Android SDK 使用说明(SIP Version)

SDK 使用说明

1. 导入SDK
   - 将SDK的jar包和 .so库拷贝到工程的libs目录下，右键jar包选择 add as library 选项
   - 在build.gradle中添加以下脚本，指定jni依赖库所在目录

   ```
   sourceSets {
       main {
           jniLibs.srcDirs = ['libs']
       }
   }
   ```

   至此完成SDK的导入
2. 添加权限

   ```
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.READ_PHONE_STATE" />
   <uses-permission android:name="android.permission.RECORD_AUDIO" />
   <uses-permission
   android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
   />
   <uses-permission android:name="android.permission.WAKE_LOCK" />
   <uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
   <uses-permission android:name="android.permission.VIBRATE" />
   <uses-permission
   android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission
   android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission
   android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
   <uses-permission android:name="android.permission.BROADCAST_STICKY" />
   <uses-permission android:name="android.permission.READ_CONTACTS" />
   <uses-permission android:name="android.permission.CALL_PHONE" />
   ```
3. SDK 调用示例说明

```
// 初始化，载入.so 动态库
NativeInterface.init();
// 设置 SDK 回调，必须实现 eventCallBack方法，下面有示例和具体说明
IVoIPNative.setVoIPCallBackParams(this, "eventCallBack",
                "(ILjava/lang/String;[BI)Ljava/lang/Object;");
IVoIPNative.setAudioContext(context);
int ret = NativeInterface.initialize();
ret = NativeInterface.setSipTransportType(2);
ret = NativeInterface.setSrtpEnabled(1, 2);
```

连接到服务器：

```
 String ip_addr = voip_ip_addr.getText().toString(); //ip

   int port =
Integer.valueOf(voip_port.getText().toString()).intValue(); //port

   String acount_ = login_account.getText().toString(); //acount

   String password = login_passwd.getText().toString(); //password

   // 连接到服务器，连接成功或失败通过事件回调通知上层

   NativeInterface.connectToCCPServer(ip_addr, port, acount_,
password);
```

通话相关接口调用实例：

```
// ---------------------------  状态 控件等 定义  ------------------------
---------- //
private static final int CCP_MEETING_TYPE_SINGLE_STEP = 0;      //单步会
议
private static final int CCP_MEETING_TYPE_CONSULT_TRANSFER = 1; //咨询后
转会议

private static final int CALL_STATE_MAKE_CALL_FAILED = 0x01;
private static final int CALL_STATE_CALL_INCOMING = 0x02;
private static final int CALL_STATE_CALL_ALERTING = 0x03;
private static final int CALL_STATE_CALL_ANSWERED = 0x04;
private static final int CALL_STATE_CALL_PAUSED = 0x05;
private static final int CALL_STATE_CALL_REMOTE_PAUSED = 0x06;
private static final int CALL_STATE_CALL_RELEASED = 0x07;
private static final int CALL_STATE_CALL_TRANSFERRED = 0x08;
private static final int CALL_STATE_CALL_VIDEO = 0x09;
private static final int CALL_STATE_CALL_PROCEEDING = 0x0a;
private static final int CALL_STATE_CALL_VIDEO_UPDATE_REQUEST = 0x0b;
private static final int CALL_STATE_CALL_VIDEO_UPDATE_RESPONSE = 0x0c;
private static final int EVENT_SIP_CONNECT = 0x11;
private static final int EVENT_SIP_LOGOUT = 0x12;
private static final int PAUSE_CALL = 90030;
private static final int RESUME_CALL =90031;
private static final int TRANSFER_CALL = 90034;
private static final int TRANSFER_MEETING = 90035;


private static final int USERDATA_FOR_TOKEN =0;
private static final int USERDATA_FOR_USER_AGENT=1;
private static final int USERDATA_FOR_INVITE=2; //发起呼叫
private static final int USERDATA_FOR_200_OK=3; //应答

// button
private Button login_button;                // 登陆
private Button call_button;                 // 呼叫
private Button button_hangup;               // 挂断
private Button button_hold;                 // 保持
private Button button_answer;               // 接听
private Button button_handfree;             // 免提
private Button button_transfer;             // 咨询转移
private Button button_consult_switch;       // 咨询切换
private Button button_consult;              // 咨询
private Button button_consult_meeting;      // 咨询会议
private Button button_consult_hungup;       // 咨询挂断
private Button button_mang_zhuang;          // 盲转
private Button button_one_step_meeting;     // 单步会议


private String callId_single_participant;
private String callid_second_participant;

private TextView state_indicator;   // 状态指示
private EditText call_account;
