Android 点播回放 Core SDK

- **Demo 地址**
- **JavaDoc 地址**
- Platform: API 21+
- CPU: ARM-v7a, ARM64-v8a
- IDE: **Android Studio** Recommend
- Change Log
- 点播3.X SDK 文档、回放 3.X SDK 文档

1. 简介

百家云 Android 点播回放 core sdk 是一个集点播和回放于一体的无 UI 纯实现的代码集，点播功能包括在线视频、离线下载、离线播放。回放功能依赖点播播放器之外还包含PPT、聊天消息等模块，同时也支持离线下载和播放，百分百还原直播场景。

1.1. 功能描述

SDK 支持Android5.0（api level 21）及以上

功能	描述
在线播放	支持百家云后台配置视频播放
播放器视图	SDK 仅提供 surfaceview/textureview 的视图，控制条等组件需要自行实现

视频缓存	支持百家云后台配置视频的缓存功能
离线播放	支持本地视频， 加密和不加密详见视频缓存模块

2. 快速集成

2.1. 添加 maven 仓库

从 4.0 SDK 起，引入了新的 `nexus.baijiayun.com` 仓库。

gradle 7.1 及以上

```
maven {
        allowInsecureProtocol true
        url 'http://nexus.baijiayun.com/nexus/content/groups/apublic/'
    }
maven { url 'https://git2.baijiashilian.com/open-android/maven/raw/master/' }
```

gradle 7.1 以下

```
maven {url 'http://nexus.baijiayun.com/nexus/content/groups/apublic/'}
maven { url 'https://git2.baijiashilian.com/open-android/maven/raw/master/' }
```

2.2. 版本号说明

版本号格式为 `大版本.中版本.小版本[-alpha(测试版本)/beta(预览版本)]` ：

- 测试版本和预览版本可能不稳定，请勿随意尝试。
- 小版本升级只改 BUG、UI 样式优化，不会影响功能。
- 中版本升级、修改功能，更新 UI 风格、布局，会新增 API、标记 API 即将废弃，但不会导致现有 API 不可用。
- 大版本任何变化都是有可能的。

首次集成建议选择最新正式版本(版本号中不带有 `alpha` 、 `beta` 字样)，**版本升级后请仔细阅读**

2.3. 添加依赖

最新版本请点击自取，Releases · Open Android / VideoPlayer2.0Demo · GitLab

```
implementation'com.baijia.player:VideoplayerCore:
```

2.4. 初始化 SDK

```
public class App extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
         new BJYPlayerSDK.Builder(this)
                .setCustomDomain("demo123")
                .setEncrypt(true)
                .build();
    }
}
```

通过 `BJYPlayerSDK.Builder` 配置参数，可配置的参数有

- 设置开发者模式。开发者模式开启后会打印关键日志、显示播放器调试模板，方便开发，**正式发版建议关掉**

```
/**
 * 设置开发者模式
 * @param isDevelopMode true 设置为开发者模式
 * @return
 */
 public Builder setDevelopMode(boolean isDevelopMode)
```

- 设置专属域名。专属域名从百家云账号中心获取，传入规则：例如专属域名为 `demo123.at.baijiayun.com`，则前缀为 `demo123`，参考 专属域名说明。

```
/**
 * 设置专属域名
 *
 * @param domain 专属域名
 */
 public Builder setCustomDomain(String domain)
```

- 设置是否加密，对在线播放和下载均有效

```
/**
 * 设置加密
 * @param isEncrypt true 加密，对在线和下载均有效
 * @return
 */
 public Builder setEncrypt(boolean isEncrypt)
```

- 设置是否解析 user 相关信令（默认未开启）

```
/**
 * 开启回放user相关信令
 *
 * @return
 */
public Builder enablePlaybackUserSignal() {
    enablePlaybackUserSignal = true;
    return this;
}
```

- 设置静态课件压缩尺寸，默认1080，取值范围[720,4096]。

  取值越大压缩精度越小，图片越清晰，内存占用越大。

  ```
  BJYPlayerSDK.STATIC_PPT_DEFAULT_SIZE = 1080;
  ```

- 设置是否禁用动态 PPT，默认开启。
  静态课件为一张张图片，动效课件为 webview 支持课件动效。

  ```
  // false 允许动态课件，true 强制走静态课件。
  BJYPlayerSDK.DISABLE_ANIM_PPT = false;
  ```

3. 点播部分

快速跑通点播功能分为以下几个步骤：

3.1. 播放视频

3.1.1. 创建播放器

通过 `VideoPlayerFactory` 构造播放器，示例代码如下，请按需配置。

```
videoPlayer = new VideoPlayerFactory.Builder()
            //开启循环播放
            .setSupportLooping(true)
            //关闭后台音频播放
            .setSupportBackgroundAudio(false)
            //开启记忆播放
            .setSupportBreakPointPlay(true, this)
            //设置在线播放清晰度匹配规则，这里仅传Audio则优先播放纯音频，如无纯音频则播放服务器返回的默认清晰度
            .setPreferredDefinitions(new ArrayList<VideoDefinition>(){{add(VideoDefinition.Audio);}})
            //绑定activity生命周期
            .setLifecycle(getLifecycle())
            .build();
```

