视跃科技
SHIYUE TECHNOLOGY

武汉视跃科技有限公司

gb/t28181 设备端 SDK(android 版)

接口说明

2019 年 9 月

视跃科技 SHIYUE TECHNOLOGY

文件变更记录

*A - 增加 M - 修订 D - 删除

版本号	日期	变更类型（A*M*D）	修改人	摘　　要	审核人	备注
V-0.1	2019-09-07	A	王杰	新建文档	王杰	
V-0.2	2020-03-12	A	张杰	新增 JNI 接口说明	王杰	
V-0.3	2020-06-20	A	李晓峰	增加通道设置接口说明	王杰	
V-0.4	2020-10-11	A	王杰	增加语音对讲的穿透模式	王杰	
V-0.5	2021-03-11	A	钱锋	增加纹理采集显示	李晓峰	
V-0.6	2021-09-12	A	王杰	增加集群对讲接口	王杰	
V-0.7	2021-12-28	A	王杰	增加 otg 外接 usb 摄像头支持	王杰	
V-1.0	2022-07-11	A	王杰	增加外部设置开发者账号和密码，优化消息反射模式	王杰	
V-2.0	2023-3-12	A	王杰	增加了 2.0 版接口	王杰	

视跃科技 SHIYUE TECHNOLOGY

目录

1 sdk 使用说明

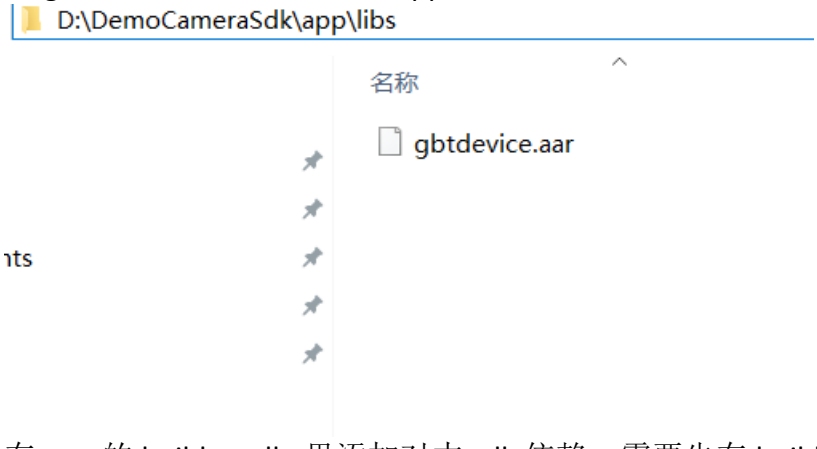
本 sdk 底层采用 native c++编写，由应用层（java 层）和 jni 层（native c++）组成，jni 层提供的是基础的功能 api 函数。应用层对其进行封装，并结合 android 原生设备相关的 sdk 功能对其进行了功能的组合与协作，实现了功能全面的 gbt 设备端应用场景，本文档提供的是应用层 sdk。

视跃科技 SHIYUE TECHNOLOGY

1.1sdk 的 aar 包的引入

Sdk 以 module 形式提供，为文件名为 gbtdevice.aar 的 aar 包，使用方法如下：

1. 将 gbtdevice.aar 文件拷贝在 app 下的 libs 文件夹里，如下图：



2. 在 app 的 build.gradle 里添加对本 sdk 依赖，需要先在 build.gradle 里增加：

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

同时，在 dependencies 里增加如下一行：

```
implementation(name:'gbtdevice', ext:'aar')
```

最终 build.gradle 如下图（红色框为增加部分）：

```
repositories {
    flatDir {
        dirs 'libs'
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'androidx.vectordrawable:vectordrawable:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.0'
    implementation(name:'gbtdevice', ext:'aar')
}
```

这样即完成了 app 对本 sdk 的引入。

1.2sdk 授权的引入

购买 sdk 后，会分配一个 sdk 的账号和密码，通过该账号和密码可以登录视跃自动授权系统 http://gbs.realgbs.com/,进入该系统后可以修改默认的密码。同时账号和密

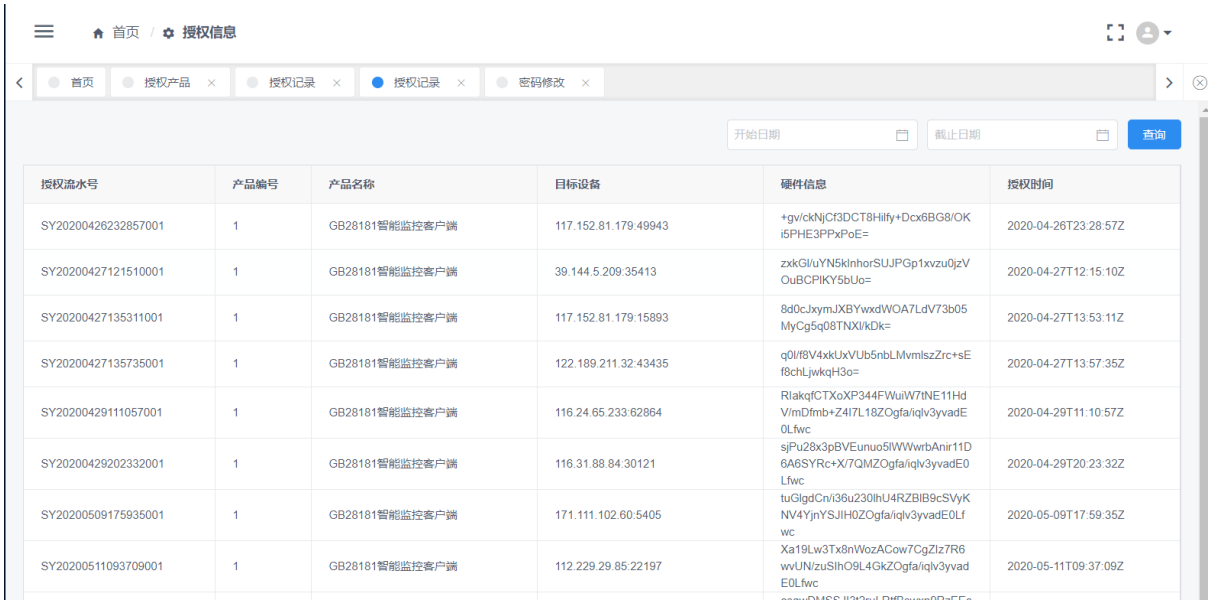
码作为开发者的 apike 信息，在基于本 sdk 开发 app 时，将其以 mete-data 元素引入 app 的 AndroidManifest.xml 的 application 标签内，如下图（1.2-1）：

```
<meta-data android:name="apikey" android:value="test" />
<meta-data android:name="secret" android:value="\12345678" />
```

