**IMediaPlayer API 文档**

**1. IMediaPlayer 接口**

由于需求中提到需要保留Android MediaPlayer 以及IjkMediaPlayer，根据IjkPlayer代码，现将这两者抽象为IMediaPlayer，接口如下：

**void** setDisplay(SurfaceHolder sh);

**void** setDataSource(Context context, Uri uri)  
 **throws** IOException, IllegalArgumentException, SecurityException, IllegalStateException;  
  
**void** setDataSource(Context context, Uri uri, Map<String, String> headers)  
 **throws** IOException, IllegalArgumentException, SecurityException, IllegalStateException;  
  
**void** setDataSource(FileDescriptor fd)  
 **throws** IOException, IllegalArgumentException, IllegalStateException;  
  
**void** setDataSource(String path)  
 **throws** IOException, IllegalArgumentException, SecurityException, IllegalStateException;  
  
**void** prepareAsync() **throws** IllegalStateException;  
  
**void** start() **throws** IllegalStateException;  
  
**void** stop() **throws** IllegalStateException;  
  
**void** pause() **throws** IllegalStateException;  
  
**int** getVideoWidth();  
  
**int** getVideoHeight();  
  
**boolean** isPlaying();  
  
**void** seekTo(**long** msec) **throws** IllegalStateException;  
  
**long** getCurrentPosition();  
  
**long** getDuration();  
  
**void** release();  
  
**void** reset();  
  
**void** setVolume(**float** leftVolume, **float** rightVolume);  
  
**int** getAudioSessionId();  
  
**boolean** isPlayable();  
  
**void** setOnPreparedListener(OnPreparedListener listener);  
  
**void** setOnCompletionListener(OnCompletionListener listener);  
  
**void** setOnBufferingUpdateListener(  
 OnBufferingUpdateListener listener);  
  
**void** setOnSeekCompleteListener(  
 OnSeekCompleteListener listener);  
  
**void** setOnVideoSizeChangedListener(  
 OnVideoSizeChangedListener listener);  
  
**void** setOnErrorListener(OnErrorListener listener);  
  
**void** setOnInfoListener(OnInfoListener listener);  
**void** setAudioStreamType(**int** streamtype);  
  
**void** setWakeMode(Context context, **int** mode);  
  
**void** setLooping(**boolean** looping);  
  
**boolean** isLooping();

以上IMediaPlayer接口均从Android MediaPlayer 中提供抽象出来作为Android MediaPlayer 和 IjkMediaPlayer的父类，他们均从IMediaPlayer中衍生实现各自的方法，因此在相关的代码中只要将申明的变量的类型从MediaPlayer改为IMediaPlayer都可以正常编译通过。

其中要注意的两个接口：getCurrentPosition 和 getDuration 他们的返回值均为 long类型

而在Android的MediaPlayer API中为 int类型，在IjkMEdiaPlayer的底层实现是long类型，因此提炼的接口也是long类型，int类型可默认赋予long类型，而long类型不可赋予int类型，除非强制转换，因此使用到这二个接口的时候要注意类型的差异，否则编译会出错

**2. IjkMediaPlayer私有接口**

IjkMediaPlayer 实现了自己的一些私有接口，但是在实际应用中我们并非会用到，所以这里仅介绍会使用到的接口：

**public void** setOption(**int** category, String name, String value)

**public void** setOption(**int** category, String name, **long** value)

这两个接口可用于配置底层 codec 是否使用MediaCodec，Render时的像素格式，

auro- rotate，opensles等等，

具体配置代码如下：

1） 使用MediaCodec

ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"mediacodec"**, 1);

2) 使能H264 H265 Mpegts硬解码（如果不使能H265 Mpegts可能不是硬解）

ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"mediacodec-all-videos"**, 1);

3）设置Auto-rotate

ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"mediacodec-auto-rotate"**, 1);

4）音频opensles优化

ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"opensles"**, 1);

5）opengles2渲染

ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"overlay-format"**, **"fcc-\_es2"**);

6）其他

ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"framedrop"**, 1);  
ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_PLAYER***, **"start-on-prepared"**, 0);  
ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_FORMAT***, **"http-detect-range-support"**, 0);  
ijkMediaPlayer.setOption(IjkMediaPlayer.***OPT\_CATEGORY\_CODEC***, **"skip\_loop\_filter"**, 48);

**3. Buffer Start & End的问题**

IMediaPlayer 提供的**void** setOnInfoListener(OnInfoListener listener);接口用于设置buffer start & end的等的相关监听器，所以apk中要实现OnInfoListener接口，该接口的定义与Android 的MediaPlayer提供的一致，然而，如果你实现的IMediaPlayer的衍生类是AndroidMediaPlayer的话，由于AndroidMediaPlayer使用的是Android系统SDK提供的MediaPlayer，与平台有关，对于buffer start & end有些平台会不上报的问题已经得到共识。使用派生类IjkMediaPlayer是一定可以获得buffer start & end的。