上述 API 说明见 4.1 节。

`IBJYVideoPlayer` 仅实现播放器相关逻辑，视频展示需要结合 3.1 节中 `BJYPlayerView` 显示。

3.1.2. 创建视频播放视图

- Java 代码初始化

```
BJYPlayerView playerView = findViewById(R.id.activity_new_video_fl);
videoPlayer.bindPlayerView(playerView);
//设置视频渲染载体，需在bindPlayerView之后设置。默认 surfaceView
playerView.setRenderType(IRender.RENDER_TYPE_S
```

```
//设置视频画面裁剪方式，默认16：9
playerView.setAspectRatio(AspectRatio.AspectRatio
```

- xml 初始化

```
<com.baijiayun.videoplayer.widget.BJYPlayerView
    android:id="@+id/activity_new_video_fl"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    app:render_type="surface_view"
    app:aspect_ratio="fit_parent_16_9"
    />
```

`BJYPlayerView` 是视频播放控件，继承自FrameLayout。对外可设置实际的渲染载体（surfaceView或textureView），默认使用 surfaceView。可设置视频界面裁剪方式，分别有如下五种：

类型	描述
fit_parent_16_9	宽高比16:9
fit_parent_4_3	宽高比4:3
match_parent	宽高直接为surfaceview或textureView
fill_parent	完全填充(宽优先)
fit_parent	等比填充(高度优先)
wrap_content	原画

3.1.3. 播放器绑定视图

```
IBJYVideoPlayer
videoPlayer.bindPlayerView(playerView);
```

`IBJYVideoPlayer` 持有 `BJYPlayerView` 视图进行视频播放。

3.1.4. 绑定视频源

- 传入 videoId 和 token，播放在线视频

```
/**
 * 设置播放百家云在线视频
 *
 * @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
 */
 void setupOnlineVideoWithId(long videoId, String token);


/**
    * 设置播放百家云在线视频
    * @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
    * @param accessKey  第三方鉴权(可选)
    */
   void setupOnlineVideoWithId(long videoId, String token, String accessKey);
```

- 传入本地视频对应的 `DownloadModel`，播放本地视频

```
/**
    * 设置播放百家云下载的本地视频(支持记忆播放)
    *
    * @param downloadModel 百家云下载的model
```

```
    */
    void setupLocalVideoWithDownloadModel(DownloadMod
     downloadModel);
```

设置完视频源之后会自动播放视频。

3.2. 设置播放器

3.2.1. 设置播放器参数

`IBJYVideoPlayer` 对外提供一系列 setter 函数设置播放器的行为，如下

```
/**
    * 设置第三方用户信息，用于后台统计播放时长等信息，建议设置。
    *
    * @param userName     第三方用户名
    * @param userIdentity 第三方用户标识
    */
    void setUserInfo(String userName, String userIdentity);

    /**
    * 设置是否支持进入后台时继续播放音频
    *
    * @param enable boolean 默认false
    */
    void supportBackgroundAudio(boolean enable);

    /**
    * 设置是否支持循环播放
    *
```

```
     * @param looping 是否循环播放
     */
    void supportLooping(boolean looping);

    /**
     * 设置启用记忆播放，仅本地设备，如果需要跨设备同步播放进度，需结合 play(startOffset) 自行实现。
     *
     * @param context needed by SharedPrefs 做持久化需要用到
     */
    void enableBreakPointMemory(Context context);

    /**
     * 设置setupVideoIdWithToken之后是否自动播放，默认自动播放
     *
     * @param autoPlay
     */
    void setAutoPlay(boolean autoPlay);
```

3.2.2. 获取播放器状态

`IBJYVideoPlayer` 提供了如下一系列 getter 函数获取播放器状态。

```
    /**
     * 获取当前播放进度，单位秒
     *
     * @return
     */
    int getCurrentPosition();

    /**
```

```
     * 获取视频时长，单位秒
     *
     * @return
     */
    int getDuration();

    /**
     * 获取当前缓存百分比
     *
     * @return
     */
    int getBufferPercentage();

    /**
     * 获取播放速率
     * @return
     */
    float getPlayRate();

    /**
     * 是否播放本地视频
     * @return true，播放本地视频
     */
    boolean isPlayLocalVideo();
```

其中 `PlayerStatus` 为播放器状态，见 PlayerStatus。

```
/**
     * 获取播放器状态
     *
     * @return 播放器状态
     */
    PlayerStatus getPlayerStatus();
```

获取视频源信息，由服务端返回，`BJYVideoInfo` 字段介绍见 BJYVideoInfo。

```
/**
 * 获取当前播放的视频信息
 *
 * @return Video Info
 */
@Nullable
BJYVideoInfo getVideoInfo();
```