图 1.2-1

其中 apikey 为用户名，secret 为密码，如果密码为纯数字，请在前面添加转义符”\”。

注意：SDK 的账号密码请严密保管，引用该账号和密码，生成的 sdk 在新的设备上运行时，会扣减一次 license 数量，并生成授权记录供查询。如下图（1.2-2）：



授权流水号	产品编号	产品名称	目标设备	硬件信息	授权时间
SY20200426232857001	1	GB28181智能监控客户端	117.152.81.179:49943	+gv/ckNjCf3DCT8Hilfy+Dcx6BG8/OKi5PHE3PPxPoE=	2020-04-26T23:28:57Z
SY20200427121510001	1	GB28181智能监控客户端	39.144.5.209:35413	zxkGl/uYN5kInhorSUJPGp1xvzu0jzVOuBCPlKY5bUo=	2020-04-27T12:15:10Z
SY20200427135311001	1	GB28181智能监控客户端	117.152.81.179:15893	8d0cJxymJXBYwxdWOA7LdV73b05MyCg5q08TNXl/kDk=	2020-04-27T13:53:11Z
SY20200427135735001	1	GB28181智能监控客户端	122.189.211.32:43435	q0l/f8V4xkUxVUb5nbLMvmlszZrc+sEf8chLjwkqH3o=	2020-04-27T13:57:35Z
SY20200429111057001	1	GB28181智能监控客户端	116.24.65.233:62864	RIakqfCTXoXP344FWuiW7tNE11HdV/mDfmb+Z4I7L18ZOgfa/iqlv3yvadE0Lfwc	2020-04-29T11:10:57Z
SY20200429202332001	1	GB28181智能监控客户端	116.31.88.84:30121	sjPu28x3pBVEunuo5lWWwrbAnir11D6A6SYRc+X/7QMZOgfa/iqlv3yvadE0Lfwc	2020-04-29T20:23:32Z
SY20200509175935001	1	GB28181智能监控客户端	171.111.102.60:5405	tuGlgdCn/i36u230lhU4RZBIB9cSVyKNV4YjnYSJIH0ZOgfa/iqlv3yvadE0Lfwc	2020-05-09T17:59:35Z
SY20200511093709001	1	GB28181智能监控客户端	112.229.29.85:22197	Xa19Lw3Tx8nWozACow7CgZlz7R6wvUN/zuSIhO9L4GkZOgfa/iqlv3yvadE0Lfwc	2020-05-11T09:37:09Z

图 1.2-2

1.3 针对局域网加密设备授权方法

对于因为加密场景需要，设备自始至终都无法连外网进行首次自动授权的用户可以采用此方法。运行未授权的 DEMO app（devicecamera.app）,如该设备未授权，则会在设备外置存储目录下(通常为 sdcard 目录下生产该设备的序列号文件 RegisterNumber.c2v)，将 RegisterNumber.c2v 文件发给我司技术支持人员即可完成授权。

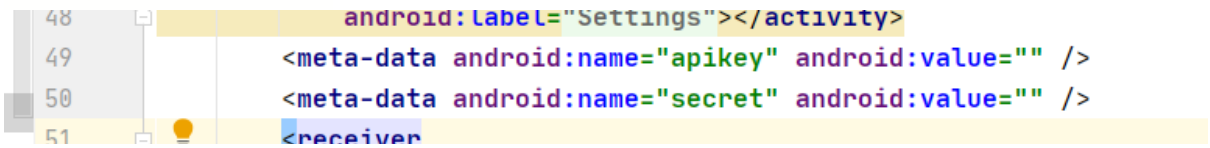
1.4 动态修改授权信息

除了通过 1.2 描述的静态的从 android 的 AndroidManifest.xml 中修改引入授权信息外，新增加了通过接口函数实现设置授权信息。其接口为

SYGbtCommon.*setapikey 函数，示例代码为：*

```
import com.shiyue.comm.SYGbtCommon;
SYGbtCommon.setapikey(this,"test","123456");
```

注意：还是调用此接口还是需要在 AndroidManifest.xml 中添加响应的 metadata，只是 value 可以为空，如下：



2. GB/T 28181 设备类接口

类 CGbtMedia 是核心的应用类，实现了设备端 gb28181 的所有功能，使用时要引入改类，import com.shiyue.media.CGbtMedia;

2.1 设置本设备端的多媒体格式

接口原型：`public boolean setFormat(MFormat fmt)`
接口参数：

fmt（in）	设置的多媒体格式，关于类型 MFormat 说明见本节下面
返回参数	False/true -- 失败/成功

接口说明: 本函数设置设备的音视频输出格式。MFormat 为多媒体格式结构体，其定义如下：

```
public class MFormat {

    public int codec;       // 视频编码
    public int width;       // 视频高
    public int height;      // 视频宽
    public int fps;         // 帧率
    public int clockRate;   // 视频时钟

    public int bitrate;     //视频编码码率
    public int gop;         // 关键帧间隔，不设置默认为 1s

    public int audioCodec;  // 音频编码
    public int channels;    // 通道数
    public int sampleRate;  // 采样率
public int audioRate;   // 音频时钟

}
```

其中 codec 对应的视频编码，其取值范围为：

属性名称	类型	说明
MCODEC_H264	int	视频采用 H264 编码
MCODEC_HEVC	int	视频采用 H265 编码

audioCodec 对应的音频编码，其取值范围为：

属性名称	类型	说明
MCODEC_G711A	int	音频采用 G711A 编码
MCODEC_G711U	int	音频采用 G711U 编码
MCODEC_AAC	int	音频采用 AAC 编码

创建多媒体结构对象示例代码：

```
MFormat fmt = new MFormat();
fmt.width = 640;
fmt.height = 480;
fmt.codec = MediaType.MCODEC_H264;
fmt.fps = 25;
fmt.gop = 1;
fmt.bitrate = 1024*1024;

fmt.audioCodec = MediaType.MCODEC_G711U;
fmt.channels = 1;
fmt.sampleRate = 8000;
fmt.clockRate = 1000000;
fmt.audioRate = 1000000;
```

