锐动 Android 手机直播 SDK

作者：RD Android Team

版本：Beta 版

日期：2017-3-8

1 编写目的

预期读者：

有视频开发经验或者无经验的，打算或者正在使用“锐动 Android 视频直播 SDK”的相关工程师。

- Android 软件工程师。
- 产品经理。
- QA

2 名词解释

- 分辨率：用于计算机视频处理的图像，以水平和垂直方向上所能显示的像素数来表示分辨率。常见视频分辨率的有 1080P 即 1920x1080，720P 即 1080x720，640x480 等。
- 帧率：每秒的帧数(fps)或者说帧率表示图形处理器处理场时每秒钟能够更新的次数。
- 码率： 数据传输时单位时间传送的数据位数,一般我们用的单位是 kbps 即千位每秒。

3 申请 APPKey 和 APPSecretkey

1、 登录 http://www.rdsdk.com/注册用户

2、 登录注册好的用户

3、 进入视频云管理 点击（新增）获取应用的 appkey 、appsecret

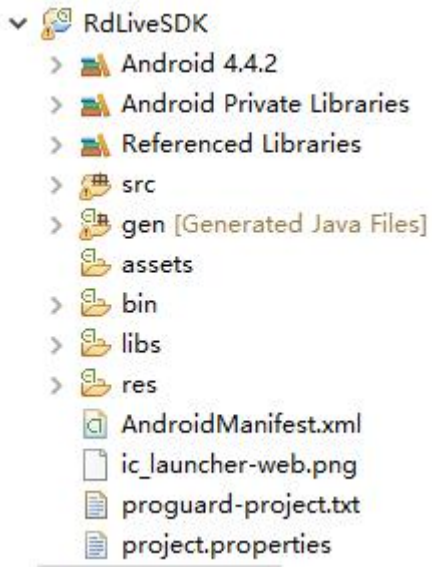
3 Eclipse 集成步骤

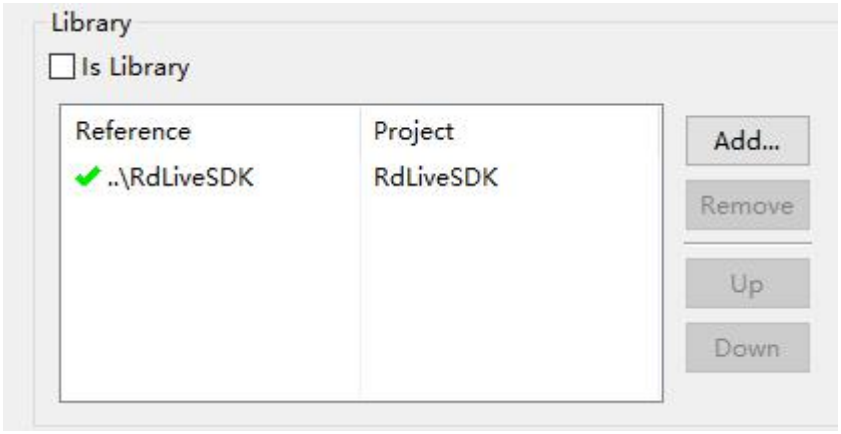
3.1 运行环境

- Android 4.1（api 16）以上；
- 处理器：双核 1GHz 以上 CPU(目前只支持 ARM CPU, X86、MIPS 暂不支持)；
  推荐四核 1.2GHz 以上 CPU
- 内存：1 GB 以上；

