ZjRTC使用说明v1.9.0

安装环境

1. 复制 `zjrtc.aar` 包到 `app的libs` 目录下；
2. 复制 `node_modules` 到工程根目录下；
3. 修改 `工程` 下的 `build.gradle` 文件：

   代码如下：

```
allprojects {
    repositories {
        jcenter()
        maven {
            url "$rootDir/node_modules/react-native/android"
        }
    }
}
```

4. 修改app下的build.gradle文件：

   代码如下：

```
android {
    defaultConfig {
        minSdkVersion 16    // <---确保最低版本在16或16以上
        ndk {
            abiFilters "armeabi-v7a", "x86"
        }
    }
    configurations.all{
         resolutionStrategy.force'com.google.code.findbugs:jsr305:3.0.0'
    }
    repositories {
        //配置aar
        flatDir {
            dirs 'libs'
        }
    }
```

```
}
dependencies {
    compile 'com.facebook.react:react-native:+'
    compile(name:'zjrtc', ext:'aar') //添加aar包
}
```

确保`External Library`中包含的`react-native`版本为`0.44.3`。

偏好设置

在进行通话前，需要先进行偏好设置。

偏好设置包括：服务器地址设置、视频参数设置、功能设置。其中服务器地址是必需设置项，没有设置将导致通话失败。视频参数和功能设置作为可选项，不设置将使用默认值。

偏好设置设置一次即可一直生效，直到重新设置将使用新参数值。

ZjVideoPreferences(Context context)

进行偏好设置需构造偏好设置对象。
如在Activity中可按以下代码进行构造：

```
ZjVideoPreferences prefs = new ZjVideoPreferences(this);
```

setDomain(String domain)(必需)

设置服务器地址，必需设置项，没有设置将导致呼叫失败。

setBandwidth(int bandwidth)

设置呼叫速率，上行/下行一致。默认为576kbps。

setBandwidth(int upBw, int downBw)

分别设置上行/下行呼叫速率。

setVideoSize(int width, int height)

设置视频分辨率，发送/接收的视频分辨率一致。默认视频分辨率为640x360。

setVideoSize(int upWidth, int upHeight, int downWidth, int downHeight)

分别设置手机发送/接收的视频分辨率。

setVideoFps(int fps)

设置视频帧率，发送/接收的视频帧率一致。默认帧率20。

setVideoFps(int upFps, int downFps)

分别设置发送/接收视频帧率。

视频属性参考表

视频属性	分辨率(宽x高)	帧率(fps)	速率(kbps)
720p	1280x720	25	800
720p	1280x720	20	640
720p	1280x720	15	512
576p	1024x576	25	600
576p	1024x576	20	512
576p	1024x576	15	400
448p	768x448	25	400
448p	768x448	20	360
448p	768x448	15	320
360p	640x360	25	360
360p	640x360	20	300
360p	640x360	15	240
WCIF	512x288	25	260
WCIF	512x288	20	192
WCIF	512x288	15	160

setPrintLogs(boolean printLogs)

打印日志信息，日志以文件形式存储在本地。
参数为true 打印日志，false不打印日志；默认值为false。
当设备有SD卡时，文件存储在SD卡中；无SD卡时，保存在应用的目录下。存储日志的文件夹名称为`应用包名+Logs`，文件名称为`日期_时间.log`。
以华为EC6108V9盒子为例，其中一条日志：
`/mnt/sdcard/com.testrnsdkLogs/'2017-09-20_15-31-52.log'`
手机可通过`文件管理`功能，查看SD卡的`com.testrnsdkLogs`文件夹。

setSoftCode(boolean softCode)

设置使用视频编码方式为软编解。
参数为true 使用软编解，false硬编解；默认值为false。
如果设备不支持硬编解码，入会后可能出现手机端接收的视频不正常，或其他端收到手机端发送的视频有问题的情况，此时可使用此方法关闭硬编解码，使用软编解方式。

setHideRNUI(boolean hideRNUI)

隐藏通话界面的默认按钮。当需要自定义通话界面时，需调用此方法隐藏默认按钮。
参数为true 隐藏按钮，false显示按钮；默认值为false。
注意：调用setHideRNUI(true)后，SDK中将不再出现任何弹出框提示信息，开发者需调用ZjVideoManager的addZjCallListener()方法，在callState()回调方法中根据state和info信息自行处理。

setTvSupport(boolean tvSupport)

当设备为TV/盒子时，需调用此方法。
参数为true 为TV/盒子，false为手机；默认值为false。

通话界面

自定义通话界面

如果需要自定义通话界面，可按以下方法添加子View至ZjVideoActivity中，你的界面将会覆盖在通话界面上。

1. 进行偏好设置，隐藏SDK自带布局：

```
ZjVideoPreferences prefs = new ZjVideoPreferences(this);
prefs.setHideRNUI(true);
```

