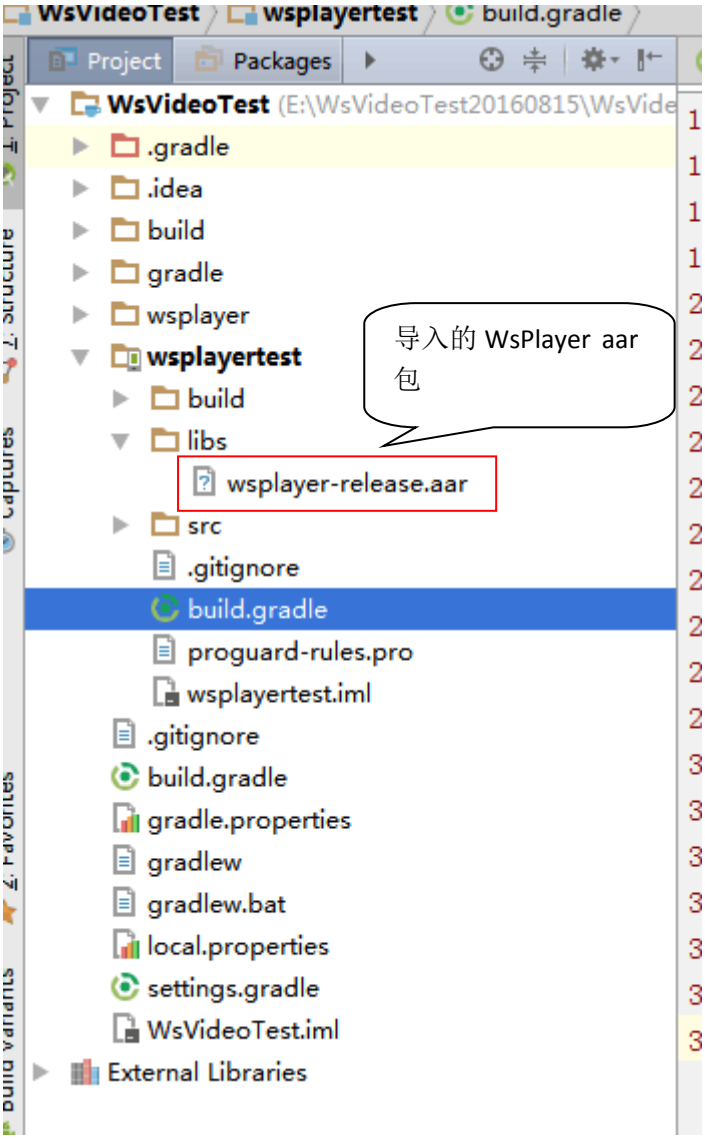
IjkMediaPlayer API 描述文档

日期	版本	修订内容	备注
2016/12/20	V 1.3	增加私有接口	

一 导入 WsPlayer aar

1.1 导入 WsPlayer aar 包

以Android Stdio为开发环境，在Android App开发目录下，按照如图所示的目录结构导入，说明：路径错误会导致加载失败。



1.2 修改 build.gradle 编入 WsPlayer

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'

    compile(name:'wsplayer-release', ext:'aar')

//    compile project(':wsplayer')
}
```

1.3 AndroidManifest 权限设置

确认AndroidManifest包含以下两个权限

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

二 IMediaPlayer API 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：

Public Methods	
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

三 IjkMediaPlayer 的私有接口

3.1 设置播放器内部参数的接口

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 Methods	
void	setOption(int category, String name, String value)
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，
auto-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）其他设置