2.2 设置字幕叠加

接口原型： public boolean setOsd(int index, OsdOverlay overlay)

接口参数：

index（in）	叠加字幕的序号（sdk 支持多层字幕叠加，通过该参数确定那层字幕）
overlay（in）	叠加字幕的结构体，关于 OsdOverlay 说明见本节后面
返回参数	False/true -- 失败/成功

接口说明：本函数的功能是对采集的视频进行字幕，时间戳或图片叠加输出，字幕叠加类为 OsdOverlay，其主要属性和说明如下：

属性名称	类型	说明
font	OsdFont	字体参数类
color	int	字体颜色
flags	int	*设置 ANTI_ALIAS_FLAG 表示框锯齿*
background	int	背景颜色
alpha	int	字体透明度，取值范围 0-255，0 表示完全透明，即不可见
transparent	boolean	背景是否透明

type	int	叠加类型 *TYPE_TEXT* 表示静态 osd 文本 *TYPE_DATETIME* 表示时间戳叠加 *TYPE_IMAGE* 图片叠加
text	String	当为文本时，该属性表示文本内容，当为时间戳时，此属性确定时间戳显示的格式，比如 yy-MM-dd HH:mm:ss
top	int	显示位置起点的 y 值，应该小于屏幕分辨率的高度
left	int	显示位置起点的 x 值，应该小于屏幕分辨率的宽度

关于字体结构 OsdFont 的定位和说明如下：

```
public static class OsdFont
{
    public int  size;          // 字体大小，取值范围：[12, 80]
    public boolean italic;     // > 0 表示 斜体
    public boolean bold;       // >0 表示粗体
    public String  name;       // 字体名称，为空或者 null 表示默认字体

    public OsdFont()
    {
        this.size = 16;
        this.italic = false;
        this.bold = false;
        this.name = "";
    }
}
```

创建文字叠加示例代码：

```
String osdText = "武汉视跃科技-4G 智能联网监控";

OsdOverlay overlay = new OsdOverlay();

overlay.font.size = 36;
overlay.font.italic = true;

overlay.color = Color.rgb(0, 255, 0);
overlay.flags = OsdOverlay.ANTI_ALIAS_FLAG;
overlay.background = Color.rgb(127, 127, 0);
overlay.alpha = 127;
overlay.transparent = false;
overlay.text = osdText;
```

```
overlay.left = 0;
overlay.top = 0;

gbtdevice.setOsd(0, overlay);
```

创建时间戳叠加示例代码:

```
OsdOverlay dateOverlay = new OsdOverlay();
dateOverlay.setDateTime("yyyy-MM-dd HH:mm:ss");
dateOverlay.top = 100;
dateOverlay.left = 20;
dateOverlay.font.size = 36;
dateOverlay.color = Color.rgb(0, 0, 255);
gbtdevice.setOsd(1, dateOverlay);
```

2.3 清除字幕叠加

接口原型：`public void clearOsd()`
接口参数：无
接口说明: 清除对视频中叠加的所有字幕。

2.4 设置相机图像预览的方向

接口原型：`public void setDisplayOrientation(int degree)`
接口参数：

degree（in）	设备的相机采集图像的预览旋转的角度，设置 0，90,180,270 时才有效

接口说明: 该接口用来设置设备端采集的视频本地预览的方向，在 open（）之前调用。

2.5 将设备注册到 gb/t28181 服务器

接口原型：`public int open(int cameraId, SurfaceHolder surfaceHolder,String serverip,String localip,int serverport,String serverid,String userid,String userpwd)`
接口参数：

cameraId（in）	摄像机的 id，0/1 – 后置/前置
surfaceHolder（in）	预览 surface 句柄
Serverip（in）	Gb/t 28181 服务器 IP 地址

Localip（in）	本地 IP 地址（有些设备会存在多网卡现象，故需指定本地 IP）
Serverport（in）	Gb/t28181 服务器端口
Serverid（in）	Gb/t 28181 服务 ID
Userid（in）	设备 ID
Userpwd（in）	设备密码
返回参数	0 表示成功;-1 表示 camera 打开失败;-2 表示注册发送失败(服务器反馈注册失败查看接口 2.6）

接口说明: 该接口用来将设备注册到 Gb/t 28181 服务器。

2.6 注册 gb/t28181 服务器状态回调（该接口新 sdk 已失效，参考第 4 章）

接口原型：public static void setloginhandler(Handler handler)
接口参数：

handler（in）	注册成功与失败信息的事件

接口说明: 该接口引用 com.shiyue.comn.GbtTrack;创建 handle，然后调用：GbtTrack.setloginhandler(handler);设置注册回调事件。实例代码如下：

```
//创建一个 Handler
private Handler handler = new Handler() {
    public void handleMessage(Message msg)
    {
        if(msg.arg1 <= 0)
        {
            //连接失败
            LoginInfo("登录失败!");
        }
        else
        {
            //连接成功
            LoginMainActivety();
        }
    }
};
```

2.7 将设备从 gb/t28181 服务器上下线（反注册）

接口原型：public void close()
接口参数：无
接口说明: 该接口功能是将设备从 gb/t28181 服务器上下线，与 open（）相对。

2.8 设备开启音视频编码与转发

接口原型：public boolean start(int cameraId, SurfaceHolder surfaceHolder
)
接口参数：

cameraId	0/1/2 后置/前置摄像头/otg 外置摄像头
surfaceHolder	SurfaceView 的句柄
返回参数	False/true -- 失败/成功

接口说明: 启动音视频编码与转发，在 open（）成功后调用。该接口针对预览在 SurfaceView 上的场景使用。

2.9 设备开启音视频编码与转发

接口原型：public boolean startcamera(int cameraId, SurfaceTexture st

)
接口参数：

cameraId	0/1/2 后置/前置摄像头/otg 外置摄像头
St	SurfaceTexture 的实例
返回参数	False/true -- 失败/成功

接口说明: 启动音视频编码与转发，在 open（）成功后调用。该接口针对预览在 SurfaceTexture 的场景使用。

2.10 设备停止音视频编码与转发

接口原型：public void stop()
接口参数：无
接口说明: 停止音视频编码与转发，与 start（）相对应。

2.11 发送（更新）设备的位置

接口原型：public void setPosition(double longitude,double latitude)
接口参数：

longitude（in）	设备的精度
latitude（in）	设备的维度

接口说明: 向 gb/t28181 服务器发送自己的位置，注册成功后该接口可调用。

视跃科技 SHIYUE TECHNOLOGY

2.12 上报报警信息

接口原型：public void sendOutAlarm(int alarmlevel ,int alarmMethod, int alarmType, String salarmDesc)

alarmlevel（in）	报警级别，取值范围 1-4，分别表示 1-4 级警情
alarmMethod（in）	报警方式，取值范围 1-7，详见本节后面
alarmType（in）	报警类型，详见本节后面
salarmDesc（in）	报警的文字描述信息