2. 新建 `ZjVideoActivity` 的子类，作为你的通话界面。
3. 重写其中的 `onCreate()` 方法，在onCreate()中调用 `addContentView()` 方法添加子界面。

   代码示例：

```
public class MyVideoActivity extends ZjVideoActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //R.layout.video_layout你的布局文件
        View view = LayoutInflater.from(this).inflate(R.layout.video_lay
out, null);
        ViewGroup.LayoutParams params = new ViewGroup.LayoutParams (View
Group.LayoutParams.MATCH_PARENT,ViewGroup.LayoutParams.MATCH_PARENT);
        addContentView(view, params);
    }
}
```

4. 最后跳转至你的通话界面，完成自定义通话界面。

```
startActivity(new Intent(this, MyVideoActivity.class));
```

横屏显示

如果需要横屏显示通话界面，可在 `AndroidManifest.xml` 中，对通话界面ZjVideoActivity或其子类的 `screenOrientation` 属性进行设置：

```
<!--使用ZjVideoActivity-->
<activity android:name="com.zjrtc.ZjVideoActivity"
    android:screenOrientation="landscape" >
</activity>
```

建立通话

呼叫说明

每次呼叫，需要调用ZjVideoManager设置显示名、呼叫地址、密码(有密码则需要设置)，然后跳转至通话界面ZjVideoActivity或其子类。

注意：

- 每次呼叫，都需要使用ZjVideoManager.getInstance()获取ZjVideoManager对象实例，在对话结束后此对象会被销毁。

示例：

```
//设置服务器地址、显示名称、呼叫地址、呼叫密码;
ZjVideoManager.getInstance().setDisplayName("");
ZjVideoManager.getInstance().setAddress("");
ZjVideoManager.getInstance().setPwd("");//没有密码不用设置
startActivity(new Intent(this,ZjVideoActivity.class));
//跳转至自定义通话界面 MyVideoActivity
//startActivity(new Intent(this,MyVideoActivity.class));
```

ZjVideoManager

在通话建立前，可调用ZjVideoManager进行呼叫参数设置。

方法如下：

setDisplayName(String displayName)(必需)

设置显示名称。

setAddress(String address)(必需)

设置呼叫地址。

setPwd(String pwd)

设置呼叫地址所需要的密码。

通话管理

ZjVideoManager

在通话过程中，可调用ZjVideoManager进行结束通话、关闭麦克风等其他操作。

disconnect()

断开通话连接，即挂断。

disconnectAll()

断开所有通话连接，即结束会议。

toggleMicrophone()

打开/关闭麦克风。

toggleCamera()

打开/关闭摄像头。

switchCamera()

切换摄像头。

openSpeaker(Context context,boolean on)

打开/关闭系统扬声器。(建立通话前调用)

参数说明：

- `context` - `Context`： Android上下文；
- `on` - `boolean`： true打开;false关闭；

addZjCallListener(ZjCallListener listener)

添加呼叫相关回调方法，当呼叫状态改变、静音状态改变时，其中回调方法会被调用。(建立通话前调用)
每次呼叫都需要创建新的ZjCallListenerBase对象，在通话结束后对象会被销毁。

参数说明：

- ZjCallListener 回调接口，可创建接口的实现类ZjCallListenerBase，按需求重写其中方法。

ZjCallListener 接口方法说明：

```
/**
 * 呼叫状态回调方法，当呼叫状态改变时此方法会被调用
 * @param state 呼叫状态
 *              CALL_INIT:       呼叫初始化
 *              CALL_CONNECTING: 呼叫连接中
```

```
     *               CALL_CONNECTED:  呼叫成功
     *               CALL_END:        呼叫断开
     *               CALL_ERROR:      呼叫错误
     * @param info 额外信息
     *               CALL_CONNECTED:  uuid
     *               CALL_END:        断开原因
     *               CALL_ERROR:      错误信息
     */
    void callState(String state, String info);

    /**
     * 音视频状态信息
     * @param state
     *               outVideoLost: 发送视频丢包率
     *               outAudioLost: 发送音频丢包率
     *               inVideoLost: 接收视频丢包率
     *               inAudioLost: 接收音频丢包率
     *               额外说明: 初次状态值为0; "NaN%"表示正在收集状态值。
     */
    void videoState(String state);


    /**
     * 静音状态回调方法，当静音状态改变时此方法会被调用
     * @param muted 是否被静音
     */
    void onMuteChanged(boolean muted);
```