3.2.3. 视频控制

`IBJYVideoPlayer` 对外提供一系列 API 控制视频播放状态，播放、暂停、快进、快退、切换清晰度、倍速播放。

```
/**
 * 开始播放
 */
void play();

/**
 * 从startOffset开始播放，单位为秒
 *
 * @param startOffset
 */
void play(int startOffset);

/**
 * 暂停播放
 */
void pause();

/**
 * 是否正在播放
```

```
     *
     * @return
     */
    boolean isPlaying();

    /**
     * 停止播放
     */
    void stop();

    /**
     * seek
     *
     * @param time 时间
     */
    void seek(int time);

    /**
     * 倍速播放[0.5 ~ 2.0]倍
     *
     * @param rate 倍率
     */
    void setPlayRate(float rate);

     /**
     * 释放播放器
     */
    void release();
```

设置清晰度播放优先级，可自定义规则，如

```
//清晰度匹配规则,720P>超清>高清>标清>1080P>音频
private List<VideoDefinition>
preferredDefinitionList = new ArrayList<>
```

```
(Arrays.asList(VideoDefinition._720P,
        VideoDefinition.SHD, VideoDefinition.HD,
VideoDefinition.SD, VideoDefinition._1080P,
VideoDefinition.Audio));
```

```
 /**
    * 设置清晰度偏好
    * 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
    *
    * @param definitions 清晰度列表
    */
   void setPreferredDefinitions(Iterable<VideoDefinition> definitions);
```

切换清晰度，definition 需要从 `videoplayer.getVideoInfo().getSupportedDefinitionList()` 选取。

```
/**
    * 改变清晰度
    * 播放的时候调用，如果没有对应的清晰度不做处理，播本地文件不生效
    *
    * @param definition 清晰度
    * @return true切换清晰度成功  false切换清晰度失败
    */
   boolean changeDefinition(VideoDefinition definition);
```

3.3. 设置监听

3.3.1. 监听播放器状态

```
/**
    * 设置播放器状态改变回调
    *
    * @param listener 播放状态改变回调
    */
   void
addOnPlayerStatusChangeListener(OnPlayerStatus(
 listener);
```

可以在这个回调里根据播放器状态处理一些逻辑，如

```
bjyVideoPlayer.addOnPlayerStatusChangeListener(s
 -> {
          subjectOfVideoStatus.onNext(status);
          if (status ==
PlayerStatus.STATE_PREPARED) {
             // 视频加载完成隐藏视频封面
          }
       });
```

3.3.2. 监听播放器进度

```
/**
    * 设置播放进度回调
    *
    * @param listener 播放进度回调
    */
   void
addOnPlayingTimeChangeListener(OnPlayingTimeC
 listener);
```

for example:

```
bjyVideoPlayer.addOnPlayingTimeChangeListener(n
 OnPlayingTimeChangeListener() {
            // currentTime 当前进度
            // duration 总时长
            @Override
            public void onPlayingTimeChange(int
currentTime, int duration) {
                // 在这里更新进度条
            }
        });
```

3.3.3. 监听 seek 结束

因为关键帧的原因，seek 结束起播的播放时间可能与预期相差 1~2 秒。

```
/**
    * 设置seek结束回调
    * @param onSeekCompleteListener
    */
   void
addOnSeekCompleteListener(OnSeekCompleteListe
 onSeekCompleteListener);

public interface OnSeekCompleteListener {

   /**
    * seek结束回调
    * @param beforeSeekPosition   seek前时间
    * @param seekPosition         seek结束时间
    */
   void onSeekComplete(int beforeSeekPosition,
int seekPosition);
}
```

3.3.4. 监听播放器出错

```
/**
    * 设置播放器出错回调
    *
    * @param listener 播放器出错回调
    */
   void addOnPlayerErrorListener(OnPlayerErrorListener listener);
```

3.3.5. 监听 token 失效

token 有时效，超过限定时间 token 即失效，可以在此回调中监听处理。

```
 /**
    * 设置token失效回调
    *
    * @param listener token失效回调
    */
   void setOnTokenInvalidListener(OnTokenInvalidListener listener);
```

for example:

```
 videoPlayer.setOnTokenInvalidListener(new OnTokenInvalidListener() {
        @Override
        public void onTokenInvalid(long vid, OnTokenFetchedListener callback) {
            // request new token
            String newToken = "newToken";
```

```
videoPlayer.setupOnlineVideoWithId(vid,
newToken);
        }
    });
```

