Android点播回放Core SDK2.0集成文档

简介

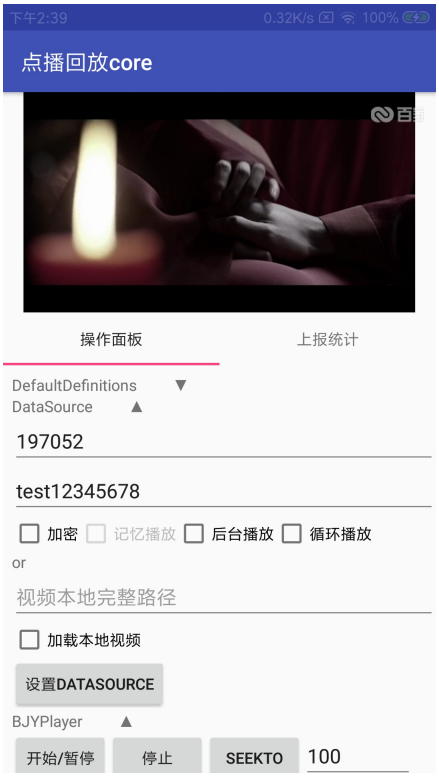
百家云Android点播回放core sdk2.0是一个集点播和回放于一体的无UI纯实现的代码集，点播功能包括在线视频、离线下载、离线播放， 底层采用ijkplayer。回放功能依赖点播播放器之外还包含PPT、聊天消息等模块，百分百还原直播场景，同时也支持离线下载和播放。您可以方便地添加到您自己的app当中。推荐使用Android Studio集成。

功能描述

SDK支持Android4.1（api level 16）及以上

功能	描述
在线播放	支持百家云后台配置视频播放
离线播放	支持本地视频绝对路径，加密和不加密详见视频缓存模块
自定义界面	sdk仅提供原始界面，用户需参考UI SDK自定义实现
视频缓存	支持百家云后台配置视频的缓存功能

示例工程



github链接：http://git.baijiashilian.com/open-android/VideoPlayer2.0Demo(apk在apk_bin目录下)

集成SDK

集成前的准备

1）推荐使用最新版Android studio集成SDK 点击下载（需科学上网）

2）在工程根目录下build.gradle添加远程仓库

```
1. maven { url 'http://git.baijiashilian.com/open-android/maven/raw/master/' }
```

3）在需要集成的module添加如下依赖

```
implemention'com.baijia.player:VideoplayerCore:2.0.0'
```

videoplayer默认依赖supprot包的版本为27，如与本地support版本冲突，可以使用如下集成方式

```
implemention ('com.baijia.player:VideoplayerCore:2.0.0'){
exclude group: 'com.android.support'
}
```

4）在build.gradle中添加ndk过滤

请在过滤so库文件务必同时包含armabi和armeabi-v7a两种类型，否则某些机型上可能出现音视频不同步的情况

```
ndk {
abiFilters 'armeabi-v7a', 'armeabi', 'x86' //x86虚拟机测试用，发版可去掉
}
```

快速集成

1）在Application初始化sdk

```
public class App extends Application {
@Override
public void onCreate() {
super.onCreate();
new BJYPlayerSDK.Builder(this)
.setDevelopMode(true)
.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)
```

点播

初始化播放器

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();

```

IBJYVideoPlayer仅实现播放器相关逻辑，视频展示需要结合surfaceview或textureview进行显示。

初始化视频播放控件

- Java代码初始化

```
BJYPlayerView playerView = findViewById(R.id.activity_new_video_fl);
//设置视频渲染载体，默认textureView
playerView.setRenderType(IRender.RENDER_TYPE_TEXTURE_VIEW);
//设置视频画面裁剪方式，默认16：9
playerView.setAspectRatio(AspectRatio.AspectRatio_16_9);
```

- 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="texture_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）,5.0以上默认使用textureView。可设置视频界面裁剪方式，分别有如下五种： 宽高比16:9

类型	描述
fit_parent_16_9	宽高比16:9

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

播放器绑定视图

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

IBJYVideoPlayer持有BJYPlayerView视图进行视频播放。

绑定视频源

- 传入videoId和token，播放在线视频

```
1. videoPlayer.setupOnlineVideoWithId(Long.parseLong(vid), token);
```

参数说明如下

```
1. /**
2. * 设置播放百家云在线视频
3. *
4. * @param videoId 视频id
5. * @param token 需要集成方后端调用百家云后端的API获取
6. */
7. void setupOnlineVideoWithId(long videoId, String token);
```

- 传入本地视频路径，播放本地视频

```
1. videoPlayer.setupLocalVideoWithFilePath(path);
```

参数说明如下：

```
1. /**
2. * 设置播放本地文件路径
3. *
4. * @param path 视频文件绝对路径
5. */
6. void setupLocalVideoWithFilePath(String path);
```

设置完视频源之后，调用IBJYVideoPlayer.play()即可播放视频。

IBJYVideoPlayer播放器接口清单