使用示例:

```
ZjVideoManager.getInstance().addZjCallListener(new ZjCallListenerBase(){

    @Override
    public void callState(String state, String info) {
        Log.i(TAG, "callState: "+state+" "+info);
    }
    @Override
    public void videoState(String state) {
        Log.i(TAG, "videoState: "+state);
    }
    @Override
    public void onMuteChanged(boolean muted) {
        Log.i(TAG, "onMuteChanged: "+muted );
    }
});
```

被呼功能

移动端被呼功能包括`点对点被呼`，和`会议室邀请被呼`两种场景。
被呼功能的实现，首先需要在APP上通过接口进行账号登录，登录成功后该账号被呼叫时，服务器端会通过极光推送发送消息给APP，APP可根据收到的消息来建立通话，或调用接口进行拒接。

极光推送

被呼功能需要使用极光推送，来接收服务器发送的消息。

关于极光推送，你需要做：

1. 在极光推送官网中创建应用；
2. 创建应用成功后，请将AppKey和Master Secret发送给紫荆云视服务保障人员，他将为你们的APP在平台上进行相关配置；
3. 创建应用成功后，完成推送设置，在Android下填写你们的应用包名；
4. 集成极光推送服务至Android APP中；

极光推送官网：https://www.jiguang.cn/
极光集成文档：https://docs.jiguang.cn/jpush/client/Android/android_guide/
紫荆云视服务保障邮箱：xiaoqiang.yue@zijingcloud.com

以下是被呼功能的具体实现过程：

登录

接口描述

账号登录功能，将账号、设备信息提交至服务器。

数据定义

请求地址： `https://domain/api/registrations/<:account>/new_session`

account为登录账号，需要对账号进行URL编码。
URL编码可参考以下代码：

```
String accountEncoded = URLEncoder.encode(account, “UTF-8”);
```

请求方式： POST
请求参数：

参数类别	参数名称	类型	注释	说明
请求头部(Header)	X-Cloud-Authorization	String	认证信息	用户名和密码的Base64加密字符串；可参考举例说明。
请求头部(Header)	Authorization	String	认证信息	用户名和密码的Base64加密字符串；
请求参数(Body)	device_id	String	设备id	格式：极光推送RegistrationID__极光推送APPKEY
请求参数(Body)	device_type	String	设备类型	必须填写android

Header举例说明：

账号`test@zijingcloud.com`，账号密码`123456`。
用户名为test，密码为123456，编码格式：`用户名:密码`
将`test:123456`进行Base64编码为dGVzdDoxMjM0NTY=
然后在已编码的字符串前添加`“x-cloud-basic ”`

最终认证信息为：

Header	消息体
X-Cloud-Authorization	x-cloud-basic dGVzdDoxMjM0NTY=
Authorization	x-cloud-basic dGVzdDoxMjM0NTY=

接收消息

获取消息

接收消息需使用自定义`广播接收器`，创建过程如下：

1. 创建自定义BroadcastReceiver，重写onReceive()方法；

```
public class ZjJPushReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
```

```
        if (JPushInterface.ACTION_MESSAGE_RECEIVED.equals(intent.getActi
on())) {
            String msg = bundle.getString(JPushInterface.EXTRA_MESSAGE);
            String json = new String(Base64.decode(msg, Base64.CRLF));
        }
    }
}
```

2. 在AndroidManifext.xml中注册receiver。

```
<receiver
    android:name=".ZjJPushReceiver"
    android:enabled="true">
    <intent-filter>
        <action android:name="cn.jpush.android.intent.MESSAGE_RECEIVED"
/>

        <category android:name="com.zijingdemo" />
    </intent-filter>
</receiver>
```

详情见Demo的ZjJPushReceiver类。

消息说明

App收到消息后，可根据消息内容来实现接听和拒接功能。
消息字段说明：

参数名称	类型	注释
remote_alias	String	呼叫发起方的账号地址
remote_display_name	String	呼叫发起方的显示名称
conference_alias	String	通话地址
token	String	通话鉴权token
time	String	呼叫发起的时间
service_type	String	呼叫类型，点对点或会议室

接听

被呼后接听，只需要调用`ZjVideoManager`的`setMsgJson()`方法，设置接收到的消息即可，方法如下：

setMsgJson(String msg)

设置接听所需的消息内容，为广播接收器中接收的json字符串;

设置完成后跳转至通话界面，即可建立通话，接听功能完成。

```
ZjVideoManager manager = ZjVideoManager.getInstance();
manager.setMsgJson(msgJson);//msgJson为广播接收器中收到的消息
startActivity(new Intent(this, ZjVideoActivity.class));
```

拒接

接口描述

账号被呼叫后，调用接口拒接此次通话。

数据定义

请求地址： `https://domain/api/services/<:address>/end_session?token=<:token>`

address即通话地址，为接收消息的`conference_alias`字段，需要对通话地址进行URL编码；token为接收消息的token字段。

请求方式： POST
请求参数： 无请求参数