一、权限设置

在 APP 的 AndroidManifest.xml 文件中添加如下权限:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
 />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAG
E" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE
" />
```

二、添加 view 设置

```
<com.wangsu.libwswebrtc.WsWebRTCSurfaceView
        android:id="@+id/id_surface_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_centerHorizontal="true"
        android:textSize="24sp"
        />
```

三、创建 sdk

1. `WsWebRTCView webrtcView = findViewById(R.id.id_surface_view);`

2. WsWebRTCView 类定义如下:

```
public interface WsWebRTCView extends VideoSink {
      //缩放模式参数，保留视频宽高比
    int SCALE_FIT = 0;
    //缩放模式参数，不保留宽高比
    int SCALE_FILL = 1;

    //sdk 初始化
    void initilize(@NonNull WsWebRTCParameters rtcParams, @NonNul
l WsWebRTCObserver rtcObs);
    //获取 context
    Context getContext();
      //开始播放
    void start();
    //暂停播放
    void pause();
    //停止播放
    void stop();
    //释放资源
    void uninitilize();
    //静音，true/false
    void mute(boolean isMute);
```

```
    //设置音量，0.0~10.0，默认 1.0
    void setVolume(double volume);
    //设置画面旋转角度:90/180/270
    void setVideoRotation(int degree);
    //设置缩放模式 SCALE_FIT/SCALE_FILL
    void setScaleType(int scaleType);
    //截图接口
    void snapshot(@NonNull WsWebRTCView.SnapshotListener listener,
 float scale);
    //截图回调
    public interface SnapshotListener {
        void onSnapshot(Bitmap bitmap);
    }
      ...
}
```

四、sdk 初始化

1. 构造初始化参数 WsWebRTCParameters 类对象

```
WsWebRTCParameters webrtcParam = new WsWebRTCParameters();
//设置客户 id,由网宿分配给客户的 id 字符串
webrtcParam.setCustomerID(customerId);
//设置播放流
webrtcParam.setStreamUrl(streamUrl);
//设置是否使用 dtls 加密，默认加密，false：加密；true：不加密
webrtcParam.disableDTLS(false);
//设置视频是否使用硬解，默认硬解，false：软解；true：硬解
webrtcParam.enableHw(true);
//设置音频使用格式,默认 OPUS,音频格式类型见 WsWebRTCParameters 类定义
webrtcParam.setAudioFormat(WsWebRTCParameters.OPUS);
//设置连接超时时间，默认 5s，单位 ms
webrtcParam.setConnTimeOutInMs(5000);
//设置音频 JitterBuffer 队列最大报文数，影响时延，默认 50
webrtcParam.setAudioJitterBufferMaxPackets(50);
//设置是否开启追帧，默认开启，false：不开启；true：开启
webrtcParam.enableAudioJitterBufferFastAccelerate(true);
//设置统计值回调频率，默认 1s，单位 ms
webrtcParam.setPortalReportPeriodInMs(1000);
//设置 rtc 日志等级，默认 LOG_NONE，日志等级类型见 WsWebRTCParameters
类定义
webrtcParam.setLoggingLevel(WsWebRTCParameters.LOG_INFO);
//设置 rtc 日志回调函数，loggable：WsWebRTCParameters.Loggable 对象，
见下文说明
webrtcParam.setLoggable(loggable);
```

2. 构造初始化参数 WsWebRTCObserver 回调类对象

```
WsWebRTCObserver observer = new WsWebRTCObserver();
```

3. 初始化

```
webrtcView.initilize(webrtcParam, observer);
```

4. WsWebRTCParameters 类说明

```
public class WsWebRTCParameters {
      //音频格式参数
    public static final int OPUS = 1;
    public static final int AAC_LATM = 2;
    public static final int AAC_ADTS = 4;
    //rtc 日志等级
    public static final int LOG_VERBOSE = 0;
    public static final int LOG_INFO = 1;
    public static final int LOG_WARNING = 2;
    public static final int LOG_ERROR = 3;
    public static final int LOG_NONE = 4;

    public String getCustomerID();
    public String getStreamUrl();
    public int getLoggingLevel();
    public WsWebRTCParameters.Loggable getLoggable();
    public boolean isEnableHw();
    public boolean isDisableDTLS();
    public int getAudioFormat();
    public String getAudioFormatString();
    public int getConnTimeOutInMs();
    public int getPortalReportPeriodInMs();
    public double getDefaultVolume();
    public int getAudioJitterBufferMaxPackets();
    public boolean isEnableAudioJitterBufferFastAccelerate();

    public interface Loggable {
        void onLogMessage(String tag, int level, String message);
    }
    ...
}
```

5. WsWebRTCObserver 类说明