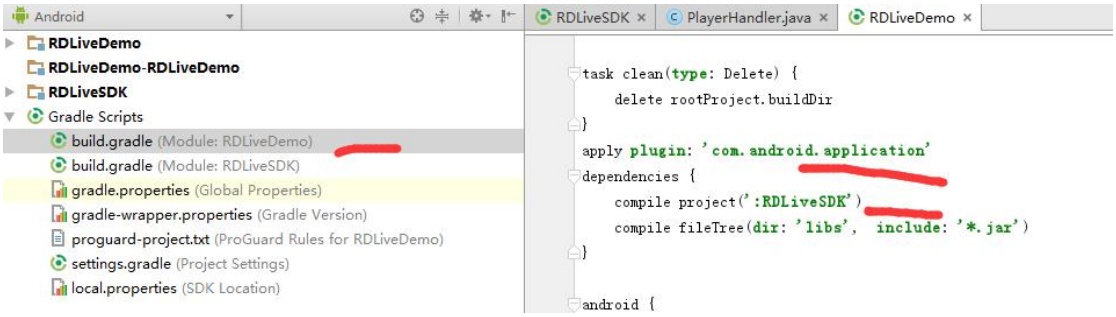
3.2 下载并导入 SDK

3.2.1 导入 RdLiveSDK





3.2.2 Studio build.gradle 配置如下:

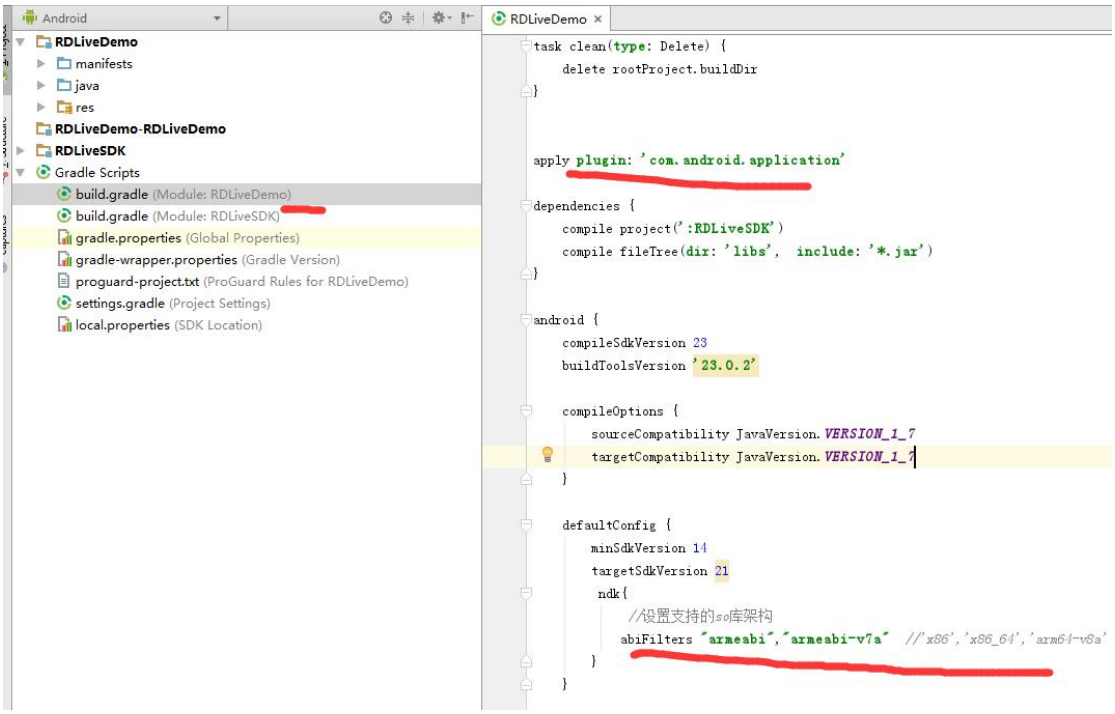


关于 arm 版本引发的冲突解决办法如下:(重要)

步骤一:sdk library 模块新增过滤

```
ındroid {
  compileSdkVersion 19
  buildToolsVersion "23.0.2"
   packagingOptions {
     //过滤掉本地libs/armeabi/和libs/armeabi-v7a/中的部分so，以jcenter
     exclude 'lib/armeabi/libRdBase.so'
     exclude 'lib/armeabi/libLiveRecorder.so'
     exclude 'lib/armeabi/libRecorderKernel.so'
     exclude 'lib/armeabi-v7a/libRdBase.so'
     exclude 'lib/armeabi-v7a/libLiveRecorder.so'
     exclude 'lib/armeabi-v7a/libRecorderKernel.so'
  }
```

