDefinitionList 列表元素为清晰度的标识字符串，现有标识符：low（标清），high（高清），superHD（超清），720p，1080p，audio（纯音频），可根据实际情况动态扩展
 #param setting      添加成功后用于设置 item 属性的回调
 #return 添加成功返回 BJVDownloadItem 实例，classID+sessionID 已存在会返回 nil
 */
- (nullable BJVDownloadItem *)addDownloadItemWithClassID:(NSString *)classID
                                               sessionID:(nullable NSString *)sessionID
                                               encrypted:(BOOL)encrypted
                                     preferredDefinitionList:(nullable NSArray<NSString *> *)preferredDefinitionList;
- (nullable BJVDownloadItem *)addDownloadItemWithClassID:(NSString *)classID
                                               sessionID:(nullable NSString *)sessionID
                                               encrypted:(BOOL)encrypted
                                     preferredDefinitionList:(nullable NSArray<NSString *> *)preferredDefinitionList
                                                 setting:(nullable void (^)(BJVDownloadItem *item))setting;
```

```
// 示例代码
if (![self.manager validateItemWithClassID:classID sessionID:sessionID]) {
    NSLog(@"playback existed: %@-%@", classID, sessionID);
    return;
}
BJVDownloadItem *downloadItem = [self.downloadManager addDownloadItemWithClassID:classID
                                                                        sessionID:sessionID
                                                                        encrypted:encrypted
                                                                preferredDefinitionList:preferredDefinitionList
                                                                          setting:^(BJVDownloadItem * _Nonnull item) {
                                                                              // 集成方鉴权，根据需要设置
                                                                              item.accessKey = accessKey;
                                                                              // 集成方自定义信息
                                                                              item.userInfo = userInfo;
                                                                          }];
```

6.3 纯音频下载

同一个视频只能下载纯音频或其他的一种清晰度，按照传入的清晰度列表匹配第一个满足条件的清晰度。下载纯音频时，调用上面添加下载任务方法时，`preferredDefinitionList` 参数传入 `@[@"audio"]` 即可。

7. 获取下载任务列表

- 所有已添加的下载任务，包括下载中、下载完成和下载失败的

```
@property (nonatomic, readonly, copy) NSArray<__kindof BJLDownloadItem *> *downloadItems;

// example: 获取所有下载任务
NSArray *allDownloadItems = self.downloadManager.downloadItems;
```

- 获取不同下载状态的下载任务

注意: `downloadItemsWithStates:` 方法的参数一定要以 `NSNotFound` 结束

```
// !!!: REQUIRES NSNotFound TERMINATION
- (nullable NSMutableArray<__kindof BJLDownloadItem *> *)downloadItemsWithStates:(BJLDownloadItemState)state, ...;

// example：获取下载中的任务
NSArray<BJVDownloadItem *> *items = [self.downloadManager downloadItemsWithStates:BJLDownloadItemState_r NSNotFound];
```

8. 下载任务数据类型 BJVDownloadItem

添加下载任务方法返回值为 `BJVDownloadItem` 实例，对应单个下载任务，包含下载任务的各项信息、状态等。

8.1 任务信息

```objc
/** 点播视频 ID*/
@property (nonatomic, readonly, nullable) NSString *videoID;
/** 回放课程、课节 ID*/
@property (nonatomic, readonly, nullable) NSString *classID, *sessionID;
/** 是否为回放 item*/
@property (nonatomic, readonly) BOOL isPlayback;
/** 文件是否加密*/
@property (nonatomic, readonly) BOOL isEncrypted;
/** 视频播放信息*/
@property (nonatomic, readonly, nullable) BJVPlayInfo *playInfo;
/** 视频、信令文件 */
@property (nonatomic, readonly, nullable) BJLDownloadFile *videoFile, *signalFile;
/** 封面、水印图片*/
@property (nonatomic, readonly, nullable) BJLDownloadFile *coverImageFile, *watermarkImageFile;
/** 视频清晰度*/
@property (nonatomic, readonly) BJVDefinitionInfo *currentDefinitionInfo;
```

8.2 监听下载任务信息变化

对于每一个下载任务，可从 `BJVDownloadItem` 的属性读取下载相关的信息。

属性对应关系如下：

属性名	含义
state	下载状态，参考

	BJLDownloadItemState
priority	下载优先级，取值范围 0.0 - 1.0，参考 NSURLSessionTask.priority
progress	下载进度，progress.fractionCompleted 取值范围 0.0 - 1.0，表示进度百分比
downloadFiles	下载文件数组
totalSize	下载文件总大小单位字节
bytesPerSecond	下载速度，单位字节每秒
coverImageFile	封面图片

8.2.1 监听 `BJVDownloadItem` 变化更新下载列表 UI

- 设置 `BJLDownloadManager` 的 `delegate` 属性

```
// 设置代理
@property (nonatomic, weak, nullable) id<BJLDownloadManagerDelegate> delegate;