private EditText login_account;     // 登录账户
private EditText login_passwd;      // 登录密码

private EditText voip_ip_addr;      // 登录 IP
private EditText voip_port;         // 登录端口

// -------------------------  接口调用实例  -----------------------------
--- //
// 登录 登出
  login_button = (Button) findViewById(R.id.button_login);
        login_button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                if(isSipConnected) {
                    isSipConnected = true;
                    Thread thread=new Thread(new Runnable()
                    {
                        @Override
                        public void run()
                        {
                            updateMessage("正在登出...");
                            NativeInterface.disConnectToCCP();
                        }
                    });
                    thread.start();
                } else {
                    isSipConnected = false;
                    Thread thread=new Thread(new Runnable()
                    {
                        @Override
                        public void run()
                        {
                            updateMessage("正在登录...");
                            String ip_addr =
voip_ip_addr.getText().toString();
                            int port =
Integer.valueOf(voip_port.getText().toString()).intValue();
                            String acount_ =
login_account.getText().toString();
                            String password =
login_passwd.getText().toString();
                            // 登录
                            NativeInterface.connectToCCPServer(ip_addr,
port, acount_, password);
                        }
                    });
                    thread.start();
                }


            }
        });

        // 呼叫
        call_button = (Button) findViewById(R.id.button_call);
        call_button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                Thread thread=new Thread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        String account =
call_account.getText().toString();
```

```
                        IVoIPNative.setUserData(USERDATA_FOR_INVITE,
voice_record_input.getText().toString());
                        callId_single_participant =
IVoIPNative.makeCall(0, account);
                        updateMessage("正在呼叫...");

                    }
                });
                thread.start();
            }
        });

        // 挂断
        button_hangup = (Button) findViewById(R.id.button_hangup);
        button_hangup.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        // 挂断电话
                        updateMessage("正在挂机...");

 IVoIPNative.releaseCall(callId_single_participant, 0);

                    }
                });
                work.start();
            }
        });

        // 接听
        button_answer = (Button) findViewById(R.id.button_answer);
        button_answer.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        IVoIPNative.setUserData(USERDATA_FOR_200_OK,
voice_record_input.getText().toString());

 IVoIPNative.acceptCall(callId_single_participant);
                        updateMessage("电话接听中...");
                    }
                });
                work.start();
            }
        });

        // 免提
        button_handfree = (Button) findViewById(R.id.button_handfree);
        button_handfree.setOnClickListener(new Button.OnClickListener()
{
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        // 免提接口
                    }
                });
                work.start();
            }
        });

        // 保持
        button_hold = (Button) findViewById(R.id.button_hold);
        button_hold.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        if(isCallPause) {
                            // 恢复
                            isCallPause = false;
                            updateButton(button_hold, "保持");

 IVoIPNative.resumeCall(callId_single_participant);
                            updateMessage("呼叫恢复.");
                        } else {
                            // 保持
                            isCallPause = true;
                            updateButton(button_hold, "恢复");

 IVoIPNative.pauseCall(callId_single_participant);