步骤二:application 模块中过滤 （很重要）



3.3 准备 AndroidManifest.xml （权限，注册 Activity)

a.添加权限：

```
<!-- 必要的权限 -->
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.RECEIVE_USER_PRESENT" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.BATTERY_STATS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

设备支持的版本号

```
<uses-sdk
  android:minSdkVersion="14"
  android:targetSdkVersion="22" />
```

android:targetSdkVersion="23' Androia 6.0请遵循请求权限规范 申请以下权限再调用之前

<uses-permission android:name="android.permission.CAMERA' />

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

如果不想影响当前项目的 AndroidManifest.xml，可以合并 RdLiveSDK 中的配置，修改 project.properties 文件如下：

```
android.library.reference.1=../RdLiveSDK
manifestmerger.enabled=true
```

4 SDK 初始化

*AppImp.java

```
1  package com.example.rdlivedemo;
2
3  import android.app.Application;
7  public class AppImp extends Application {
8      /**
9       *锐动云控平台申请的appKey,Secret
10      *http://dianbook.17rd.com/business/app/myapp
11      */
12     private final String APPKEY = "631c39cb35047e92";
13     private final String APPSECRET = "be9d6d860c077aef2
14     @Override
15     public void onCreate() {
16         super.onCreate();
17         // 初始appkey和appsecret
18         RdLiveSDK.init(this, APPKEY, APPSECRET);
19     }
20 }
```

5 关于 SDK 权限

5.1 获取权限

```
/**
 *
 * -1代表Uid与Url直播方式都不可用 {@link RDLiveSDK#AT_INVALID}<br>
 * 2代表仅支持Uid直播方式 {@link RDLiveSDK#AT_UID}<br>
 * 3代表仅支持Url直播方式 {@link RDLiveSDK#AT_URL}<br>
 * 1代表两种直播方式都支持 {@link RDLiveSDK#AT_URL_OR_UID}<br>
 *
 * @return 返回当前SDK支持的直播方式
 */
```

```
RDLiveSDK.getAuthType();
```

6 直播(确保步骤 4 已经初始化)

6.0 初始化直播环境

初始化当前直播环境 。推荐在 startActivity 调用,原因:部分手机省电模式下从后台切回前台会新开进程

```
RDLiveSDK.onInit(context);
```

6.1 直播初始化参数

```
// 直播画面打开之前配置直播参数(可包含摄像头、输出尺寸、帧码率、是否美颜等)
// 通过此方法设置摄像头方向、是否美颜只在 onprepared(RelativeLayout,listener)之前调用有效
//初始化之后设置前后置方向使用 RDLiveSDK.switchCamera()、RDLiveSDK.enableBeautify(bEnableBeautify)
RDLiveSDK.setEncoderConfig(LiveConfig);
```

6.2 初始化直播界面

```
/**
 * 准备 SDK
 * @param rlParent
 *           用于显示摄像头的父布局
 * @param iListener
 *           SDK 消息 Listener
 */
RdLiveSDK.onPrepare(m_pflPreviewLayout, iListener);
```

一般用在 Activity.onStart()中

6.3 关于直播摄像头的释放

```
//用于响应 Activity.onDestroy()
   RdLiveSDK.onExit(this);
```

6.4 开始、结束直播

```
    /**
     * * 开始直播
     * @param uidORtmp
     *            集成两种直播方式(uid 或直播推流地址)
     * @param title
     *            直接推流方式 title 可传 null,否则 title 不能为 null
     */
RDLiveSDK.startPublish(uidORtmp, title);
//结束直播
RdLiveSDK.stopPublish();
//是否正在直播...
RdLiveSDK.isLiving();
```

6.5 关于回调接口