```
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);
ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "max-fps", -1);
```

3.2 获取播放器内部信息的接口

为了调试方便，播放器对外提供了一些获取调试参数的接口，通过这些接口可以获取播放器的视频的额定帧率，缓冲大小，tcp 传输速度软解码一帧的时间和渲染一帧的时间，以及当前进行同步的音视频的时间戳差值和延迟时间等

Public Methods	
float	getSpeed(**float** speed) 获取播放速率
int	getVideoDecoder() 获取视频的解码信息，硬解或者软解 返回值如下： IjkMediaPlayer.FFP_PROPV_DECODER_AVCODEC == 软解 IjkMediaPlayer.FFP_PROPV_DECODER_MEDIACODEC == 硬解
float	getVideoOutputFramesPerSecond() 获取 IjkMediaPlayer 内部统计的每秒钟的渲染帧数
float	getVideoDecodeFramesPerSecond() 获取 IjkMediaPlayer 内部统计的每秒钟的解码帧数
long	getVideoCachedDuration() 获取视频缓冲的多少时长
long	getAudioCachedDuration() 获取音频缓冲了多少时长
long	getVideoCachedBytes() 获取视频缓冲了多少字节
long	getAudioCachedBytes() 获取音频缓冲了多少字节
long	getVideoCachedPackets() 获取视频缓冲了多少视频包
long	getAudioCachedPackets() 获取音频缓冲了多少音频包
long	getSelectedVideoStream() 获取所选择播放的视频流的 index
long	getSelectedAudioStream() 获取所选择播放的音频流的 index
long	getBitrate() 获取该视频的比特率，目前 m3u8 和 ts 格式的 bitrate 获取仍有问题
long	getTcpSpeed() 获取 Tcp 的传输速度
long	getVideoSoftDecTime() 获取软解一帧视频所耗费的时间
long	getVideoRefreshTime() 获取渲染一帧视频的所耗费的时间
long	getVideoFrameDropEarly() 获取视频解码后检测出赶不上音频时到目前为止丢弃的帧数量
long	getVideoFrameDropLate() 获取在视频渲染时，检测到视频帧已过了渲染时间时到目前为止已经丢弃了的帧数量
float	getAvDelay() 获取当前音视频同步时，视频帧需要的延迟值

float	getAvDiff() 获取音视频同步时，当前音视频时间戳的差值
float	getEfps() 获取视频的额定帧率
float	getTbr() 获取视频的辅助额定帧率

以上几个 IjkMediaPlayer 接口很实用，可以观察同步信息，可以通过软解以及渲染的时间判断盒子的性能，可以查看视频的网络传输速度，缓冲了多少等！建议把这些做成可以隐藏的 debug 窗口，如按某个按键可以显示这些信息！

四 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);
```

五 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 getVideoWPublic Methods	
int	getVideoWidth()
int	getVideoHeight()

从调试的输出结果来看还是比较准确的

```
06-24 06:00:59.536 23206-23206/com.chinanetcenter.wsvideotest D/SurfaceViewTestActivity: onVideoSizeChanged width: 1280 height: 720
06-24 06:00:59.536 23206-23206/com.chinanetcenter.wsvideotest D/SurfaceViewTestActivity: onVideoSizeChanged getVideoWidth: 1280 getVideoHeight: 720
06-24 06:01:00.214 23206-23206/com.chinanetcenter.wsvideotest D/SurfaceViewTestActivity: onVideoSizeChanged width: 1280 height: 720
06-24 06:01:00.214 23206-23206/com.chinanetcenter.wsvideotest D/SurfaceViewTestActivity: onVideoSizeChanged getVideoWidth: 1280 getVideoHeight: 720
```

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。

六 错误码上报

IjkMediaPlayer 在初始化或者播放过程中，可能存在播种原因导致流程不能继续走下去，这时需要上报相应的错误码供上层决定是否需要重试，或者显示给用户，
现将错误码归纳如下：

MP错误标示	MP错误码 arg1	IJK错误标示	IJK 错误码 arg2
MEDIA_ERROR_IO	-1004	EIO	-5
MEDIA_ERROR_UNKNOWN	1	ENOMEM	-12
MEDIA_ERROR_MALFORMED	-1007	N/A	N/A
MEDIA_ERROR_UNSUPPORTED	-1010	N/A	N/A
		ECONNREFUSED	-111
		AVERROR_INVALIDDATA	-1094995529
		ETIMEDOUT	-110
		EIO	-5
		ENETUNREACH	-101
		EHOSTUNREACH	-113
		AVERROR_HTTP_NOT_FC	-875574520
MEDIA_ERROR_TIMED_OUT	-110		