3.4. 其它功能

3.4.1. 点播评论

参考点播评论获取token接口 获取 token。

注意：启用点播评论功能还需要后台开启配置项，可通过 `IBJYVideoPlayer.isEnableVideoComment()` 查询是否已配置。

注：点播评论相关的token传入一致使用”Bearer $token”

```
PlayerDataLoader
/**
 * 获取视频评论列表
 */
public Observable<LPCommentDataModel>
getVideoCommentList(String token, Strin videoId,
int page, int pageSize)
/**
 * 发布评论
 * @param avatar 头像，可为null
 */
public Observable<Boolean>
postComment(String token, String videoId, String
comment, String avatar)
/**
 * 点赞评论
 */
```

```
14. public Observable<Boolean> likeComment(String token, String videoId, String commentId, boolean isLike)
15. /**
16.  * 删除评论
17.  */
18. public Observable<Boolean> deleteComment(String token, String videoId, String commentId)
```

3.4.2. 自定义水印

播放器内部会读取后台配置的水印，也可以自行通过以下 API 设置自定义水印：

```
1. BJYPlayerSDK.waterMark = new VideoItem.WaterMark();
2. // 左上角显示
3. BJYPlayerSDK.waterMark.pos = LPConstants.LPPosition.LEFT_TOP;
4. BJYPlayerSDK.waterMark.url = "https://img1.baidu.com/it/u=1890390320,3399874
   w=1422&h=800";
```

注：水印大小跟随视频窗口动态调整

4. 回放部分

回放模板依赖点播播放器，同时包含一套信令处理逻辑，通过暴露特定接口实现课件翻页、聊天信息、画笔绘制、在线人员等模块。简言之，回放 = 点播播放器 + ppt + 聊天信息 + 在线人员信息。

快速跑起回放流程，分为以下步骤：

4.1. 快速集成

4.1.1. 创建播放器

参考 3.1 节。

4.1.2. 创建 PBRoom

`PBRoom` 是对回放房间的抽象接口，包含进房间、绑定播放器、暴露信令处理后的各种信息接口。

`BJYPlayerSDK.newPlayBackRoom()` 提供了多个重载的实现，轻松获取多个场景下的回放房间 PBRoom 实例。

普通回放

```
PBRoom mRoom = BJYPlayerSDK.newPlayBackRoom(this, classId, classToken);
```

参数说明如下：

```
/**
 * 创建录播回放房间    在线回放
 * @param context    上下文
 * @param classId    教室id
 * @param token      token
 * @return
 */
public static PBRoom newPlayBackRoom(Context context, long classId, String token){
    return new PBRoomImpl(context, classId, token);
}
```

长期课回放

```
PBRoom mRoom = BJYPlayerSDK.newPlayBackRoom(this, classId, sessionId, classToken);
```

参数说明如下：

```
/**
 * 创建录播回放房间 -- 在线回放[长期房间，分段]
 * @param context   上下文
 * @param classId   教室 ID
 * @param sessionId 分段id
 * @param token     token
 * @return
 */
public static PBRoom newPlayBackRoom(Context context, long classId, long sessionId, String token) {
      return new PBRoomImpl(context, classId, sessionId, token);
   }
```

裁剪回放

```
PBRoom mRoom = BJYPlayerSDK.newPlayBackRoom(this, classId, sessionId, version, classToken);
```

参数说明如下：

```
/**
   * 创建录播回放房间 -- 在线回放[长期房间，分段]
   *
   * @param context
   * @param classId   教室 ID
```

```
    * @param sessionId 分段id
    * @param version   裁剪版本
    * @param token     token
    * @return
    */
   public static PBRoom
newPlaybackRoom(Context context, long classId,
long sessionId, int version, String token)
```

合并回放

合并回放为多段回放拼接起来的回放，顺序播放。视频时长为多段视频的总时长。

```
/**
    * 合并回放进房间
    *
    * @param context
    * @param mixedId    合并id
    * @param mixedToken 合并token
    * @return
    */
   public static PBRoom
newPlaybackRoom(Context context, String
mixedId, String mixedToken)
```

离线回放

```
/**
    * 创建回放房间 -- 离线回放, 直接传
downloadModel(支持记忆播放)
    *
    * @param context            上下文
    * @param videoDownloadModel  视频 model
    * @param signalDownloadModel 信令 model
```

```
     * @return
     */
    public static PBRoom
newPlaybackRoom(Context context,
DownloadModel videoDownloadModel,
DownloadModel signalDownloadModel)
```

4.1.3. 绑定播放器

`PBRoom` 通过持有 `IBJYVideoPlayer` 实例才能获取到播放器的各种状态回调，通过监听视频播放回调不断驱动信令引擎检索信令从而达到还原直播时的各种事件。

```
pbRoom.bindPlayer(bjyVideoPlayer);
```

4.1.4. 绑定课件

```
PPTView pptView.attachRoom(mRoom);
```

`PPTView` 的详细介绍参考 4.2 节。

4.1.5. 进入房间

```
pbRoom.enterRoom(new LPLaunchListener() {
            @Override
            public void onLaunchSteps(int step, int
totalStep) {
                // step:当前处于第几步  totalStep:总步
数  step/totalStep即进房间进度百分比
            }
            @Override
            public void onLaunchError(LPError
error) {
                //进房间出错回调
```

```
            }
            @Override
            public void onLaunchSuccess(PBRoom room) {
                //进房间成功回调
            }
        });
```