                            updateMessage("呼叫保持中...");
                        }
                    }
                });
                work.start();
            }
        });

        // 盲转
        button_mang_zhuang = (Button)
findViewById(R.id.session_turn_btn);
        button_mang_zhuang.setOnClickListener(new
Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        updateMessage("盲转中...");
                        //
IVoIPNative.pauseCall(callId_single_participant);
                        String account =
transfer_number.getText().toString();

 IVoIPNative.transferCall(callId_single_participant, account, 0);
                        istransfer = true;
                    }
                });
                work.start();
            }
        });

        // 咨询
        button_consult = (Button) findViewById(R.id.consult_btn);
        button_consult.setOnClickListener(new Button.OnClickListener()
{
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {

 IVoIPNative.pauseCall(callId_single_participant);
                        isConsulting = true;
                        updateMessage("请等待，正在咨询...");
                    }
                });
                work.start();
            }
        });


        // 咨询挂断
        button_consult_hungup = (Button)
findViewById(R.id.consult_hungup_btn);
        button_consult_hungup.setOnClickListener(new
Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        updateMessage("咨询挂断中...");

 IVoIPNative.releaseCall(callid_second_participant, 0);
                    }
                });
                work.start();
            }
        });

        // 咨询切换
        button_consult_switch = (Button)
findViewById(R.id.consult_switch_btn);
        button_consult_switch.setOnClickListener(new
Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        if(isConsulting) {
                            updateMessage("与客户通话切换中...");

 IVoIPNative.pauseCall(callid_second_participant);

 IVoIPNative.resumeCall(callId_single_participant);
                            isConsulting = false;
                        } else {
                            updateMessage("与专家通话切换中...");

 IVoIPNative.resumeCall(callid_second_participant);

 IVoIPNative.pauseCall(callId_single_participant);
                            isConsulting = true;
                        }
                    }
                });
                work.start();
            }
        });



        // 咨询转移
        button_transfer = (Button) findViewById(R.id.button_transfer);
        button_transfer.setOnClickListener(new Button.OnClickListener()
{
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                           updateMessage("咨询转中...");
                        isConsultingTransfer = true;
                         String account =
transfer_number.getText().toString();
//IVoIPNative.releaseCall(callid_second_participant, 0);

 IVoIPNative.consultTransferCall(callId_single_participant,
callid_second_participant, account);

                    }
                });
                work.start();
            }
        });

        // 单步会议
        button_one_step_meeting = (Button)
findViewById(R.id.button_one_step_metting);
        button_one_step_meeting.setOnClickListener(new
Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        updateMessage("开始单步会议...");

 IVoIPNative.transferMeeting(CCP_MEETING_TYPE_SINGLE_STEP,
callId_single_participant, null,
three_person_metting_text.getText().toString());
                    }
                });
                work.start();
            }
        });

        // 咨询会议
        button_consult_meeting = (Button)
findViewById(R.id.consult_meeting_btn);
        button_consult_meeting.setOnClickListener(new
Button.OnClickListener() {
            public void onClick(View v) {
                Thread work = new Thread(new Runnable() {
                    @Override
                    public void run() {
                        updateMessage("开始咨询会议...");
                        isTransferMeeting = true;

 IVoIPNative.transferMeeting(CCP_MEETING_TYPE_CONSULT_TRANSFER,
callId_single_participant, callid_second_participant, null);
                    }
                });
                work.start();
        }
    });
```

SDK 回调Callback实现：

```