接口说明: 向 gb/t28181 服务器上报报警信息，注册成功后该接口可调用。

报警方式的取值对应的含义如下：

取值	含义
1	电话报警
2	设备报警
3	短信报警
4	Gps 报警
5	视频报警
6	设备故障报警
7	其他报警

当报警方式 alarmMethod 为设备报警（值为 2）时，报警类型的取值对应的含义如下：

取值	含义
1	视频丢失报警
2	设备防拆报警
3	存储设备磁盘满报警
4	设备高温报警
5	设备低温报警

当报警方式 alarmMethod 为视频报警（值为 5）时，报警类型的取值对应的含义如下：

取值	含义
1	人工视频报警
2	运动目标检测报警
3	遗留物检测报警
4	物体移除检测报警
5	拌线检测报警
6	入侵检测报警
7	逆行检测报警
8	徘徊检测报警
9	流量统计报警

10	密度检测报警
11	视频异常检测报警
12	快速移动报警

当报警方式 alarmMethod 为设备故障报警（值为 6）时，报警类型的取值对应的含义如下：

取值	含义
1	存储设备磁盘故障报警
2	存储设备风扇故障报警

3 jni 接口使用说明

JNI 接口提供一些基于 native c++层的接口。如果不需要打开设备的摄像机，直接外部传媒体流，则可直接使用 jni 接口，同时 jni 接口也提供了一下其他设备相关性直接属性的设置。直接调用 jni 接口需要引用 import com.shiyue.gbtdevice.GbtDeviceJni;

3.1 注册 gb/t28181 服务器

接口原型：`public native static long gbtdevice_open(String serverip,String localip, int serverport, int localPort,String serverID,String userID,String userPwd, MFormat fmt);`

接口参数：

Serverip（in）	Gb/t 28181 服务器 IP 地址
Localip（in）	本地 IP 地址（有些设备会存在多网卡现象，故需指定本地 IP）
Serverport（in）	Gb/t28181 服务器端口
localport（in）	设备本地 Gb/t28181 通信端口
Serverid（in）	Gb/t 28181 服务 ID
Userid（in）	设备 ID
Userpwd（in）	设备密码
fmt（in）	媒体格式
返回参数	0 表示失败，非 0 成功，成功返回的为 gb28181 模拟设备端实例

接口说明：该接口用来不通过打开设备的摄像头产生媒体流，而是直接从外部传媒体流。用来当开发者的 app 已经打开了摄像头，无法重复打开摄像头，那么想集成 sdkGB28181 的媒体业务，调用该接口，外部传媒体流。

3.2 提交媒体流

接口原型：`public native static int gbtdevice_write(long handle, MPacket pkt);`

接口参数：

long handle（in）	gb28181 模拟设备端实例

MPacket pkt（in）	媒体包数据结构体
返回参数	0 表示失败，-1 失败

接口说明: 该接口功能为将媒体数据提交给 GB28181 设备端实例，主要用于外部传入媒体流的调用方式，配合 3.1 接口使用，当平台发起预览请求时，媒体流会自动发出，该接口会使用 MPacket 中传入的时间戳。
MPacket 为媒体包结构类型，其原型如下:

```
public class MPacket {

    public int type;        //0/1 - 视频/音频
    public byte[] data;     //媒体数据
    public long pts;        //时间戳
    public int duration;    // 时长
    public int flags;       // 关键帧标记 0/1 - 非关键帧，关键帧
}
```

关于时间戳 pts 的设置，取决与 gbtdevice_open 函数中 MFormat fmt 参数中视频时钟（clockRate）和音频时钟（audioRate）的值。

1. 默认值 clockRate 与 audioRate 为 0 时，MPacket 里的时间戳传 rtp 时间戳。音频时钟 audioRate 的值为其采样率时，音频传 rtp 时间戳，视频时钟 clockRate 值为 90000 时，视频包传 RTP 时间戳。
2. clockRate 与 audioRate 为 1000000 时，传微秒时间戳。

3.3 提交媒体流(重排时间戳)

接口原型：`public native static int gbtdevice_writeex(long handle, MPacket pkt);`
接口说明：同 3.1，不采用 MPacket 中的时间戳，sdk 自动重排时间戳。
该接口为自动计算微秒时间戳。调用该接口用户无需填写 pkt 中的 pts 参数（时间戳），但是注意前面在调用 gbtdevice_open 时，务必要将音视频的采样时钟设置为 1000000（微秒单位），即：

```
fmt.clockRate = 1000000;
fmt.audioRate = 1000000;
```

3.4 对讲模式设置

接口原型： `public native static void gbtdevice_setTalkMode(long handle, int mode);`
接口参数：

long handle（in）	gb28181 模拟设备端实例
mode（in）	对讲媒体流的通信模式，为 1 时对讲采用 udp 通信，为 3 时为被动
返回参数	无

接口说明：语音对讲在外网时使用 udp 必须要穿透，如果采用本公司的平台，那么 udp 外网是做了 NAT 穿透的，调用本 sdk 对接其他平台时，根据其他平台对讲的协议说明，选用不同的对讲通信模式。一般采用被动 tcp 是无需穿透的。

视跃科技 SHIYUE TECHNOLOGY

3.5 设备信息设置

接口原型： `public native static void gbtdevice_setDeviceConfig(String nameprefix, String manufacturer,String address, String model, String owner,String civilCode, String block);`

接口参数：

nameprefix（in）	设备名称前缀，如果设置“test”，GB28181 平台上则为 test1,test2…
manufacturer（in）	公司名称
Address(in)	地址
Model（in）	设备 Model
Owner（in）	设备 Owner
civilCode	设备 civilCode
block	设备 block

接口说明：该接口设置设备的信息。公司名称等。注意若开发者的这些信息传中文信息，如果 android 如果设置的编码方式为 UTF-8，请传入该字符串时将字符转为 GB2312 编码。