至此，回放流程已经全部完成。

4.2. PPTView

4.2.1. 简介

`PPTView` 继承自 FrameLayout，实现课件显示、画笔绘制、手势缩放。支持静态课件（图片实现）和动态课件（webview 实现，支持课件动效）两种类型，能无感知在两种类型之间切换。

4.2.2 创建 PPTView

支持 Java代码创建和 xml 声明两种方式。

```
<com.baijiayun.playback.ppt.PPTView
        android:id="@+id/pptview"
        android:layout_width="match_parent"
        android:layout_height="200dp" />
```

```
PPTView pptView = new PPTView(context)
```

4.2.3. 监听状态

```
pptView.setPPTErrorListener(new PPTView.OnPPTErrorListener() {
        @Override
```

```
        public void onAnimPPTLoadError(int errorCode, String description) {
            // 动效课件加载超时
            if (errorCode == LPError.CODE_ERROR_ANIMPPT_LOAD_TIMEOUT) {
                // 弹框让用户选择手动切换到静态课件或者继续等待
                // 切换为静态课件
                pptView.setAnimPPTEnable(false);
            }
        }

        @Override
        public void onAnimPPTLoadFinish() {
          // 动态课件加载成功
        }

        @Override
        public void onAnimPPTLoadStart() {
            // 动态课件开始加载
        }
    });
```

4.3. 聊天

```
mRoom.getChatVM().getObservableOfNotifyDataCh

.observeOn(AndroidSchedulers.mainThread())
            .subscribe((List<IMessageModel> iMessageModels) -> {
                //返回当前时刻的聊天消息集合

messageAdapter.notifyDataSetChanged();
```

```
6.              });
```

4.4. 获取在线人员

SDK 默认不处理在线人员，如需开启请参考 2.4 节开启。

```
1. mRoom.getOnlineUserVM().getObservableOfOnlineU
2. 
.observeOn(AndroidSchedulers.mainThread())
3.              .subscribe((List<IUserModel>
iUserModels) -> {
4.                  //这里返回当前在线人员集合
5. 
userAdapter.notifyDataSetChanged();
6.              });
```

4.5. 监听视频状态

监听老师开关摄像头状态，通过占位图来优化用户体验。

```
1. // 获取回放录制类型
2. //0普通大班课回放，1大班课webrtc回放，2小班课
webrtc回放, 3合流回放
3. recordType = room.getRecordType();
```

```
1.  //recordType = 1的回放是webrtc录所有人视频，强制
显示视频
2.      //recordType = 3的回放是混合录制，也强制显示
视频
3. 
compositeDisposable.add(pbRoom.getObservableO
4.              .filter(aBoolean -> recordType != 1 &&
recordType != 3)
```

```
.observeOn(AndroidSchedulers.mainThread())
        .subscribe(isVideoOn -> {
            // isVideoOn 为 false 时可以显示占位图，避免视频内容黑屏
        }));
```

4.6. 教室工具

4.6.1. 公告

```
pbRoom.getObservableOfAnnouncementChange()
.observeOn(AndroidSchedulers.mainThread())
        .subscribe(announcementModel -> {
            // 公告正文
            announcementModel.getContent();
            // 公告跳转链接
            announcementModel.getLink();
        }));
```

4.6.2. 答题器

```
pbRoom.getToolBoxVM().getObservableOfAnswerSt
.observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Consumer<LPAnswerModel>() {
                    @Override
                    public void accept(LPAnswerModel lpAnswerModel) throws Exception {
                        // 题目正文
```

```
lpAnswerModel.messageType;
                                    // 题目类型
                                    lpAnswerModel.type;
                                }
                            });
```

更详细的使用参考 QuestionToolFragement

4.6.3. 测验

测验实现为 webview，默认逻辑是离线播放不展示。

```
compositeDisposable.add(pbRoom.getToolBoxVM().
                .filter(lpJsonModel ->
!pbRoom.isPlaybackOffline())

.observeOn(AndroidSchedulers.mainThread())
                .subscribe(this::quizStart));
```

更详细的使用参考 QuizDialogFragment

4.6.4. 问答

```
compositeDisposable.add(pbRoom.getToolBoxVM().
                .subscribe(lpQuestionPullResItems -> {
                    if
(!lpQuestionPullResItems.isEmpty()) {

questionAnswerBtn.setImageResource(R.drawable.l
                    }
                }));
```

更详细的使用参考 QuestionAnswerFragment

4.7. 退出房间

```
1. //回收PPT相关资源
2. pptView.destroy();
3. //退出房间
4. mRoom.quitRoom();
```

5. 下载

5.1. Android 8.0兼容

android 8.0之后不再支持应用后台创建service，需要用户自行实现后台下载保活，具体可以参考core demo下的CustomDownloadService。之前的DonwloadService依旧保留，不过不再兼容8.0系统，同时被标记废弃。