// 电话状态回调
   public Object eventCallBack(int event, String id, byte[] message,
int state) {
        switch (event) {
            case EVENT_SIP_CONNECT:
                if(state == 200) {
                    isSipConnected = true;
                    updateMessage("登陆成功!");
                    updateButton(login_button, "登出");

                } else {
                    isSipConnected = false;
                    updateMessage("登陆失败!");
                }
                break;
            case EVENT_SIP_LOGOUT:
                if(state == 200) {
                    isSipConnected = false;
                    updateButton(login_button, "登录");
                    updateMessage("退出成功! ");
                } else {
                    updateMessage("退出失败! ");
                }
                break;
            case CALL_STATE_CALL_INCOMING:
                callId_single_participant = id;
                updateMessage("有电话呼入! ");

 IVoIPNative.enableRingBackTone(callId_single_participant);
                break;
            case CALL_STATE_MAKE_CALL_FAILED:
                updateMessage("呼叫失败! ");
                break;
            case CALL_STATE_CALL_ALERTING:
                updateMessage("正在振铃...");
                break;
            case  CALL_STATE_CALL_ANSWERED:
                updateMessage("呼叫接听，通话中...");
                break;
            case CALL_STATE_CALL_PAUSED:
                break;
            case  CALL_STATE_CALL_REMOTE_PAUSED:
                break;
            case CALL_STATE_CALL_RELEASED:
            {
                if(callid_second_participant != null &&
callid_second_participant.equals(id)) {
                    callid_second_participant = null;
                    if(isTransferMeeting) {
                        isTransferMeeting = false;
                        updateMessage("已退出咨询会议！");
                        return null;
                    } else if(isConsultingTransfer){
                        isConsultingTransfer = false;
                        return null;
                    }
                    updateMessage("咨询已挂断，取回主叫通话");
                    IVoIPNative.resumeCall(callId_single_participant);

                } else if(callId_single_participant != null &&
callId_single_participant.equals(id)) {
                    callId_single_participant = null;
                    if(isTransferMeeting) {
                        return null;
                    }
                    updateMessage("电话已挂断！");
                    callid_second_participant = null;
                }

            }
                break;
            case  CALL_STATE_CALL_TRANSFERRED:
                break;
            case CALL_STATE_CALL_VIDEO:
                break;
            case CALL_STATE_CALL_PROCEEDING:
                break;
            case CALL_STATE_CALL_VIDEO_UPDATE_REQUEST:
                break;
            case CALL_STATE_CALL_VIDEO_UPDATE_RESPONSE:
                break;
            case PAUSE_CALL:
            {
                // step 1
                if(state == 200) {
                    updateMessage("通话保持成功.");
                } else {
                    updateMessage("通话保持失败!");
                }
                // step 2
                if(isConsulting && state == 200) { // 咨询
```

```
                    callid_second_participant = IVoIPNative.makeCall(0,
transfer_number.getText().toString());
                    isConsulting = false;
                } else if(istransfer && state == 200){ // 盲转
                    //String account =
transfer_number.getText().toString();

 //IVoIPNative.transferCall(callId_single_participant, account, 0);
                    istransfer = false;
                }
            }
                break;
            case RESUME_CALL:
            {
                if(state == 200) {
                    updateMessage("恢复通话成功！");
//                    if(isConsultingTransfer) {
//                        updateMessage("正在转接...");
//                        isConsultingTransfer = false;
//                        String acount =
transfer_number.getText().toString();
//
IVoIPNative.transferCall(callId_single_participant, acount, 0);
//                    }
                } else {
                    updateMessage("恢复通话失败！");