```
1. public interface IBJYVideoPlayer extends LifecycleObserver {
2.
3. /**
4. * 绑定PlayerView
5. *
6. * @param view
7. */
8. void bindPlayerView(BJYPlayerView view);
```

```
9.
10. /**
11. * 设置第三方用户信息，用于统计
12. *
13. * @param userName 第三方用户名
14. * @param userIdentity 第三方用户标识
15. */
16. void setUserInfo(String userName, String userIdentity);
17.
18. /**
19. * 设置播放本地文件路径
20. *
21. * @param path 视频文件绝对路径
22. */
23. void setupLocalVideoWithFilePath(String path);
24.
25. /**
26. * 设置播放百家云下载的本地视频
27. *
28. * @param downloadModel 百家云下载的model
29. */
30. void setupLocalVideoWithDownloadModel(DownloadModel downloadModel);
31.
32. /**
33. * 设置播放百家云在线视频
34. *
35. * @param videoId 视频id
36. * @param token 需要集成方后端调用百家云后端的API获取
37. */
38. void setupOnlineVideoWithId(long videoId, String token);
39.
40. /**
41. * 设置播放百家云在线视频
42. * @param videoId 视频id
43. * @param token 需要集成方后端调用百家云后端的API获取
44. * @param accessKey 第三方鉴权(可选)
45. */
46. void setupOnlineVideoWithId(long videoId, String token, String accessKey);
47.
48. void setupOnlineVideoWithVideoItem(VideoItem videoItem);
49.
50. /**
51. * 开始播放
52. */
53. void play();
54.
55. /**
56. * 从startOffset开始播放
57. *
58. * @param startOffset
59. */
60. void play(int startOffset);
```

```
61.
62. /**
63. * 暂停播放
64. */
65. void pause();
66.
67. /**
68. * 是否正在播放
69. *
70. * @return
71. */
72. boolean isPlaying();
73.
74. /**
75. * 停止播放
76. */
77. void stop();
78.
79. /**
80. * 释放播放器
81. */
82. void release();
83.
84. /**
85. * seek
86. *
87. * @param time 时间
88. */
89. void seek(int time);
90.
91. /**
92. * 倍速播放[0.5 ~ 2.0]倍
93. *
94. * @param rate 倍率
95. */
96. void setPlayRate(float rate);
97.
98. /**
99. * 改变清晰度
100. * 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
101. *
102. * @param definition 清晰度
103. * @return true切换清晰度成功 false切换清晰度失败
104. */
105. boolean changeDefinition(VideoDefinition definition);
106.
107. /**
108. * 设置清晰度偏好
109. * 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
110. *
111. * @param definitions 清晰度列表
112. */
113. void setPreferredDefinitions(Iterable<VideoDefinition> definitions);
```

```
113.  void setPreferredDefinitions(Iterable<VideoDefinition> definitions);
114.
115.
116.  /**
117.  * 设置是否支持进入后台时继续播放音频
118.  *
119.  * @param enable boolean 默认false
120.  */
121.  void supportBackgroundAudio(boolean enable);
122.
123.  /**
124.  * 设置是否支持循环播放
125.  *
126.  * @param looping 是否循环播放
127.  */
128.  void supportLooping(boolean looping);
129.
130.  /**
131.  * 设置启用记忆播放
132.  *
133.  * @param context needed by SharedPrefs 做持久化需要用到
134.  */
135.  void enableBreakPointMemory(Context context);
136.
137.  /**
138.  * 设置是否支持广告
139.  *
140.  * @param enable boolean 默认false
141.  */
142.  void supportAdvertisement(boolean enable);
143.
144.  /**
145.  * 设置token失效回调
146.  *
147.  * @param listener token失效回调
148.  */
149.  void setOnTokenInvalidListener(OnTokenInvalidListener listener);
150.
151.  /**
152.  * 设置播放器状态改变回调
153.  *
154.  * @param listener 播放状态改变回调
155.  */
156.  void addOnPlayerStatusChangeListener(OnPlayerStatusChangeListener listener);
157.
158.  /**
159.  * 设置播放进度回调
160.  *
161.  * @param listener 播放进度回调
162.  */
163.  void addOnPlayingTimeChangeListener(OnPlayingTimeChangeListener listener);
164.
165.  /**
```