5.2. DownloadManager

DownloadMananger使用单例模式，方便获取实例。

```
1. DownloadManager downloadManager =
   DownloadManager.getInstance(context);
```

5.2.1. 设置缓存路径

```
1. manager.setTargetFolder(Environment.getExternalS
   + "/bb_video_downloaded/")
```

5.2.2. 加载缓存记录

```
manager.loadDownloadInfo();
```

默认只加载一次，如需重新加载，需调用 loadDownloadInfo(true);

```
/**
    * 加载下载记录
    */
   public void loadDownloadInfo() {
       loadDownloadInfo(false);
   }
```

```
 /**
    * @param reload true:每次都清零后重新获取
    */
   public void loadDownloadInfo(boolean reload)
{
       loadDownloadInfo("", reload);
   }
```

```
/**
    * 按用户角色加载下载记录
    * @param userIdentify 用户唯一标识
    * @param reload 是否每次强制刷新，默认传false
    */
   public void loadDownloadInfo(String
userIdentify, boolean reload)
```

5.2.3. 设置下载清晰度匹配规则

```
manager.setPreferredDefinitionList(definitionList);
```

```
/**
 * 设置下载清晰度优先匹配顺序
 * @param definitionList
 */
public void setPreferredDefinitionList(List<VideoDefinition> definitionList)
```

如不设置，sdk默认的匹配规则如下，用户可参考自行调整顺序。

```
//下载清晰度匹配规则,音频>720P>超清>高清>标清>1080P
private List<VideoDefinition> preferredDefinitionList = new ArrayList<>(Arrays.asList(VideoDefinition.Audio, VideoDefinition._720P,
        VideoDefinition.SHD, VideoDefinition.HD, VideoDefinition.SD, VideoDefinition._1080P));
```

5.2.4. 获取下载记录

```
/**
 *
 * 获取下载记录，需在loadDownloadInfo之后
 * @return
 */
public List<DownloadTask> getAllTasks()
```

5.2.5. 删除下载

```
downloadManager.deleteTask(task);
```

```
/**
```

```
 * 删除任务 并同时删除文件
 *
 * @param task
 */
public void deleteTask(DownloadTask task)
```

5.3. 点播下载

创建点播DownloadTask

```
videoDownloadDisposable = downloadManager.newVideoDownloadTask("video", videoId, token, "extraInfo")
.observeOn(AndroidSchedulers.mainThread())
            .subscribe(downloadTask -> adapter.notifyDataSetChanged(), throwable -> {
                throwable.printStackTrace();
Toast.makeText(SimpleVideoDownloadActivity.this, throwable.getMessage(), Toast.LENGTH_LONG).show();
            });
```

参数说明：

```
/**
    * 创建点播下载
    *
    * @param fileName    保存的下载的文件名称
    * @param videoId     视频id
    * @param token       视频token
    * @param extraInfo   额外信息，客户自己定义，这里只是转存
```

```
     * @return
     */
    public Observable<DownloadTask> newVideoDownloadTask(final String fileName, final long videoId, final String token,
                                                         final List<VideoDefinition> definitions, final String extraInfo)
```

其它重载的参数说明如下：

```
/**
     * 创建点播下载
     *
     * @param fileName    保存的下载的文件名称
     * @param videoId     视频id
     * @param token       视频token
     * @param extraInfo   额外信息，客户自己定义，这里只是转存
     * @param accessKey   第三方鉴权key
     * @return
     */
    public Observable<DownloadTask> newVideoDownloadTask(String fileName, long videoId, String token, String extraInfo, String accessKey) {
        return newVideoDownloadTask(fileName, videoId, token, extraInfo, accessKey, BJYPlayerSDK.IS_ENCRYPT);
    }
```

```
/**
     * 创建点播下载
     *
```

```
    * @param fileName    保存的下载的文件名称
    * @param videoId     视频id
    * @param token       视频token
    * @param extraInfo   额外信息，客户自己定义，这里只是转存
    * @param accessKey   第三方鉴权key
    * @param isEncrypt   加密参数，true加密 （仅对当前下载有效）
    * @return
    */
   public Observable<DownloadTask> newVideoDownloadTask(String fileName, long videoId, String token, String extraInfo, String accessKey, boolean isEncrypt) {

       return newVideoDownloadTask(fileName, videoId, token, extraInfo, accessKey, isEncrypt, preferredDefinitionList);
   }
```

```
   /**
    * 创建点播下载
    *
    * @param fileName        保存的下载的文件名称
    * @param videoId         视频id
    * @param token           视频token
    * @param accessKey       第三方鉴权key
    * @param extraInfo       额外信息，客户自己定义，这里只是转存
    * @param isEncrypt       加密参数，true加密（仅对当前下载有效）
    * @param definitionList  清晰度匹配列表，从前往后匹配 （仅对当前下载有效）
    * @return
    */
```

```
13.    public Observable<DownloadTask>
newVideoDownloadTask(final String fileName,
final long videoId, final String token, final String
extraInfo, String accessKey, boolean isEncrypt,
List<VideoDefinition> definitionList)
```