```
public interface WsWebRTCObserver {
      //异常回调
    void onWsWebrtcError(String err, WsWebRTCObserver.ErrCode cod
e);
      //收到首包数据，0:audio, 1:video
    void onFirstPacketReceived(int mediType);
      //首帧渲染
    void onFirstFrameRendered();
    //卡顿状态通知，0：消除卡顿，1：发生卡顿
```

```
    void onNotifyCaton(int status);
      //分辨率切换
    void onResolutionRatioChanged(int width, int height);
      //统计数据回调，WsWebRTCPortalReport 类见下文说明
    void onPortalReport(WsWebRTCPortalReport webRTCPortalReport);

      //异常码
    public static enum ErrCode {
        ERR_CODE_SIGNAL_TIMEOUT,    //信令协商超时
        ERR_CODE_SIGNAL_REFUSE,           //信令被拒绝
        ERR_CODE_WEBRTC_CONN_FAILED,//rtc 连接失败
        ERR_CODE_WEBRTC_DISCONN,    //rtc 断开连接

        private ErrCode() {
        }
    }
    ...
}
```

6. 统计 WsWebRTCPortalReport 类说明

```
public class WsWebRTCPortalReport {
    private static final String TAG = "WsWebRTCPortalReport";

    //video stats
    public long   mFirstVideoPacketDelayMs;           //从首包时间
    public long   mFirstFrameRenderDelayMs;     //从首屏时间
    public float  mVideoDecodeFps;                    //解码帧率
    public float  mVideoDecoderAvgFps;                //平均帧率
    public float  mVideoRenderFps;              //渲染帧率
    public long   mVideoRenderReceived;         //渲染收到的帧率
    public long   mVideoRenderDropped;          //时丢弃的帧数
    public long   mTotalFrozenTimeMs;           //总卡顿时长
    public float  mFrozenRate;                        //总卡顿时长/播放时长
    public long   mVideoBitrate;                      //视频码率
    public long   mFramesDecoded;                     //解码帧数
    public long   mFramesDropped;                     //丢帧数
    public long   mFramesReceived;                    //接收帧数
    public int    mVideoPacketsLost;                  //丢包个数
    public long   mVideoPacketsReceived;        //接收包数
    public long   mFrameWidth;                              //视频宽度
    public long   mFrameHeight;                       //视频高度
    public long   mVideoDelayMs;                      //视频延迟
    public long   mVideoJitterBufferDelayMs;    //视频缓存延迟
    public long   mVideoNacksSent;
```

```
    public long   mRTT;                                        //rtp
 rtt

    //audio stats
    public long   mFirstAudioPacketDelayMs;           //从启动到收
到第一包音频数据的延时
    public int    mAudioPacketsLost;                  //丢包个数
    public long   mAudioPacketsReceived;        //接收包数
    public long   mAudioBitrate;                      //音频码率
    public long   mAudioDelayMs;
    public long   mAudioJitterBufferDelayMs;
    public long   mAudioNacksSent;

    //play stats
    public long   mAverageBitRate;                    //平均码率
    public long   mPlayTimeMs;                                 //播放
时长
    ...
}
```

五、异常回调处理

```
void onWsWebrtcError(String err, WsWebRTCObserver.ErrCode code) {
      ...
      webrtcView.stop();
      webrtcView.uninitilize();
}
```

六、sdk 使用

1. 启动 sdk

```
webrtcView.start();
```

2. 暂停播放

```
webrtcView.pause();
```

3. 重新播放

```
webrtcView.start();
```

4. 退出播放

```
webrtcView.stop();
```

5. 释放资源

```
webrtcView.uninitilize();
```