```
    /**
     * 打开摄像头成功并且开始预览界面
     * @param nResult
     *            返回值 >={@link ResultConstants#SUCCESS} 代表成功，否则为失败
     * @param strResultInfo
     *            具体返回消息
     */
     //打开摄像头成功并且开始预览界面
IRecorderListener.onPrepared(int nResult, String strResultInfo);
    /**
     * 响应直播开始
     * @param nResult
     *            返回值 >={@link ResultConstants#SUCCESS} 代表成功，否则为失败
```

```
     * @param strResultInfo
     *          具体返回消息
     */
```

//直播成功的回调

```
IRecorderListener.onRecordBegin(int nResult, String strResultInfo);
    /**
     * * 响应获取已录制时间
     * @param nPosition 已录制时间 (ms)
     * @param nRecordFPS 录制帧率
     * @param delayed 延迟时间
     */
```

//获取直播信息，如已直播时长和直播帧率、直播延迟

```
IRecorderListener.onGetRecordStatus(int nPosition, int nRecordFPS, int delayed);
    /**
     * 响应直播结束
     * @param nResult
     *          返回值 >={@link ResultConstants#SUCCESS} 代表成功，否则为失败
     * @param strResultInfo
     */
IRecorderListener.onRecordEnd(int nResult, String strResultInfo);
    /**
     * 录制出现错误
     * @param nResult
     *          返回值
     * @param strResultInfo
     *          具体返回消息
     */
IRecorderListener.onRecordFailed(int nResult, String strResultInfo);
    /**
     * 响应摄像头打开信息
     * @param nResult
     *          返回值 >={@link ResultConstants#SUCCESS} 代表成功，否则为失败
     * @param strResultInfo
     *          具体返回消息
     */
    //相机打开失败，权限不足
IRecorderListener.onCamera(int nResult, String strResultInfo);
  //新增推流重连逻辑接口
    /**
     * @param nResult
```

```
     *           {@link ResultConstants#RECONNECTION_PREPARE} 开始重连
     *           {@link ResultConstants#RECONNECTION_SUCCESS} 重连成功
     *           {@link ResultConstants#ERROR_RECONNECTION_TIMEOUT} 重连超时
     * @param msg
     *          文本内容
     */
```

IRecorderListener.onReconnection(int nResult, String strResultInfo);

//Android 6.0 请在调用 onprepare 前确保存储、相机、录音机权限均已授权成功;否则请打开权限

```
@Override
public void onPermissionFailed(int nResult, String strResultInfo) {
    if (nResult == ResultConstants.PERMISSION_FAILED) {
        Log.e(TAG, "onPermissionFailed->" + strResultInfo);
        finish();
    }
};
```

6.6 直播关于摄像头相关(必须在 onprepare 回调之后才有效)

```
//聚焦
RdLiveSDK.cameraAutoFocus();
// 是否为前摄像像头
RdLiveSDK.isFaceFront();
// 切换摄像头
RdLiveSDK.switchCamera();
```

6.7 闪光灯(必须在 onprepare 回调之后才有效)

```
//获取闪光灯状态
RdLiveSDK.getFlashMode();
//打开闪光灯
RdLiveSDK.setFlashMode(boolean enable);
```

6.8 美颜

```
//是否支持美颜
RdLiveSDK.isBeautifyEnabled();
//打开或关闭美颜
RdLiveSDK.enableBeautify(boolean bEnableBeautify);
```

6.9 截图 (必须在 listener.onprepared 回调之后才有效)

```
    /**
     * 当前画面截图
     * @param path
     *           截图文件路径
     * @param width
     *           截图宽度
     * @param height
     *           截图高度
     */
RdLiveSDK.screenshot(path, 480, 640);
    /**
     * 响应保存截图
     * @param nResult
     *           返回值 =={@link ResultConstants#SUCCESS} 代表成功，否则为失败
     * @param msg
     *
     */
IRecorderListener.onScreenShot(int nResult, String path);
```

6.10 静音

