阿里巴巴集团

SDK 参考手册

安卓媒体推流器 SDK 参考手册

POWERED BY QUPAI

文档修订历史

版本	作者	工作描述	修订历史	修改日期
1.0	德胜	安卓推流 SDK 文档		2016-06-21
1.1	德胜	安卓推流 SDK 文档		2016-07-02

参考文献

无

目录

文档说明

本文档面向所有使用该SDK的开发人员、测试人员以及对此感兴趣的用户，要求读者具有一定的Android开发能力。

1. 简介

安卓推流SDK是在安卓平台上使用的软件开发工具包(Software Development Kit)，为Android开发者提供简单易用的接口，帮助开发者实现Android平台上的推流应用开发。

1.1 功能说明

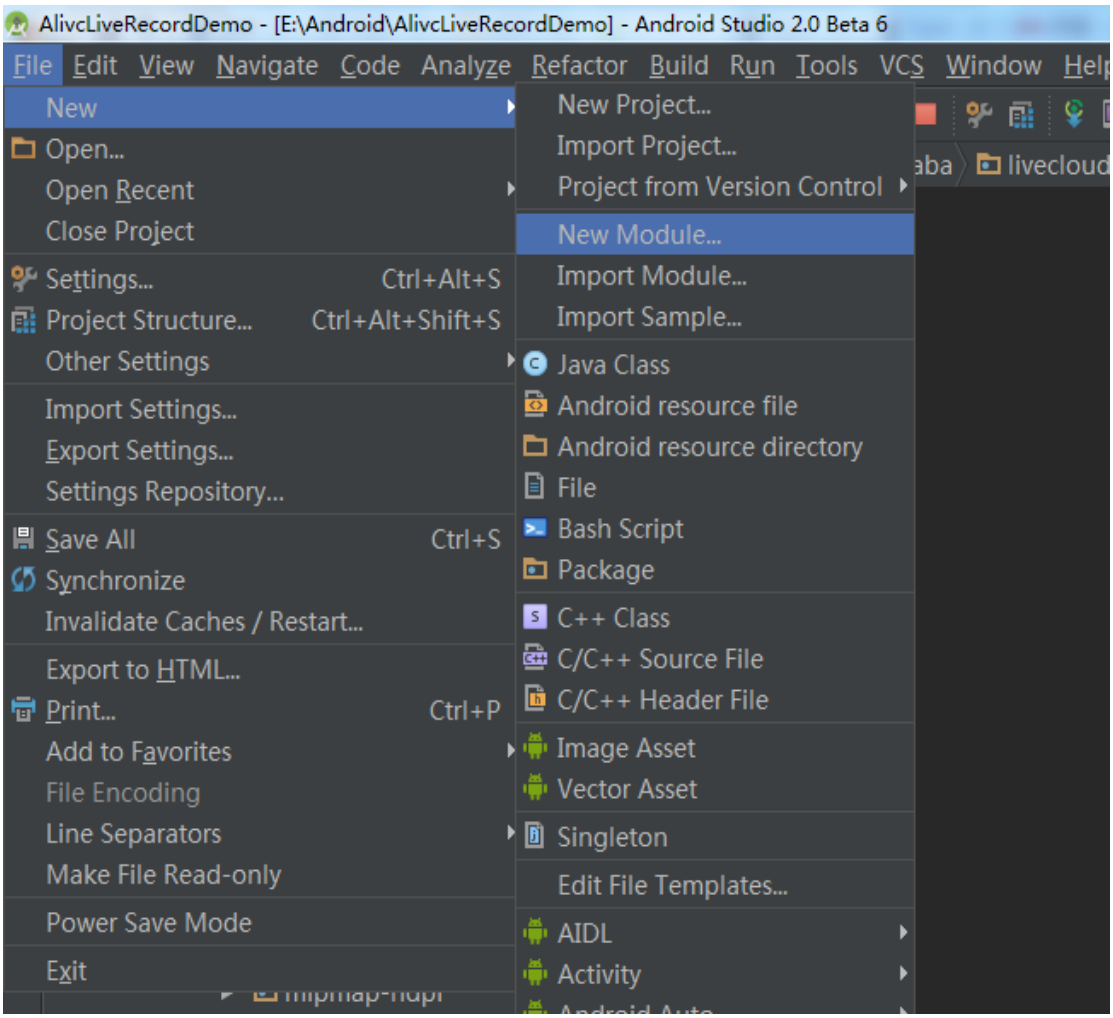
1. 方便快捷、低门槛实现媒体推流功能。用户无须关心内部实现细节，只需要自定义界面既可以实现专业级的推流应用。
2. 推流支持格式：rtmp
3. 编码目前为硬编