5.4. 回放下载

`DownloadManager` 提供一系列重装的静态函数创建 `DownloadTask` 下载任务。

普通回放

```
1. /**
2.     * 创建回放下载任务
3.     * @param fileName     文件名称
4.     * @param roomId       房间id
5.     * @param sessionId    长期房间id，如果不是长期
房间传0的值即可
6.     * @param token        房间token
7.     * @param extraInfo    额外信息，客户自己存储的
字符串信息，sdk只是转存。不使用可传空值
8.     * @return
9.     */
10.    public Observable<DownloadTask>
newPlaybackDownloadTask(final String fileName,
final long roomId, final long sessionId, final String
token,
11.                                                final String
extraInfo)
```

裁剪回放

```
1. /**
2.     * 创建回放下载任务
```

```
    * @param fileName     文件名称
    * @param roomId       房间id
    * @param sessionId    长期房间id，如果不是长期房间传0的值即可
    * @param token        房间token
    * @param extraInfo    额外信息，客户自己存储的字符串信息，sdk只是转存。不使用可传空值
    * @param version      裁剪版本
    * @return
    */
   public Observable<DownloadTask> newPlaybackDownloadTask(final String fileName, final long roomId, final long sessionId, final String token,
                                                           final String extraInfo, int version)
```

单独设置加密和清晰度规则

```
/**
    * 创建回放下载任务
    * @param fileName        文件名称
    * @param roomId          房间id
    * @param sessionId       长期房间id，如果不是长期房间传0的值即可
    * @param token           房间token
    * @param extraInfo       额外信息，客户自己存储的字符串信息，sdk只是转存。不使用可传空值
    * @param isEncrypt       加密参数，true为加密（仅对当前下载有效）
    * @param definitionList  清晰度优先列表，从前往后匹配（仅对当前下载有效）
    * @return
    */
```

```
12.   public Observable<DownloadTask>
newPlaybackDownloadTask(final String fileName,
final long roomId, final long sessionId, final String
token,
13.                                         final String
extraInfo, boolean isEncrypt,
List<VideoDefinition> definitionList)
```

5.5. 下载状态回调

```
1. downloadTask.setDownloadListener(new
DownloadListener() {
2.             @Override
3.             public void onProgress(DownloadTask
task) {
4.                 //更新下载进度
5.                 int progress = (int)
(task.getDownloadedLength() / (float)
task.getTotalLength() * 100);
6.             }
7.
8.             @Override
9.             public void onError(DownloadTask task,
HttpException e) {
10.                //下载出错
11.                Log.e("download", "onError " +
e.getCode() + ", message=" + e.getMessage());
12.            }
13.
14.            @Override
15.            public void onPaused(DownloadTask
task) {
16.                //暂停
17.            }
```

```
18.
19.             @Override
20.             public void onStarted(DownloadTask
task) {
21.                 //开启下载
22.             }
23.
24.             @Override
25.             public void onFinish(DownloadTask
task) {
26.                 //下载完成
27.             }
28.
29.             @Override
30.             public void onDeleted(DownloadTask
task) {
31.                 //删除下载任务及文件
32.             }
33.         });
```

5.6. DownloadTask接口清单

```
1. public interface DownloadTask {
2.
3.     /**
4.      * 开始任务
5.      */
6.     void start();
7.
8.     /**
9.      * 暂停任务
10.     */
11.    void pause();
12.
```

```
13.     /**
14.      * 重新开始任务
15.      */
16.     void restart();
17. 
18.     /**
19.      * 取消任务
20.      */
21.     void cancel();
22. 
23.     /**
24.      * 删除任务和文件
25.      */
26.     void deleteFiles();
27. 
28.     /**
29.      * 设置下载状态的监听
30.      *
31.      * @param listener
32.      */
33.     void setDownloadListener(DownloadListener
    listener);
34. 
35.     /**
36.      * 获取视频相关信息
37.      *
38.      * @return
39.      */
40.     DownloadModel getVideoDownloadInfo();
41. 
42.     /**
43.      * 获取信令相关信息
44.      * @return
45.      */
46.     DownloadModel getSignalDownloadInfo();
47. 
```

```
48.    /**
49.     * 获取下载状态
50.     * New(0),          // new -> downloading
51.     * Downloading(1),  // downloading -> pause, error, finish
52.     * Pause(2),        // pause -> downloading
53.     * Error(3),        // error -> new
54.     * Finish(4),       //
55.     * Cancel(5);
56.     * @return
57.     */
58.    TaskStatus getTaskStatus();
59.
60.    /**
61.     * 获取下载速度，单位为byte
62.     * @return
63.     */
64.    long getSpeed();
65.
66.    /**
67.     * 获取下载进度
68.     * @return
69.     */
70.    float getProgress();
71.
72.    /**
73.     * 获取要下载的总字节数
74.     * @return
75.     */
76.    long getTotalLength();
77.
78.    /**
79.     * 获取已下载字节数
80.     * @return
81.     */
82.    long getDownloadedLength();
```

```
83.
84.     /**
85.      * 获取下载类别
86.      * @return
87.      */
88.     DownloadType getDownloadType();
89.
90.     /**获取视频文件名称
91.      * @return
92.      */
93.     String getVideoFileName();
94.
95.     /**
96.      * 获取信令文件名称
97.      * @return
98.      */
99.     String getSignalFileName();
100.
101.    /**
102.     * 获取视频时长
103.     * @return
104.     */
105.    long getVideoDuration();
106.
107.    /**
108.     * 获取下载视频的全路径
109.     * @return
110.     */
111.    String getVideoFilePath();
112.
113.    /**
114.     * 获取下载的信令文件的全路径
115.     * @return
116.     */
117.    String getSignalFilePath();
118. }
```