// example
self.downloadManger.delegate = self;
```

- 实现 `BJLDownloadManagerDelegate` 协议中的 `downloadManager:downloadItem:didChange:` 方法，在该方法中更新视图

```
- (void)downloadManager:(BJVDownloadManager *)downloadManager
        downloadItem:(BJVDownloadItem *)downloadItem
            didChange:(BJLPropertyChange *)change
{
```

```
    NSInteger index = [self.downloadManager.downloadItems indexOfObject:downloadItem];
    if (index == NSNotFound) {
        return;
    }

    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:index inSection:0];
    BJVDownloadItemCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
    [cell updateWithItem:downloadItem];
}
```

8.2.2 显示下载进度和速度

由于下载进度的变化过于频繁，并且不在主线程回调，需要切换到主线程才能更新 UI，因此会严重影响渲染帧率，多个任务同时下载时更加严重。

另外，当进度没有变化时速度需要用 `timer` 来实现衰减的效果，但是每个 `BJVDownloadItem` 内置一个 `timer` 就有点浪费了。

因此

- 使用一个全局的 `timer` 定时、主动获取下载进度和速度，并且只在必要时才去一次性的重新渲染视图

```
    bjl_weakify(self);
    self.progressTimer = [NSTimer bjl_scheduledTimerWithTimeInterval:0.1 repeats:YES block:^(NSTimer * _Nonnull timer) {
        // self dealloc 后销毁 timer
        bjl_strongify_ifNil(self) {
            [timer invalidate];
```

```
            return;
        }
        // 视图不显示时跳过
        if (!self.isViewLoaded || !self.view.window) {
            return;
        }
        // 只刷新可见 cell
        NSArray<NSIndexPath *> *indexPaths =
[self.tableView indexPathsForVisibleRows];
        for (NSIndexPath *indexPath in indexPaths) {
            BJVDownloadItem *item =
[self.downloadManager.downloadItems
bjl_objectAtIndex:indexPath.row];
            BJVDownloadItemCell *cell =
[self.tableView cellForRowAtIndexPath:indexPath];
            // 尝试更新
            [cell tryToUpdateWithItem:item];
        }
    }];
    // 避免滑动时 timer 回调不执行
    [[NSRunLoop currentRunLoop]
addTimer:self.progressTimer
forMode:NSRunLoopCommonModes];
```

- 在 `UITableViewCell` 中缓存上一次下载进度和速度，在 timer 执行时检查变化，只有变化足够大时才更新视图

```
// 缓存进度和速度
@property (nonatomic) BJLProgress
cachedProgress;
@property (nonatomic) long long
cachedBytesPerSecond;

- (void)prepareForReuse {
    [super prepareForReuse];
```

```
    // prepare 时重置缓存
    self.cachedProgress = BJLProgressMake(0, 0);
    self.cachedBytesPerSecond = - 1;
}

- (void)tryToUpdateWithItem:(BJVDownloadItem *)item {
    // 尝试更新时与缓存比较，这里 `progress.fractionCompleted` 变化达到 0.1% 或者下载速度有变化时才会更新视图
    if ([item compareWithProgress:self.cachedProgress bytesPerSecond:self.cachedBytesPerSecond]) {
        [self updateWithItem:item];
    }
}

- (void)updateWithItem:(BJVDownloadItem *)item {
    // 更新视图时更新缓存
    self.cachedProgress = item.progress;
    self.cachedBytesPerSecond = item.bytesPerSecond;
    // 更新视图
    ...
}
```

8.3 下载状态说明

需要注意的是：在拿到 `BJVDownloadItem` 更新UI时，需要结合 `BJVDownloadItem` 的 `state` 和 `error` 对当前下载任务作出正确的判断。

- `BJLDownloadItemState_paused` && `error` 表示下载中出错

- `BJLDownloadItemState_completed` && `error` 表示下载完文件丢失

```
typedef NS_ENUM(NSInteger, BJLDownloadItemState) {
    BJLDownloadItemState_invalid = - 1, // NOT be added to any BJLDownloadManager, or be invalidated
    BJLDownloadItemState_running,
    BJLDownloadItemState_paused, // paused + error = error occurred
    BJLDownloadItemState_completed, // completed + error = file lost
    BJLDownloadItemState_suspended = BJLDownloadItemState_paused // 与 BJLDownloadItemState_paused 等效，将废弃
};
```

9. 下载任务控制

- 开始：调用 `BJVDownloadItem` 的 `resume` 方法

```
[item resume];
```

- 暂停：调用 `BJVDownloadItem` 的 `pause` 方法

> *`pause` 方法用于替代 `suspend`（`suspend` 现在仍然可用），解决了频繁调用 `suspend`、`resume` 可能会出现的下载进度不动的问题，因此 `restart` 方法也不需要了*

```
[item pause];
```

- 删除：调用 `BJVDownloadManager` 的 `removeDownloadItemWithIdentifier:` 方法

```
1. [self.downloadManager
   removeDownloadItemWithIdentifier:item.itemIdenti
```