```
166.  * 设置播放缓存进度回调
167.  *
168.  * @param listener 播放缓存进度回调
169.  */
170. void addOnBufferUpdateListener(OnBufferedUpdateListener listener);
171.
172. /**
173.  * 设置缓冲状态回调
174.  *
175.  * @param listener 缓冲状态回调
176.  */
177. void addOnBufferingListener(OnBufferingListener listener);
178.
179. /**
180.  * 设置播放器出错回调
181.  *
182.  * @param listener 播放器出错回调
183.  */
184. void addOnPlayerErrorListener(OnPlayerErrorListener listener);
185.
186. /**
187.  * 播放上报监听
188.  */
189. void addReportListener(OnPlayerReportListener onPlayerReportListener);
190.
191. /**
192.  * 设置seek结束回调
193.  * @param onSeekCompleteListener
194.  */
195. void addOnSeekCompleteListener(OnSeekCompleteListener onSeekCompleteListener);
196.
197. void rePlay();
198.
199. /**
200.  * 获取播放器状态
201.  *
202.  * @return 播放器状态
203.  */
204. PlayerStatus getPlayerStatus();
205.
206. /**
207.  * get player current play progress.
208.  *
209.  * @return
210.  */
211. int getCurrentPosition();
212.
213. /**
214.  * get video duration
215.  *
216.  * @return
217.  */
```

```
218.  int getDuration();
219.
220.  /**
221.  * get player buffering percentage.
222.  *
223.  * @return
224.  */
225.  int getBufferPercentage();
226.
227.  /**
228.  * 获取播放速率
229.  * @return
230.  */
231.  float getPlayRate();
232.
233.  /**
234.  * 获取当前播放的视频信息，如果通过setupLocalVideoWithFilePath设置数据源则该方法返回null
235.  *
236.  * @return Video Info
237.  */
238.  @Nullable
239.  BJYVideoInfo getVideoInfo();
240.
241.  /**
242.  * 是否播放本地视频
243.  * @return true，播放本地视频
244.  */
245.  boolean isPlayLocalVideo();
246.
247.  /**
248.  * 获取播放器调试相关参数
249.  * @return
250.  */
251.  MediaPlayerDebugInfo getMediaPlayerDebugInfo();
252.  }
```

回放

回放模板依赖前述点播播放器，同时包含一套信令处理逻辑，通过暴露特定接口实现PPT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+信令逻辑=点播+ppt+聊天信息+在线人员信息。 回放的快速集成包含以下几个步骤：

初始化播放器

```
1.  //创建默认播放器实例
2.  IBJYVideoPlayer bjyVideoPlayer = VideoPlayerFactory.createDefaultVideoPlayer();
3.  //播放器实例绑定BJYPlayerView
4.  bjyVideoPlayer.bindPlayerView(playerView);
```

这部分可参看上述点播部分介绍。

初始化回放房间信息

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。这里

需要介绍下长期课的概念，长期课是拥有相同教室id的回放集合，长期课以sessionId来区分各个子课程。

- 创建在线回放PBRoom

1、非长期课回放

```
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);
}
```

2、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);
}
```

- 创建离线回放PBRoom

```
/**
 * 创建回放房间 -- 离线回放, 直接传未解压的文件
 *
 * @param context 上下文
 * @param videoFilePath 视频文件
 * @param signalFilePath 离线信令包文件
 * @return
 */
public static PBRoom newPlayBackRoom(Context context, String videoFilePath, String signalFilePath) {
return new PBRoomImpl(context, videoFilePath, signalFilePath);
}
```

注册播放器状态回调

注册播放器回调同点播部分，可参考上述介绍，自行添加监听器，下面仅注册播放时间变化监听和播放器状态改变监听。

```
bjyVideoPlayer.addOnPlayingTimeChangeListener(new OnPlayingTimeChangeListener() {
@Override
public void onPlayingTimeChange(int currentTime, int duration) {
debugFragment.setPlayerTime("当前时间：" + currentTime + ", 总时间：" + duration);
}
});
bjyVideoPlayer.addOnPlayerStatusChangeListener(new OnPlayerStatusChangeListener() {
@Override
public void onStatusChange(PlayerStatus status) {
debugFragment.setPlayerStatus(status.name());
}
});
```

PBRoom绑定播放器实例

PBRoom通过持有IBJYVideoPlayer实例才能获取到播放器的各种状态回调

```
mRoom.bindPlayer(bjyVideoPlayer);
```

进房间并注册enterRoom回调

```
mRoom.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

回放相关信令监听

绑定PPT

1）WhitebaordView简介

WhitebaordView继承自ImageView,实现PPT显示、画笔绘制、手势缩放。

2)初始化WhiteboardView

同样，这里有Java代码创建和xml声明两种方式，下面介绍xml的方式。

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

```
WhiteboardView whiteboardView = findViewById(R.id.big_container);
//设置ppt背景色
whiteboardView.setBackgroundColor(ContextCompat.getColor(this, R.color.lp_ppt_bg));
```

3）绑定PBRoom