5.7. DownloadModel字段说明

DownloadModel包含下载任务的全部信息，为链表结构，链头为视频/音频信息，链尾为信令信息，如果为点播则仅有链尾，nextModel为null。

```
1.  public long videoId;                //vid
2.     public long sessionId;             //sessionId
3.     public long roomId;                //roomId
4.     public String url;                 //url
5.     public VideoDefinition definition;  //清晰度
6.     public long videoDuration;         //视频长度（服务器没传就是0）
7.     public FileType fileType;          //file type, Audio/Video/Signal
8.     public String targetName;          //保存的文件名
9.     public String videoName;           //视频标题
10.    public TaskStatus status = TaskStatus.New;  //当前状态
11.    public String targetFolder;        //保存文件夹
12.    public long totalLength;           //总大小
13.    public long downloadLength;        //已下载大小
14.    public long speed;                 //下载瞬时速度
15.    public Serializable data;          //额外的数据
16.    public boolean isEncrypt;          //加密类型 true 加密
17.    public String videoToken;          //视频token，过期逻辑上层处理
18.    public String extraInfo;           //额外信息，客户自己定义，这里只是转存
19.
20.    public DownloadModel nextModel;
```

6. 接口说明

6.1. PlayerStatus

视频状态枚举

```
public enum PlayerStatus {
    /**
     * 出错
     */
    STATE_ERROR,
    /**
     * 未初始化
     */
    STATE_IDLE,
    /**
     * 初始化
     */
    STATE_INITIALIZED,
    /**
     * 数据已准备好，待播放
     */
    STATE_PREPARED,
    /**
     * 播放中
     */
    STATE_STARTED,
    /**
     * 暂停状态
     */
    STATE_PAUSED,
    /**
     * 终止状态(已释放播放器实例)
     */
```

```
    STATE_STOPPED,
    /**
     * 播放结束
     */
    STATE_PLAYBACK_COMPLETED
}
```

6.2. VideoDefinition

清晰度枚举

```
public enum VideoDefinition {
    UNKNOWN(-1),未知
    SD(0),//标清
    HD(1),//高清
    SHD(2),//超清
    _720P(3),//720p
    _1080P(4),//1080p
    Audio(5);//音频
}
```

6.3. BJYVideoInfo

视频信息接口

```
public interface BJYVideoInfo {

    /**
     * 获取视频id
     *
     * @return
     */
    long getVideoId();

```

```
10.
11.     /**
12.      * 获取视频时长（服务端返回的时长）
13.      *
14.      * @return
15.      */
16.     int getDuration();
17.
18.     /**
19.      * 后台配置的默认清晰度
20.      *
21.      * @return
22.      */
23.     VideoDefinition getDefinition();
24.
25.     /**
26.      * 后端转码的清晰度的集合
27.      *
28.      * @return
29.      */
30.     @Nullable
31.     List<VideoDefinition>
    getSupportedDefinitionList();
32.
33.     /**
34.      * 获取视频标题
35.      * @return
36.      */
37.     String getVideoTitle();
38.
39.     /**
40.      * 获取字幕model
41.      * @return
42.      */
43.     List<SubtitleItem> getSubtitleItemList();
44. }
```

7. 错误码

错误码	说明
-10000、-101	ijkplayer内部错误,比如文件异常
10087	视频文件路径错误
10088	网络错误
-1	无网络连接
-2	移动网络播放
-4	未知错误
-5	没有找到对应清晰度
-6	离线播放传入非法路径
-7	离线播放对应路径的文件不存在
5101、5102、5103	token异常
403	视频地址过期

注：ijkplayer没有给出具体错误码，详见源码的头文件
ijkplayer_ android_def.h
其余错误码可参考AndroidSDK的MediaPlayer

8. 混淆规则 & 常见问题

见 https://dev.baijiayun.com/wiki/detail/313