3.6 设备通道设置

接口原型：`public native static void gbtdevice_setVideoChannelID(String ChannelId);`

接口参数：

String ChannelId	通道 ID，GB/T28181 规定的 20 位数字编码

该接口不调用，则会默认为设备分配一个 20 位的通道编码，类型为 131 类型。

3.7 打开音频 G711A 编码器

该接口所属类为 JniG711Encoder

```
/**
 * 打开编码器
 * @param codec
 * @param freq    输入频率
 * @param channels    输入通道
 * @return 通道句柄，0 表示失败
 */
public native static int g711_open(int codec, int freq, int channels)
```

3.8 关闭音频 G711A 编码器

该接口所属类为 JniG711Encoder

```
/**
 * 关闭编码器
```

```
 * @param handle
 */
public native static void g711_close(int handle);
```

3.9 编码一帧音频数据为 G711A 格式

该接口所属类为 JniG711Encoder

```
/**
 * 编码一帧音频数据
 * @param handle
 * @param audioData    音频数据
 * @param length       长度
 * @param pts          时间戳
 * @param pkt          编码后的数据包
 * @return
 */
public native static int g711_encode(int handle, byte[] audioData, int
length, long pts, MPacket pkt);
```

4 被动事件的回调与反射

设备注册时平台反馈的状态或者被呼叫时的事件，相对设备属于被动事件，被动事件时通过反射到上层 sdk 接口的。反射的接口原型就是 SYGbtSink。

4.1SYGbtSink 的原型

SYGbtSink 接口实现了两个反射函数，原型如下：

```
    void  ConnectStatus(int status, long statuscode, String info);
    void  callin(int status,int type, String remoteid, String
groupinfo);
}
```

4.1.1Connectstatus 反射函数

该函数为通用状态反射，参数 status 为其类型，statuscode 为子类型标记值，info 为具体的字符信息。比如注册失败的原因，或者呼叫方的信息等等。其具体取值如下，

Status 值	含义	remoteid	groupinfo
201	Type=0 表示挂断呼叫	远端发起 id	群组信息
	Type=1 纯音频呼叫	远端发起 id	群组信息
	Type=2 纯视频呼叫	远端发起 id	群组信息
	Type=3 音视频呼叫	远端发起 id	群组信息

该接口详见《补充接口》文档。

4.1.1callin 反射函数

该反射函数为非国标呼叫事件反射，当被其他设备或者平台以非国标协议进行呼叫，集群对讲，视频会议时，会触发该反射函数，具体取值以及含义如下：

Status 值	含义	备注
0	登录失败	Info 参数为登录失败原因
1	登录成功	
11	收到国标呼叫	Info 参数为呼叫信息
12	挂断国标呼叫	Info 参数为挂断方信息
200	设备授权成功	

4.1.2 云台控制

原型：`void onPtzControl(int id,String deviceid, int directive, int speedOrNum);`

参数：

参数	类型	备注
id	Int	操作 id
deviceid	String	设备 id
directive	Int	云台操作类型，方向
speedOrNum	int	速度或倍率

说明：当设备收到来自国标平台的云台控制消息时，通过反射会回调该函数。

4.1.3 设备重启

原型：`void onTeleBoot(int id,String deviceid, String cmd);`

参数：

参数	类型	备注
id	Int	id
deviceid	String	设备 id
cmd	String	重启命令

说明：当设备收到来自国标平台的设备重启消息时，通过反射会回调该函数。

视跃科技 SHIYUE TECHNOLOGY

4.1.4 强制 I 帧

原型：**void** onForceKeyFrame(**int** id,String deviceid);
参数：

参数	类型	备注
id	Int	id
deviceid	String	设备 id

说明：当设备收到来自国标平台的设备强制 I 帧消息时，通过反射会回调该函数。

4.1.5 手动录像

原型：**void** onRecordCmd(**int** id,String deviceid,**int** cmd);
参数：

参数	类型	备注
id	Int	id
deviceid	String	设备 id
cmd	String	*cmd =0/1 -- 启动录像/停止录像*

说明：当设备收到来自国标平台的设备手动录像消息时，通过反射会回调该函数。

4.1.6 布防撤防

原型：void onGuardCmd(**int** id,String deviceid,**int** cmd);
参数：

参数	类型	备注
id	Int	id
deviceid	String	设备 id
cmd	String	*cmd =0/1 -- 布防/撤防*

说明：当设备收到来自国标平台的布防/撤防消息时，通过反射会回调该函数。

4.1.7 报警开关

原型：**void** onAlarmCmd(**int** id,String deviceid, String cmd);

参数：

参数	类型	备注
id	Int	id
deviceid	String	设备 id
cmd	String	*cmd =0/1 – 打开/关闭*

说明：当设备收到来自国标平台的报警开关消息时，通过反射会回调该函数。

4.1.8 预置位

原型：`void onSetHomePosition(int id,String deviceid, int enabled, int resetTime,int presetIndex);`

参数：

参数	类型	备注
id	Int	id
deviceid	String	设备 id
enabled	Int	0 = 关闭, 1= 开启.
resetTime	**int**	自动归位时间间隔, 单位为秒
presetIndex	**int**	预置位编号, 取值: [0,255].

说明：当设备收到来自国标平台的预置位消息时，通过反射会回调该函数。

4.1.9 设备参数配置

原型：`void onDeviceConfig(int id,GbtBasicParam bp, GbtVideoParamConfig vp);`

参数：

参数	类型	备注
id	Int	id
bp	GbtBasicParam	设备的国标基本参数
vp	GbtVideoParamConfig	设备视频参数

说明：当设备收到来自国标平台的参数配置消息时，通过反射会回调该函数。

类型 GbtBasicParam 的定义如下：

```
public class  GbtBasicParam
{
    public String name;         /// 名称.

    public String deviceID;
    public String serverID;
    public String serverIP;
    public int  serverPort;
    public String domainName;

    public int expiration;   ///
    public int heartBeatInterval;
    public int heartBeatCount;

    public int positionCapability;
    public double longitude;
    public double latitude;
}
```

类型 GbtVideoParamConfig 的参数定义如下：

```
public class  GbtVideoParamConfig
{
    public String name; /// Stream1, Stream2, ...
    public int format;
    public int resolution;
    public int framerate;
    public int bitrate;
    public int bitrateType;
}
```