```
//设置静音
RDLiveSDK.setMute(boolean mute);

//静音状态
RDLiveSDK.isMute();
```

6.11 设置推流重连超时

//设置推流重连超时(单位：毫秒)该方法在开始直播之前调用

RDLiveSDK.setReconnectionTimeOut(int nTimeOut)

6.12 设置推流超时

//设置推流超时时间(1000 到 30000 毫秒之间)

RDLiveSDK.setApiLiveSetRtmpUploadPacketTimeout(int ntimeOut);

6.13 开启低配美颜(美颜需 Android4.3 及以上)

//该方法针对 4.3 以上，低配置机器（不常用）

```
RDLiveSDK.enableDeviceLowBeautify(boolean enable);
```

6.14 获取当前低配美颜状态

RDLiveSDK.isDeviceLowBeautify();

6.15 设置美颜等级

//1-5 数字越大等级越高

```
RDLiveSDK.setBeautifyLevel(int level);
```

6.16 获取美颜等级

```
Int level=RDLiveSDK.getBeautifyLevel();
```

6.17 直播前台切到后台

```
/**
 * 程序即将切到后台，保存sdk内部推流状态
 */
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    RDLiveSDK.onSaveInstanceState(this);
}

@Override
protected void onStop() {
    super.onStop();
    mhandler.removeCallbacks(repushRunnable);
    if (!doExitLiving) {
        // App切到后台,暂停推流
        RDLiveSDK.pausePublish();
    }
    Log.i("onstop->removeCallbacks", this.toString());
}
```

6.18 直播后台切回前台

在 Activity.onStart()中调用

1.获取直播是否暂停的状态

获取是否暂停状态， pausePublish()--->onRestoreInstanceState() 场景:从直播切到后台，到未继续推流之前 为 true;且他时间段为 false

```
boolean isLivePaused = RDLiveSDK.isLivePaused(this);
Log.e("响应UI调整", "直播是否在暂停中：" + isLivePaused);
```

2 继续推流

```
/**
 * 检测之前是否完全退出，若未完全退出，继续直播
 */
isLivingUI = RDLiveSDK.onRestoreInstanceState(this);
```

isLivingUI :true 继续推流,sdk 将继续直播

6.19 水印相关

//开启水印

RDLiveSDK.registerOSDBuilder(DemoOSDBuilder.class);

//关闭水印

RDLiveSDK.registerOSDBuilder(null);

```
Public class DemoOSDBuilder extends OSDBuilder{
      //设置水印位置
      setOSDGravity(int nGravity);
    //刷新水印内容
     protected void onRefreshOSDView(View vOSD) {}
}
```

注混淆：

```
-keepclassmembers class * extends com.rd.recorder.OSDBuilder{
    *;
}
```

6.20 在线直播列表

```
RDLiveSDK.getLivingList(new ILivingCallBack() {

    @Override
    public void getLivingList(ArrayList<String> list) {
        if(null!=list){
            int len=list.size();
            for (int i = 0; i < len; i++) {
                Log.i("---"+i, "uid-->"+list.get(i));
            }
        }
        onToast((null == list) ? "没有在线主播"
                : ("在线人数" + list.size()));
    }
});
```

6.21 横竖屏录制

说明:输出视频的录制方向，内部通过获取屏幕方向得到

//开始直播录制

第一步：锁定屏幕当前的方向(固定输出方向) ，录制结束恢复自动旋转

第二步：设置输出推流的输出尺寸，推流尺寸统一参照竖屏（sdk 内部取了屏幕旋转值）

第三步：执行推流

```
<activity
    android:name="com.example.rdlivedemo.LiveActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:screenOrientation="sensor"
    android:theme="@style/Theme.rd.fullscreen"
    android:windowSoftInputMode="stateAlwaysHidden|adjustPan" />
```

//录制前,获取当前屏幕的方向 0、90、180、270 ， 并锁定当前方向即确认录制输出的方向