                }
            }
                break;
            case TRANSFER_CALL: {
                if (state == 200) {
                    updateMessage("转接成功！");
                } else {
                    updateMessage("转接失败！");
                }
            }
                break;
            case TRANSFER_MEETING:
            {
                if(state == 200) {
                    updateMessage("会议成功！");
                } else {
                    updateMessage("会议失败！");
                }
            }
                break;
            default:
                break;
        }
```

```
            return null;
        }
```

SDK 回调状态列表：

```
private static final int CCP_MEETING_TYPE_SINGLE_STEP = 0;       //单步会议
private static final int CCP_MEETING_TYPE_CONSULT_TRANSFER = 1; //咨询后转会议

private static final int CALL_STATE_MAKE_CALL_FAILED = 0x01;
private static final int CALL_STATE_CALL_INCOMING = 0x02;
private static final int CALL_STATE_CALL_ALERTING = 0x03;
private static final int CALL_STATE_CALL_ANSWERED = 0x04;
private static final int CALL_STATE_CALL_PAUSED = 0x05;
private static final int CALL_STATE_CALL_REMOTE_PAUSED = 0x06;
private static final int CALL_STATE_CALL_RELEASED = 0x07;
private static final int CALL_STATE_CALL_TRANSFERRED = 0x08;
private static final int CALL_STATE_CALL_VIDEO = 0x09;
private static final int CALL_STATE_CALL_PROCEEDING = 0x0a;
private static final int CALL_STATE_CALL_VIDEO_UPDATE_REQUEST = 0x0b;
private static final int CALL_STATE_CALL_VIDEO_UPDATE_RESPONSE = 0x0c;
private static final int EVENT_SIP_CONNECT = 0x11;
private static final int EVENT_SIP_LOGOUT = 0x12;
private static final int PAUSE_CALL = 90030;
private static final int RESUME_CALL =90031;
private static final int TRANSFER_CALL = 90034;
private static final int TRANSFER_MEETING = 90035;
```

SDK 接口列表及说明

```
// ---------------------------------- NativeInterface ---------------------------------- //
/*! @function
 ***************************************************************************
 函数名    : getVersion
 功能      : 获取SDK版本信息
 返回值    : 版本信息字符串。
 说明      : 版本信息格式为："版本号#平台#ARM版本#音频开关#视频开关#编译日期 编译时间"
            版本号：格式为X.X.X 如1.1.18
            平台： Android、 Windows、 iOS
            ARM版本： arm、 armv7、 armv5
            音频开关： voice=false、 voice=true
            编译日期： "MM DD YYYY"  如"Jan 19 2013"
            编译时间： "hh:mm:ss"    如"08:30:23")
 **************************************************************************/
```

```
public static native String getVersion();

/**
* 加载.so动态库
*/
public static void init() {
   System.loadLibrary("ECMedia");
   System.loadLibrary("serphone");
}

/**
* 客户端初始化
* 返回值   : 是否初始化成功 0: 成功;  非0失败
*/
public static native int initialize();

/*! @function
 ***************************************************************************
******
 函数名   : connect
 功能     : 连接呼叫控制服务器
 参数     : [IN]  proxy_addr   : 呼叫控制服务器地址
 [IN]  proxy_port     : 呼叫控制服务器端口
 [IN]  account     : 用户帐号
 [IN]  password    : 用户密码
 返回值   : 是否连接成功, 0: 成功;  非0失败
 ***************************************************************************
******/
public static native int connectToCCP(String proxy_addr, int proxy_port,
String account, String password);

/**
* 功能: 与服务器断开连接
* 返回值   : 是否成功, 0: 成功;  非0失败
*/
public static native int disConnectToCCP();

/**
 * 功能     : 设置显示用户名
 * 参数     : [IN]  username   : 用户名
 * 返回值   : 是否设置成功 0: 成功;  非0失败
*/
public static native int setUserName(String name);


// --------------------------------- IVoIPNative --------------------------
----- //
 /**
     * 功能: 设置音频设备上下文
     * 参数: [IN] obj : 上下文句柄
     * 返回值    : 是否设置成功, 0: 成功;  非0失败
     */
    public static native int setAudioContext(Object obj);

    /**
     * 功能      : 发起呼叫
     * 参数      : [IN]  called       : 被叫方号码
     *            [IN]  type : 0: 音频 1: 视频
     * 返回值    : 返回值为callid,本次呼叫的唯一标识; NULL表示失败.
     */
    public static native String makeCall(int type, String called);

    /**
     * 设置底层库回调接口
     * @param obj 回调接口所在类实例
     * @param method 回调方法
     * @param methodSig 回调方法参数
     */
    public static native void setVoIPCallBackParams(Object obj, String
method, String methodSig);



    /**
     * 功能      : 挂机
     * 参数      : [IN]  callid       : 当前呼叫的唯一标识,  如果callid 为NULL,这代
表所有呼叫.