5 对外接 OTG 摄像头的支持

设备端 sdk 增加了对外接 usb 摄像头即插即用的支持。其使用步骤如下

5.1 加入需要依赖的库文件

将 common-2.12.4.aar 与 libuvccamera.aar 包文件拷贝在 app 下的 libs 文件夹里，与 gbtdevice.aar 处于同一目录。如果是更新 sdk，请拷贝新

的包后，记得 clean 工程，然后重新编译。

在工程的 build.gradle 文件的 dependencies 中添加其依赖，如下图：

```
    implementation(name:'common-2.12.4', ext:'aar')
    implementation(name:'libuvccamera', ext:'aar')
    implementation(name:'gbtdevice', ext:'aar')
```

5.2UVC 摄像头初始化步骤

（1）先导入该类 import com.shiyue.uvccamera.SYUVCCameraAdapter;
（2）创建对象 public SYUVCCameraAdapter syuvc = new SYUVCCameraAdapter();
（3）在 app 的初始化函数（如 onCreate）中打开 usb 摄像头，调用函数为 syuvc.openUSBMonitor(this);this 为环境变量。
（4）在 app 的退出销毁函数（如 onDestroy）中关闭摄像头，调用函数为 syuvc.CloseUSBMonitor();

完成以上步骤的初始化后，即可使用 otg 摄像头，无需额外的编码，直接进行无缝切换。在打开摄像头的接口 startcamera(2,mSurfaceTexture);中，摄像头类型传 0 位后置，1 位前置。现在 2 就为 otg 摄像头。详见接口 2.9。

6 音视频互动/集群对讲的接口支持

本 sdk 新增加了对于音视频互动/集群对讲的支持，集群对讲需要基于视跃公司的 realgbs 平台进行。Android 集群对讲基于 webrtc 协议，所以集成了 webrtc 良好的音频特性，比如回声消除，噪声抑制等。

6.1 使用步骤

（1）首先在工程的 build.gradle 文件的 dependencies 中添加其依赖，如下图：

```
    implementation(name:'gbtdevice', ext:'aar')
    implementation 'org.webrtc:google-webrtc:1.0.30039'
}
```

（2）然后在代码中导入类 SYRTCMediaAdapter。

```
import com.shiyue.syrtc.SYRTCMediaAdapter;
```

（3）创建对象：SYRTCMediaAdapter symediaadapter = new SYRTCMediaAdapter("gbs.realgbs.com", this, null,true);

（4）开始向平台推流（以便平台或其他设备可以取到本设备的流）：

```
symediaadapter.startpush("android");
```

（5）开始取流（拉取平台或其他设备的流）：

```
symediaadapter.startplay("admin",null);
```

6.2 相关类说明

6.2.1 虚基类 SYRTCMediaSink

SYRTCMediaSink 是一个虚拟接口类，它的功能是可以在会话建立后，通过它的接口可以抛出相关媒体的信息，类继承它后，需要实现它的相关函数。

6.2.2.1 接口 void onRemoteVideoTrack(VideoTrack videotrack);

功能:抛出对方的视频信息，方便进行视频预览，该接口（包括属的类）在音视频双向互动中需要，单纯的语音对讲可以不用继承和实现该类与接口。

接口参数：

videotrack（in）	视频媒体信息类，用来表示远端的视频信息，通过 view 进行预览
void（return）	

6.2.2 类 SYRTCMediaAdapter

SYRTCMediaAdapter 是音视频互动与对讲的核心封装类，它将实现整体的交互功能。

6.2.2.1 接口 SYRTCMediaAdapter(String domain,SYRTCMediaSink sink, String ice,boolean isSingleAudio)

功能: SYRTCMediaAdapter 类的初始化创建

参数说明:

domain（in）	本设备端 sdk 对接平台的 ip 或者域名（该功能仅针对我司平台）
sink（in）	设置回调对象，回调会话交互中远端的视频信息，纯对讲可以传 null
Ice（in）	Ice 服务器，可以传 null
IsSingleAudio	表示是否是纯音频交互，true 为纯对讲，false 为音视频互动

视跃科技 SHIYUE TECHNOLOGY

6.2.2.2 接口 void startpush(String id,SYRTCCallback cb)

功能:将本设备的流推到平台供平台或其他设备拉取
参数说明:

id（in）	推送的标记，唯一表示一路设备流的标识符，可以为设备国标 id
cb（in）	表示推送状态的回调，用来调用者查看调试，可以设置为 null

6.2.2.3 接口 void startplay(String id ,SYRTCCallback cb)

功能:将发布到平台上流供备拉取到本机
参数说明:

id（in）	流的标记，如果拉的是平台上的用户或者其他设备推送的流，此处传 4.2.2.2 接口中推流着传的 id，如果是平台上监控的设备流也可以拉，格式是 0/国标 id，比如 0/43000000801320000003
cb（in）	表示推送拉流的回调，用来调用者查看调试，可以设置为 null

6.2.3 虚基类 interface SYRTCCallback

纯虚基类，用来回调在推流和拉流过程中的状态。

6.2.3.1 接口 public void onCreateSesstionSuccess();

功能:用来回调表示拉流或推流的会话创建成功。
参数说明:

6.2.3.2 接口 public void onCreateSesstionFailure(String s);

功能:用来回调表示拉流或推流的会话创建失败。
参数说明:

s（in）	S 字符串表示失败原因

6.2.3.3 接口 void onRequestStream(boolean isSucess);

功能:用来回调表示拉流或推流是否成功
参数说明:

isSucess（in）	状态参数，表示推流或拉流是否成功

7 设备端 2.0 版本

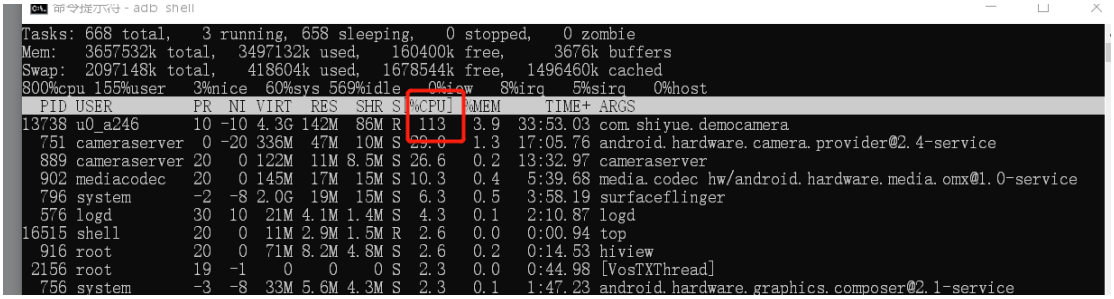
设备端 2.0 版通过充分使用硬件加速功能，提升了设备的使用性能，视频采集也使用 camera2.0 的系统接口。同时通过充分的封装，在接口使用方面相对于之前的版本接口改动比较小。