```
// 确认当前屏幕方向锁定屏幕
private void onLockScreen() {
    int displayRotation = RDLiveSDK.getDisplayRotation(this);
    if (0 == displayRotation) {// 标准竖屏
        lastOrientation = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT;
        setRequestedOrientation(lastOrientation);
    } else if (90 == displayRotation) {// 标准横屏
        lastOrientation = ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE;
        setRequestedOrientation(lastOrientation);
    } else if (180 == displayRotation) {// 反向竖屏
        lastOrientation = ActivityInfo.SCREEN_ORIENTATION_REVERSE_PORTRAIT;
        setRequestedOrientation(lastOrientation);
    } else if (270 == displayRotation) {// 反向横屏
        lastOrientation = ActivityInfo.SCREEN_ORIENTATION_REVERSE_LANDSCAPE;
        setRequestedOrientation(lastOrientation);
    }
}
```

//从后台切回前台时,需要重新锁定屏幕方向

```
onSureOrientation(lastOrientation);
```

6.22 自定义推流数据(4.3 以上的设备支持)

第一步:检测是否支持该功能:

RDLiveSDK.enableCustomData()

第二步:实现接口

com.rd.recorder.ICustomData

设置接口实现:

```
@Override
public void onPrepared(int nResult, String strResultInfo)
    onToast("onPrepared->" + "初始化成功" + strResultInfo);
    RDLiveSDK.setCameraZoomHandler(m_hlrCameraZoom);
    //设置自定义推流数据
    if (RDLiveSDK.enableCustomData()) {
        RDLiveSDK.setICustomData(iCustomdata);
    }
```

开启或关闭自定义推流:

```
/**
 *sdk 显示要自定义的图片
 *bmp==null ，即关闭自定义;bmp!=null ,即为自定义推流
 */
@Override
public Bitmap getBmp() {
    synchronized (this) {
        return bmp;
    }
}
```

//混淆

#自定义推流

-keep class * implements com.rd.recorder.ICustomData {*;}

6.23 混音播放器

```
player = new AudioPlayer();
```

(只支持播放本地音乐,支持的音频格式 mp3、mp2、 aac、 wma、 wmv、 ac3、 ogg)

此播放器支持混音功能.场景:插上耳机，传输一路音频流到看直播端

```
player.setOnPreparedListener(listener);

player.setOnInfoListener(infolistener);

player.setOnCompletionListener(completlistener);

player.setOnErrorListener(onErrorListener);

player.prepareAsync();
```

6.24 是否开启混音功能

//设置是否开启混音

```
RDLiveSDK.enableMixAudio(isChecked);
```

//获取当前是否开启混音

```
RDLiveSDK.isEnableMixAudio()
```

6.25 设置混音占比

//设置混音占比 0-100

```
RDLiveSDK.setMixFactor(progress);
```

//获取当前混音占比

```
RDLiveSDK.getMixFactor()
```

6.26 滤镜

//获取支持的滤镜 在 listener.onprepared() 回调成功之后才能获取支持的滤镜

```
effects = RDLiveSDK.getSupportedColorEffects();
```

//设置滤镜

```
RDLiveSDK.setColorEffect(color);
```

6.27 人脸面具

//说明:sdk 内置了几个遮罩面具，须写入到 sd 目录才能使用

//android 4.3 及以上支持 listener.onprepared()回调成功之后操作人脸面具才生效

//第一步:判断是否支持该功能

```
RDLiveSDK.isSupportFaceFlash();
```

//判断该功能是否打开

```
RDLiveSDK.isFaceFlashEnabled();
```

//第二步:启用该功能(flase 关闭该功能)

```
RDLiveSDK.enableFaceFlash(true);
```

//第三步:设置本地人脸面具的路径

```
RDLiveSDK.reloadFaceFlash(info.getSdPath());
```

7 看直播（支持 UID，(RTMP、RTSP 两种方式处理逻辑一样)）

7.1 在布局文件中编写直播视频播放控件 通过控件 id 找到控件