     [IN]    reason: 预留参数,挂断原因值, 可以传入大于1000的值, 通话对方会在
onMakeCallFailed收到该值
     * 返回值    : 是否成功 0: 成功;  非0失败
     */
    public static native int releaseCall(String callid, int reason);
    /**
     * 功能      : 应答呼入
     * 参数      : [IN]  callid       : 当前呼叫的唯一标识
     * 返回值    : 是否成功 0: 成功;  非0失败
     */
    public static native int acceptCall(String callid);

    /**
     * 转接接口
     * @param callid  与用户通话callid
     * @param dest  转接目标号码
     * @param type 统一 为0
     * @return 是否成功 0: 成功;  非0失败
     */
    public static native int transferCall(String callid, String dest, int
type);
```

```
    /**
     *  会议
     * @param type  CCP_MEETING_TYPE_SINGLE_STEP = 0;      //单步会议,
     *              CCP_MEETING_TYPE_CONSULT_TRANSFER = 1; //咨询后转会议
     * @param callid  用户通话callid
     * @param consultedCallid  咨询callid
     * @param consultedUser  咨询用户号码
     * @return 是否成功 0: 成功;  非0失败
     */
    public static native int transferMeeting(int type, String callid,
String consultedCallid, String consultedUser);
    /**
     * 被叫设置是否给主叫方回复呼叫回铃音
     * @param callId 当前呼叫的唯一标识
     * @return 是否成功 0: 成功;  非0失败
     */
    public static native int enableRingBackTone(String callId);


    /**
     * 功能     : 拒绝呼叫
     * 参数     : [IN]  callid      : 当前呼叫的唯一标识
     *          [IN]  reason    : 拒绝呼叫的原因
     * 返回值   : 是否成功 0: 成功;  非0失败
     */
    public static native int rejectCall(String callid, int reason);

    /**
     * 功能     : 暂停呼叫，呼叫暂停以后，本地的语音数据将不再传递到对方.
     * 参数     : [IN]  callid      : 当前呼叫的唯一标识
     *           [IN]     reason    : 拒绝呼叫的原因
     * 返回值   : 是否成功 0: 成功;  非0失败
     */
    public static native int pauseCall(String callid);

    /**
     * 功能     : 恢复暂停的呼叫
     * 参数     : [IN]  callid      : 当前呼叫的唯一标识
     * 返回值   : 是否成功 0: 成功;  非0失败
     */
    public static native int resumeCall(String callid);

    /**
     * 功能     : 取消呼叫(尚未实现，不能使用)
     * 参数     :  [IN]  callid           : 当前呼叫的唯一标识
     * 返回值   : 是否成功 0: 成功;  非0失败
     */
    public static native int cancelCall(String callid);
    /**
     * 功能      : 发送按键信息
     * 参数      : [IN]  callid      : 当前呼叫的唯一标识.
     [IN]    dtmf      : 一个按键值，'0'-'9' '*' ,'#'
     * 返回值    : 是否成功 0: 成功;  非0失败
     */
    public static native int sendDTMF(String callid, char dtmf);

    /**
     * 功能      : 获取通话状态
     * 参数      : [IN]  callid      : 当前呼叫的唯一标识.
     * 返回值    : 是否成功 0: 成功;  非0失败
     */
    public static native int getCallState(String callid);

    /**
     *  设置录音文件名接口
     * @param type     主叫时 : USERDATA_FOR_INVITE=2; //发起呼叫,  被叫时:
SERDATA_FOR_200_OK=3; //应答
     * @param userData  录音文件名
     * @return 是否成功 0: 成功;  非0失败
     */
    public static native int setUserData(int type, String userData);

    /**
     * 咨询转移
     * @param callid  用户callid
     * @param consultCallid  咨询callid
     * @param destination 被转接号码
     * @return 是否成功 0: 成功;  非0失败
     */
    public static native int consultTransferCall(String callid, String
consultCallid, String destination);
```