IjkMediaPlayer 内部将会上报错误码两个参数：last_error 为 Android MediaPlayer API 规定的统一参数即上图的 arg1，last_error_2 为 IjkMediaPlayer 内部自定义的参数为上图的 arg2.

```
ffp_notify_msg3(ffp, FFP_MSG_ERROR, last_error, last_error_2);
```

app 层必须实现 IMediaPlayer.OnErrorListener 接口，并将实例传给 setOnErrorListener 函数。

如下：

```
// 错误监听回调函数
mediaPlayer.setOnErrorListener(this);
```

七 wsplayer 使用示例

7.1 声明变量

```
private IMediaPlayer mediaPlayer;
```

7.2 根据需求初始化播放器

```
if (playerText.equals("WsPlayer")) {
    mediaPlayer = new IjkMediaPlayer();
} else {
    mediaPlayer = new AndroidMediaPlayer();
}
// 重置mediaPaly,建议在初始滑mediaplay立即调用。
Log.d(TAG, "########## playVideo ##########");
//mediaPlayer.reset();

if (mediaPlayer instanceof IjkMediaPlayer) {
    initializeIjkMediaPlayer((IjkMediaPlayer)mediaPlayer);
}
```

7.3 IjkMediaPlayer 的参数设置

```
private void initializeIjkMediaPlayer(IjkMediaPlayer ijkMediaPlayer) {
    ijkMediaPlayer.native_setLogLevel(IjkMediaPlayer.IJK_LOG_DEBUG);
    String playerText = switchCodecButton.getText().toString();
    if (playerText.equals("MediaCodec")) {
        ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "mediacodec", 1);
        ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "mediacodec-all-videos", 1);
    }
    ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "mediacodec-auto-rotate", 1);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
        Log.d(TAG, "overlay-format fcc-_es2");
        ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "overlay-format","fcc-_es2");
    } else {
        Log.d(TAG, "overlay-format IjkMediaPlayer.SDL_FCC_RV32");
        ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "overlay-format", //"fcc-i420");
                IjkMediaPlayer.SDL_FCC_RV32);
    }
    ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "framedrop", 1);
    ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "start-on-prepared", 0);
    ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_PLAYER, "max-fps", -1);
    ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_FORMAT, "http-detect-range-support", 1);
    ijkMediaPlayer.setOption(IjkMediaPlayer.OPT_CATEGORY_CODEC, "skip_loop_filter", 48);
```

7.4 在 onPrepared 回调中开始播放

```
mediaPlayer.setDisplay(surfaceHolder);
if (mFirstCreated) {
    changeVideoSize();
    mFirstCreated = false;
}
  if (mediaPlayer instanceof IjkMediaPlayer) {
      ((IjkMediaPlayer)mediaPlayer).selectTrack(1);
  }
// 播放视频
mediaPlayer.start();
```

7.5 上报错误码的操作

```java
@Override
public boolean onError(IMediaPlayer mp, int what, int extra) {
    switch (what) {
        case MediaPlayer.MEDIA_ERROR_UNKNOWN:
            Toast.makeText(this, "MEDIA_ERROR_UNKNOWN (1, " + extra + ")", Toast.LENGTH_SHORT).show();
            break;
        case MediaPlayer.MEDIA_ERROR_SERVER_DIED:
            Toast.makeText(this, "MEDIA_ERROR_SERVER_DIED (100, " + extra + ")", Toast.LENGTH_SHORT).show();
            break;
        case MediaPlayer.MEDIA_ERROR_TIMED_OUT:
            switch(extra) {
                case -111: //Connection refused
                case -1094995529: //Invalid data found when processing input
                    Toast.makeText(this, "Timeout, Media Server error (-110, " + extra + ")" , Toast.LENGTH_SHORT).show();
                    break;
                case -110: //Connection timed out
                case -101: //Network is unreachable
                case -5: //I/O error
                case -113: //No route to host
                    Toast.makeText(this, "Timeout, Network Disconnection (-110, " + extra + ") ", Toast.LENGTH_SHORT).show();
                    break;
                default:
                    Toast.makeText(this, "Timeout, unknown error (-110, " + extra + ")" , Toast.LENGTH_SHORT).show();
            }
            return reconnectVideo();
        case MediaPlayer.MEDIA_ERROR_IO:
            Toast.makeText(this, "MEDIA_ERROR_IO (-1004, " + extra + ")", Toast.LENGTH_SHORT).show();
```

八 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