```
<com.rd.live.ijk.RdLivePlayer
    android:id="@+id/vv_play"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

RdLivePlayer    vvPlay = (RdLivePlayer) findViewById(R.id.vv_play);
```

7.2 检测主播是否退出房间

```
vvPlay.setOnCompletionListener(new IMediaPlayer.OnCompletionListener() {

    @Override
    public void onCompletion(IMediaPlayer mp) {
        // Log.e("complet....", "wanbi....." + mp.getDuration());
        if (playTag != null && playTag == LiveType.mp4) {
            onComplete();
            vvPlay.removeCallbacks(m_getPlayProgressRunnable);
        } else {
            onExitLiving("主播已退出房间");
        }

    }
});
```

7.3.1 Uid 查看直播

//通过 Uid 获取直播流(参考 demo 中 MainActivity# R.id.mBtnPlayLiving)

RDLiveSDK.getLivingUid(String uid, RDLiveSDK.ILivingListener listener)

//可以直播流的地址(rtmp://) 不知道直播 liveId 可传 null(liveId 由 listener 返回)

vvPlay.setVideoPath("<您要观看的视频 rtmp>",liveId);

7.3.2 Url 查看直播

//可以直播流的地址(rtmp://)

vvPlay.setVideoPath("<您要观看的视频 rtmp>");

7.4 播放器设置

* 是否开启全屏显示(内容会被裁减)默认全屏

true 全屏显示部分内容会被裁剪;false 显示全部全部视频内容

vvPlay.enableFullScreen(boolean enable);

7.5 查看播放器预览模式是否全屏

vvPlay.isFullScreen();

7.6 播放器回调

// IMediaPlayer.ANCHOR_PASUING 主播离开(只针对观看 UID 直播)

//what == IMediaPlayer.MEDIA_INFO_BUFFERING_START (网络视频，开始缓冲)

//what == IMediaPlayer.MEDIA_INFO_BUFFERING_END(缓冲结束)

//what == IMediaPlayer.MEDIA_INFO_AUDIO_RENDERING_START (vvPlay.start()，开始音频回调,播放器接收到画面，可隐藏遮罩部分(防止一开始黑屏))

```
vvPlay.setOnInfoListener(new IMediaPlayer.OnInfoListener() {

    int last = 0;

    @Override
    public boolean onInfo(IMediaPlayer mp, int what, int extra) {
        // Log.e("info....", what + "...." + extra);
        if (what == IMediaPlayer.ANCHOR_PASUING) {// 主播离开的特别标识
            SysAlertDialog.cancelLoadingDialog();
            if (null != flIndicator
                    && flIndicator.getVisibility() != View.VISIBLE) {
                flIndicator.setVisibility(View.VISIBLE);
            }

        } else if (what == IMediaPlayer.MEDIA_INFO_BUFFERING_START) {
```

//onError 回调监听主播退出

//what==IMediaPlayer.ANCHOR_EXIT 主播退出

```
vvPlay.setOnErrorListener(new IMediaPlayer.OnErrorListener() {

    @Override
    public boolean onError(IMediaPlayer mp, int what, int extra) {
        Log.e(this.toString(), "setOnErrorListener...onError." + what
                + "...." + extra);
        if (null != flIndicator
                && flIndicator.getVisibility() == View.VISIBLE) {
            flIndicator.setVisibility(View.GONE);
        }
        if (what == -IMediaPlayer.ANCHOR_EXIT
                && !(playTag == LiveType.mp4)) {
```

8 打包混淆

需要在 proguard.cfg 文件中添加如下配置项：

```
-dontwarn    com.rd.**

-keep class com.rd.** { *; }

-dontwarn tv.danmaku.**

-keep class tv.danmaku.** { *; }

#水印

-keepclassmembers class * extends com.rd.recorder.OSDBuilder{

       * ;

}

#自定义推流

-keep class * implements  com.rd.recorder.ICustomData {*;}
```