7.1 使用 2.0 版本的优点

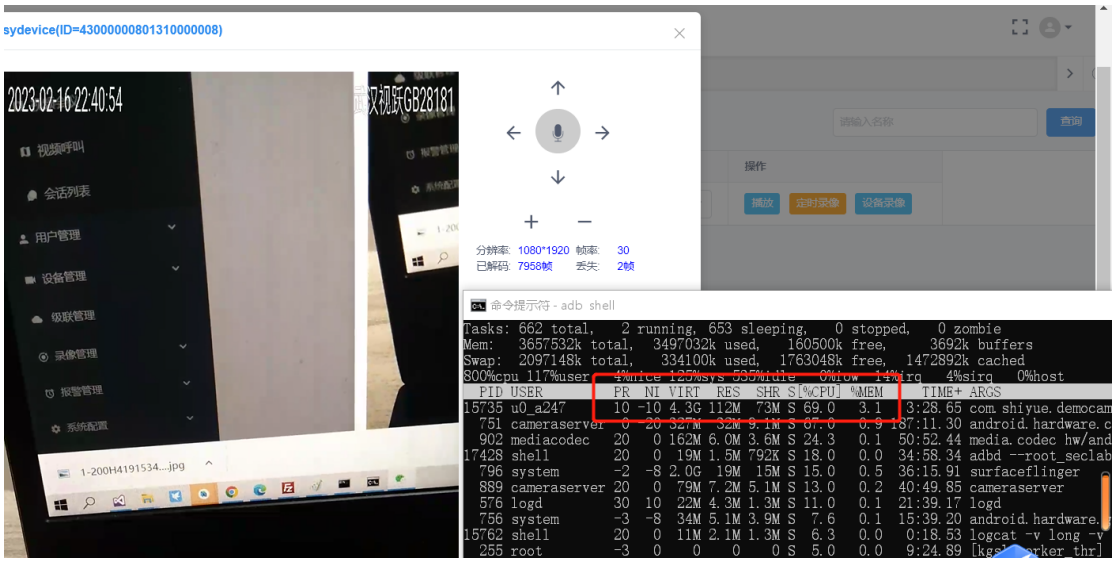
在硬件配置比较低的设备上需要进行图传时，并要对音视频帧率和分辨率有较高要求时，可以使用 2.0 版本。

7.1.1 性能对比

在同一测试设备，荣耀 8C 上，使用 sdk1.0 采集视频，使用 1080P 叠加 4 路动态 osd（叠加位置分别左上，左下，右上，右下）时，视频帧率只能跑到 10 几帧，同时 cpu 持续在 113%，长时间拷机进行这样的功能运行后，一般几个小时后帧率下降更快，具体情况如下图：

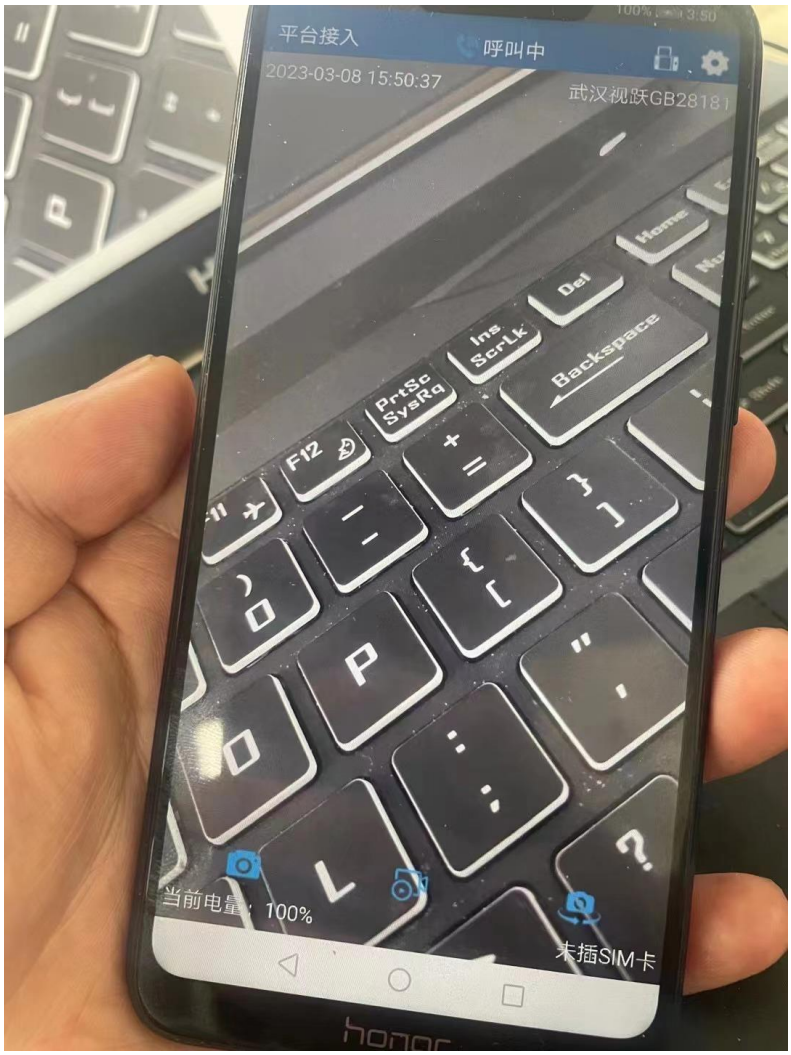


同样的设备，使用 2.0 版本，使用 1080P 叠加 4 路动态 osd（叠加位置分别左上，左下，右上，右下）时，视频帧率跑到 30fps，同时 cpu 在啊 69%，长时间拷机运行均在 30fps，具体如下图：