```
whiteboardView.attachPBRoom(mRoom);
```

获取聊天消息

```
disposable = mRoom.getChatVM().getObservableOfNotifyDataChange()
.observeOn(AndroidSchedulers.mainThread())
.subscribe((List<IMessageModel> iMessageModels) -> {
//返回当前时刻的聊天消息集合
messageAdapter.notifyDataSetChanged();
});
```

获取在线人员

```
onlineUserDisposable = mRoom.getOnlineUserVM().getObservableOfOnlineUser()
.observeOn(AndroidSchedulers.mainThread())
.subscribe((List<IUserModel> iUserModels) -> {
//这里返回当前在线人员集合
userAdapter.notifyDataSetChanged();
});
```

退出房间

关闭回放页面，需要退出房间并释放资源

```
//回收PPT相关资源
whiteboardView.destroy();
if(mRoom != null){
//退出房间
mRoom.quitRoom();
}
```

下载

初始化DownloadManager

DownloadMananger使用单例模式，方便获取实例。

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

设置缓存路径

```
manager.setTargetFolder(Environment.getExternalStorageDirectory().getAbsolutePath() +
"/bb_video_downloaded/")
```

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)
```

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));
```

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

回放下载

```
playbackDisposable = downloadManager.newPlaybackDownloadTask("playback", roomId, sessionId, 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:

```
/**
* @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
* @param
*/
public Observable<DownloadTask> newPlaybackDownloadTask(final String fileName, final long roomId, final long sessionId, final String token, final String extraInfo)
```

其它重装方法的参数说明如下：

```
/**
* 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
* @param isEncrypt 加密参数，true为加密 （仅对当前下载有效）
* @return
*/
public Observable<DownloadTask> newPlaybackDownloadTask(String fileName, long roomId, long sessionId, String token, String extraInfo, boolean isEncrypt){
return newPlaybackDownloadTask(fileName, roomId, sessionId, token, extraInfo, isEncrypt, preferredDefinitionList);
}
```

```
/**
* 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
* @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
*/
public Observable<DownloadTask> newPlaybackDownloadTask(final String fileName, final long roomId, final long sessionId, final String token,
final String extraInfo, boolean isEncrypt, List<VideoDefinition> definitionList)
```

下载状态回调

```
downloadTask.setDownloadListener(new DownloadListener() {
@Override
public void onProgress(DownloadTask task) {
//更新下载进度
int progress = (int) (task.getDownloadedLength() / (float) task.getTotalLength() * 100);
}

@Override
public void onError(DownloadTask task, HttpException e) {
//下载出错
Log.e("download", "onError " + e.getCode() + ", message=" + e.getMessage());
}

@Override
public void onPaused(DownloadTask task) {
//暂停
}

@Override
public void onStarted(DownloadTask task) {
//开启下载
}

@Override
public void onFinish(DownloadTask task) {
//下载完成
}

@Override
public void onDeleted(DownloadTask task) {
//删除下载任务及文件
}
});
```

DownloadTask接口清单

```
public interface DownloadTask {

/**
* 开始任务
*/
void start();

/**
* 暂停任务
*/
void pause();

/**
* 重新开始任务
*/
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.  }
```

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。

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

public DownloadModel nextModel;
```

参数说明

PlayerStatus

```
public enum PlayerStatus {

STATE_ERROR,//出错
STATE_IDLE,//未初始化
STATE_INITIALIZED,//初始化
STATE_PREPARED,//已准备好数据
STATE_STARTED,//播放中
STATE_PAUSED,//暂停状态
STATE_STOPPED,//终止状态(已释放播放器实例)
STATE_PLAYBACK_COMPLETED//播放结束
}
```

VideoDefinition

```
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
}
```

错误码

错误码	说明
-10000	ijkplayer内部错误,比如文件异常
-1	无网络连接
-2	移动网络播放
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（需科学上网）

常见问题

1）为什么播本地视频总是提示-10000错误？

-10000错误码为ijkplayer底层报出的错误，请检查本地文件是否为合法视频文件，尝试使用系统播放器能不能正常播放。

2）为什么我集成之后播放视频一直在loading，没有播放出来？

首先检查网络连接是否正常，再者视频渲染采用surfaceview，初始化需要一定的时间，可在调用播放视频方法的时候尝试加入500ms左右延时。

3）为什么一直提示我token解析失败？

一个视频id对应一个token，token是百家云后台生成的，您可以联系后台开发人员确定token是否是正确的，再者查看初始化播放器的时候部署环境是否为在线环境。

4）播放器提示-10000，什么原因导致？

播放器报-10000错误码，可能出现的原因比较多，比如connect timeout IO Error等都会导致播放器提示-10000。这种情况下重试播放即可恢复正常。

5）为什么某些机器上音视频不同步？

请确认同时引入armeabi-v7a和armeabi两种so库。