快直播LEB WebRTC Android SDK接入文档

1. SDK简介

快直播（Live Event Broadingcasting）腾讯基于WebRTC技术的超低延时直播，通过LEBWebRTC Android SDK，接入商只需对接几个接口，快速实现Android平台实现接入和播放。

2. SDK集成接入

2.1 jcenter接入方式

SDK提供两种方方式接入入: jcenter 和 AAR ， 可以根据需要选择接入入方方式，分别如下:

首先，在app的最上层 build.gradle 中加入入 jcenter的仓库依赖

```
buildscript {
    repositories {
        jcenter()
    }
}
```

然后，在相关module的build.gradle中加入入依赖

```
dependencies {
   implementation 'com.tencent.xbright:lebwebrtcsdk:1.0.0'
}
```

2.2 AAR接入入方方式

获取SDK的压缩包之后，把 lebwebrtcsdk-release.aar 文文件拷贝贝到app工工程的libs目目录下面面，然后在相关

module的 build.gradle 中添加对aar文文件的依赖

```
dependencies {
    implementation files('lebwebrtcsdk-release.aar')
}
```

2.3 SO库的ABI说明

SDK内包含SO库，目目前仅支支持 armeabi-v7a 、 arm64-v8a 两种ABI架构。

2.4 权限

需要配置一一些必要的权限才能正常运行行行，请确认app的 AndroidManifest.xml 中添加了了如下权限:

```
<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_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

2.3 渲染硬件加速

如需提高高显示渲染效率，可以在 AndroidManifest.xml 中将硬件加速打开

```
 <application android:hardwareAccelerated="true">
```

3. SDK接口

SDK封装类为LEBWebRTCView

3.1 初始化

```
LEBWebRTCView mWebRTCView = findViewById(R.id.id_surface_view);
```

```
mWebRTCView.initilize(LEBWebRTCParameters rtcParams, LEBWebRTCEvents rtcEvents)
```

参数如下：

参数	类型	说明
parameters	LEBWebRTCParameters	配置参数，具体见下文
events	LEBWebRTCEvents	事件回调，具体见下

LEBWbRTCView构造见下面示例：

```
LEBWebRTCView mWebRTCView = findViewById(R.id.id_surface_view);
R.id.id_surface_view:
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/id_framelayout_cg"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">
    <com.tencent.xbright.lebwebrtcsdk.LEBWebRTCView
        android:id="@+id/id_surface_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />
</FrameLayout>
```

LEBWebRTCParameters构造见下面示例：

```
//创建参数对象
LEBWebRTCParameters mLEBWebRTCParameters = new LEBWebRTCParameters();
//设置播放码流链接，webrtc://xxxxx
mLEBWebRTCParameters.setStreamUrl(mWebRTCUrl);
//设置是否硬解，默认为硬解
mLEBWebRTCParameters.enableHwDecode(true);
//设置连接超时时间，默认为5000ms
mLEBWebRTCParameters.setConnectionTimeOutInMs(5000);
//设置播放状态回调事件周期，默认为1000ms
mLEBWebRTCParameters.setStatsReportPeriodInMs(1000);
//设置日志级别，默认为LOG_NONE
mLEBWebRTCParameters.setLoggingSeverity(LEBWebRTCParameters.LOG_NONE);
```

LEBWebRTCEvents事件回调定义如下：

```
public interface LEBWebRTCEvents {
    enum ConnectionState
    {
        // 开始建立连接
        STATE_BEGIN,
        // OFFER创建
        STATE_OFFER_CREATED,
        // ICE完成
        STATE_ICE_COMPLETED,
        // 连接建立
        STATE_WEBRTC_CONNECTED,
        // 渲染首帧
        STATE_FIRST_FRAME_RENDERED,
        // 连接超时
        STATE_WEBRTC_TIMEOUT,
    }

    // offer创建成功
    void onEventOfferCreated(String sdp);
    // 连接成功
    void onEventConnected();
    // 连接失败
    void onEventConnectFailed(ConnectionStats state, String error);
    // 断开断开
    void onEventFirstPacketReceived(int mediType);//0:audio, 1:video
    // 渲染首帧
    void onEventFirstFrameRendered();
    // 分辨率切换
    void onEventResolutionChanged(int width, int height);
    // 统计数据
    void onEventStatsReport(LEBWebRTCStatsReport webRTCStatsReport);
}
```

其中onEventStatsReport(LEBWebRTCStatsReport webRTCStatsReport)用来回调播放状态，包含音视频播放性能、播放帧率、码率和时长等数据，LEBWebRTCStatsReport定义如下：

```
public class LEBWebRTCStatsReport {
    //video stats
    public long   mFirstVideoPacketDelay; //从启动到收到第一包视频数据的延时
    public long   mFirstFrameRenderDelay; //从启动到首帧渲染延时
    public double mFramerate; //解码帧率
    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    mPacketsLost; //丢包个数
    public long   mFrameWidth; //视频宽度
    public long   mFrameHeight; //视频高度


    //audio stats
    public long   mFirstAudioPacketDelay;//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


    //play stats
    public double mAverageFrameRate;//平均帧率
    public long   mAverageBitRate;//平均码率
    public long   mPlayTime;//播放时长
}
```

3.1 启动过程

初始化后开始启动过程，步骤如下

1. 开始启动sdk，sdk在p2p未连接时会创建offer(local sdp)
   void startPlay()
2. Offer创建成功后回调，用户可以回调中实现向信令服务器发送offer，并获取remote sdp
   void onEventOfferCreated(String sdp)
3. 将remote sdp设置给SDK，sdk会发起p2p连接，连接成功后开始播放
   void setRemoteSDP(String sdp)

3. 2 暂停播放

//暂停播放，保持连接

void pausePlay()

3.3 继续播放

//在暂停后恢复播放
void startPlay()

3.3 退出播放

//退出播放、并断开连接
void stopPlay()

也可以在Activity的 onResume 、onPause、onStop 中启动、暂停和停止播放，并在其生命周期内进行相应处理:

```
public class LEBWebRTCACtivity extends AppCompatActivity {
    private LEBWebRTCView mWebRTCView;

    @Override
    protected void onResume() {
        super.onResume();
         mWebRTCView.startPlay();//开始播放
    }
    @Override
    protected void onPause() {
        super.onPause();
        mWebRTCView.pausePlay();//暂停播放
    }

    @Override
    protected void onStop() {
        super.onStop();
         mWebRTCView.stopPlay();//停止播放并断开连接
    }
}
```

3.4 释放资源

```
//释放SDK相关资源
void release()
```