ZjRTC 使用说明-v1.8.10

1. 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**。

2. 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。

3. 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. 新建 ZjVideoActivity 的子类。
2. 重写 onCreate()方法。
3. 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_layout, null);
        ViewGroup.LayoutParams params = new ViewGroup.LayoutParams
(ViewGroup.LayoutParams.MATCH_PARENT,ViewGroup.LayoutParams.MATCH_P
ARENT);
        addContentView(view, params);
```

```
    }
}
```

在进行呼叫前，你还需要进行偏好设置，隐藏自带按钮：

调用 ZjVideoPreferences 的 setHideRNUI(true)方法。

最后跳转至你的自定义通话界面：

```
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>
```

4. 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

在通话过程中，可调用 ZjVideoManager 进行呼叫参数设置、挂断关闭麦克风或其他功能操作。

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。

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