示例代码：

**private** IMediaPlayer.OnInfoListener **mInfoListener** =  
 **new** IMediaPlayer.OnInfoListener() {  
 **public boolean** onInfo(IMediaPlayer mp, **int** arg1, **int** arg2) {

。。。 。。。

**switch** (arg1) {

**case** IMediaPlayer.***MEDIA\_INFO\_BUFFERING\_START***:  
 Log.*d*(***TAG***, **"MEDIA\_INFO\_BUFFERING\_START:"**);  
 **progressBar**.setVisibility(View.***VISIBLE***);  
 **break**;  
 **case** IMediaPlayer.***MEDIA\_INFO\_BUFFERING\_END***:  
 Log.*d*(***TAG***, **"MEDIA\_INFO\_BUFFERING\_END:"**);  
 **progressBar**.setVisibility(View.***GONE***);  
 **break**;

。。。 。。。

}

将mInfoListener通过接口setOnInfoListener设置进去。具体可参考提供的WsVideoTest的代码demo，可见在视频缓冲的时候出现转圈信息，Log也有输出

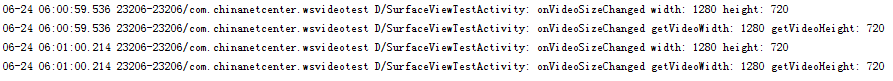
**mediaPlayer**.setOnInfoListener(**mInfoListener**);

**4.视频显示宽高调整**

IMediaPlayer提供的API接口中提供了两个接口分别获取视频的宽高

**int** getVideoWidth();  
**int** getVideoHeight();

从调试的输出结果来看还是比较准确的



从WsVideoTest demo当中，也有提供相应的代码实现全屏还是实际尺寸播放。

**5.对于m3u8视频提供拖动保留最后一帧**

因为m3u8其实是将一个视频切成一个片段一个片段的播放，每个片段都是唯一的url，MediaPlayer播放的时候是重新初始化整个过程，设置url等，这个暂时没想到方法，也不知道拖动的时候啥情况。

**6.IjkMediaPlayer Seek的问题**

在测试对比中，IjkMediaPlayer有一个比较严重的bug，就是seek的时候定位不准确，原因是当你拖动SeekBar的时候，SeekBar的回调会不停的发送seekbar的位置信息，而每次发出的位置信息就会被通过调用IMediaPlayer->seekto函数进行设置传递给底层进行seek，IjkMediaPlayer再底层实现的时候通过变量seek\_req控制seek，seek请求和 seek是不同的线程，而且seek是阻塞的，耗费时间比较长，当第一次seek到达时，seek\_req被置位，实际操作seek的线程检查到seek\_req被置位，就开始seek，由于seek耗时较长，seek完成后seek\_req才会被置位回0值，这时上层发下来的其他请求，由于seek操作未完成，seek\_req仍为置位状态，其他的请求均被丢弃。

seek问题的解决办法：

1）上层只发在最后一次seek

2）底层解决

目前已经从底层解决了这个问题，如果连续发送多次seek操作的时候，底层会进行两次seek操作，第一次和最后一次seek。由于无法取消上一次的seek操作，seek比较耗时，相对来说两次seek可能会久，目前还在想办法优化此方案。

对于seek的问题，最好的方式就是引入可取消的延迟消息队列，对于每次的用户拖动操作都会有一系列的seek请求，如果每次seek请求在加入消息队列的时候，不是理解没执行线程执行，而是有一个延迟，那么后续的seek请求，再加入消息队列前可以检查当前消息队列里面是否已经有seek请求，如果有则删除之，并加入消息队列。这样既可保证用户在进行拖动进度条的时候，底层只进行最后一次seek请求，因为seek是同步的耗时的操作，这样可让seek更快，用户体验更佳，目前seek的延迟时间设为500ms

**7.接口问题**

鉴于我们这边不参与apk的开发，且接口已经讨论后有我们来定，目前，从开发经验来判断。我们这里提供的最简化的接口：

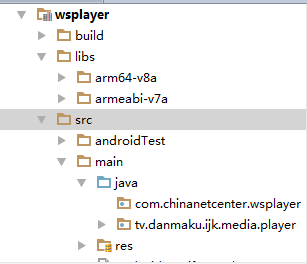
1）提供IMediaPlayer接口，可在代码中完全适配已经编码的Android的 MediaPlayer的接口，可以简单实现代码的迁移，减少工作量

2）不提供类似VideoView等widget，这个可根据提供的IMediaPlayer接口在应用层自己实现（如有需要）

3）最后提供的aar包包含so库和IMediaPlayer接口等

**8.关于WsVideoTest demo**

**WsVideoTest** demo是从IjkPlayer这个项目中抽出来的一个干净的项目，里面包含两个module：WsPlayer 和 WsPlayerTest



**WsPlayer**

该module实现了底层so库和上层的接口IMediaPlayer，最后打包成一个aar

**WsPlayerTest**

这是一个基本的demo，可用于测试**WsPlayer**