7.1.2 osd 叠加时设备本地可视

因为充分使用了显存和 gpu 的协作，所以在本地渲染时，也可以预览到 OSD 叠加的效果，即在带有 osd 叠加视频图传时，所见即所传，如下图：



7.3 使用 1.0 和 2.0 版本的接口差异

使用 2.0 版时的的国标类为 CGbtMedia2（之前为 CGbtMedia），类的接口相对于原来仅两处改动，具体改动如下：

1. 创建对象，原 CGbtMedia 为：`CGbtMedia gbtdevice = new CGbtMedia（）。`
现在使用 CGbtMedia2 为 `CGbtMedia2 gbtdevice = new CGbtMedia2（getApplicationContext()）`
2. 打开摄像头：原 CGbtMedia 为

```
gbtdevice.startcamera(1,mSurfaceTexture);
```

现在使用 CGbtMedia2 为

```
boolean ret = gbtdevice.startcamera2(1,mSurfaceTexture, viewWidth, viewHeight);
```

7.2 2.0 版的新增（改变的）接口

2.0 接口会增加一些视频图像处理相关接口。

视跃科技 SHIYUE TECHNOLOGY

7.2.1 打开摄像头

原型：`gbtdevice.startcamera2(int cameraid,int mSurfaceTexture, int viewWidth,int viewHeight);`

参数说明：

cameraid（in）	摄像头的 id
mSurfaceTexture（in）	摄像头预览视图的纹理
viewWidth	摄像头预览的宽度，为 view 的宽度
viewHeight	摄像头预览的高度，为 view 的宽度

说明：打开摄像头，替代 1.0 版本的 `startcamera` 接口。

7.2.2 视频角度旋转

原型：`gbtdevice.setDegrees(int angle)`

参数说明：

angle（in）	采集的视频需要旋转的角度

说明：改接口改变预览角度的同时，也改变图传的角度和分辨率，比如原始分辨率是 1080*1920，旋转 90 度后，图像旋转，图传分辨率也改为 1920*1080

7.2.3 jni 纯视频图像处理—旋转

原型：`SYRender.setRotate(float angle, float x, float y, float z);`

参数说明：

angle（in）	采集的视频需要旋转的角度
x（in）	旋转的轴 0.0/1.0
y（in）	旋转的轴
z（in）	旋转的轴

说明：直接对视频的处理，当极个别硬件摄像头与屏幕物理角度与系统接口信息不符，调用 7.2.2 接口图像达不到要求，或者自己编码时可以采用该接口。

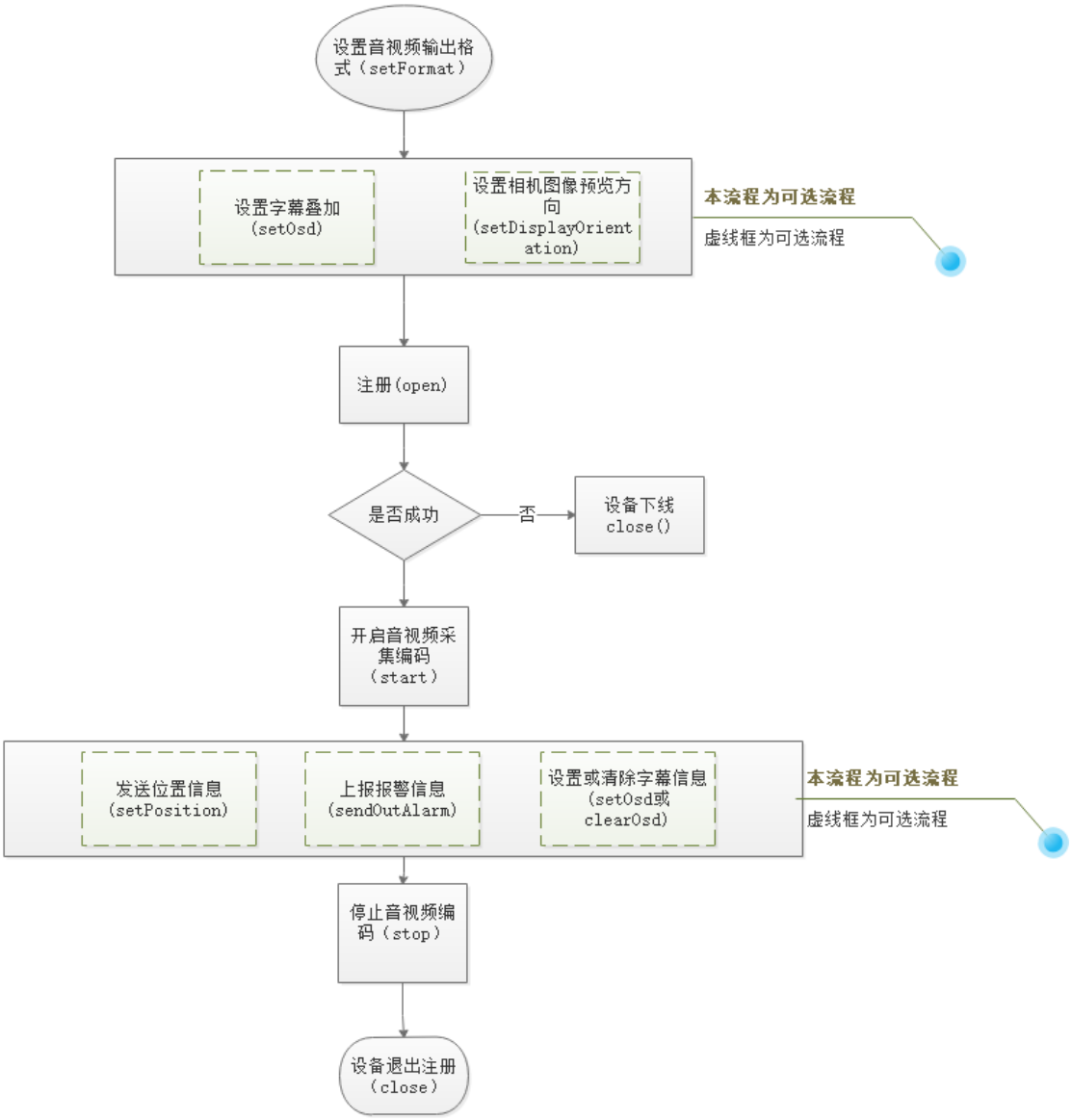
7.2.4 jni 纯视频图像处理—复位

原型：`resetMatrix();`

参数说明：无

说明：调用该接口，图像之前设备的旋等变换将复位。

8 接口调用流程



9 代码示例

详见 demo 代码示例。