1.2 安装包说明

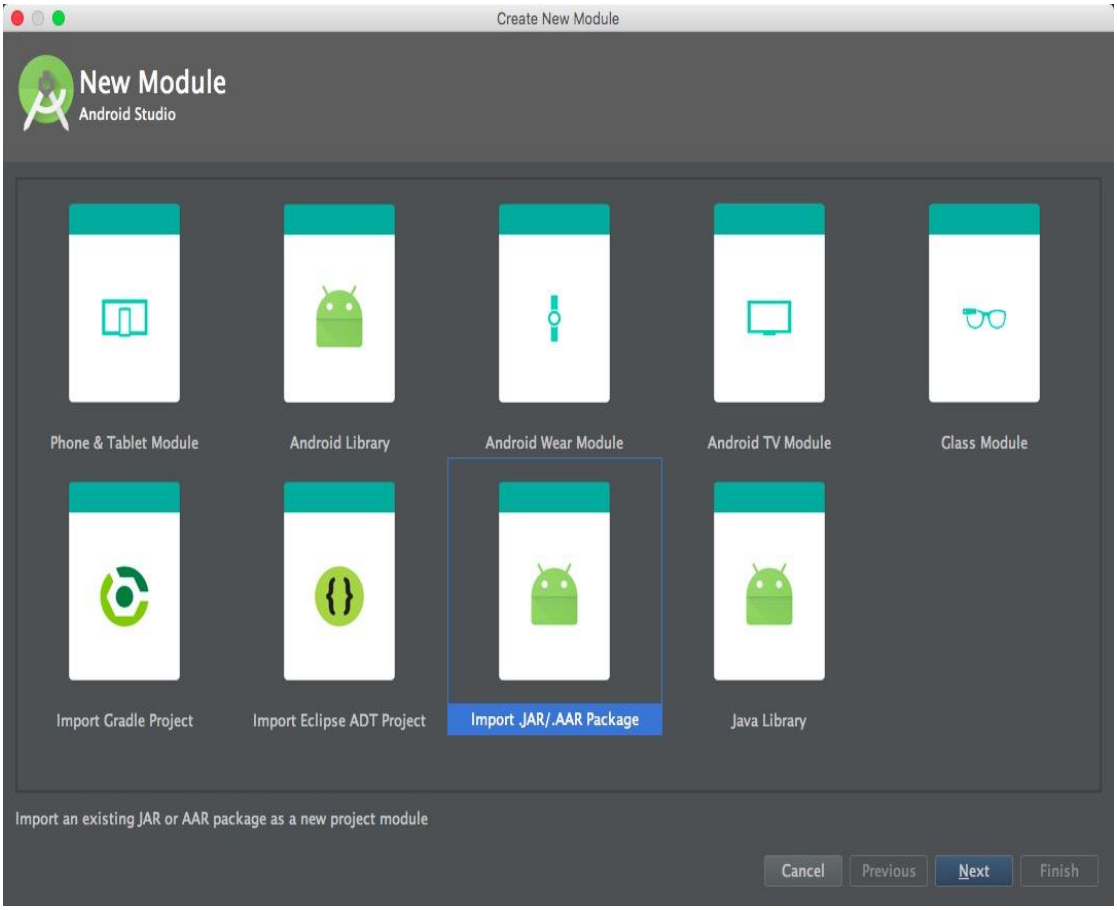
推流器SDK的完整下载包中包含test、doc、libs等：

1. test：主要存放了调用SDK的示例工程，可以帮助用户了解如何使用该SDK,其中已经包含SDK和demo示例.导入即可使用

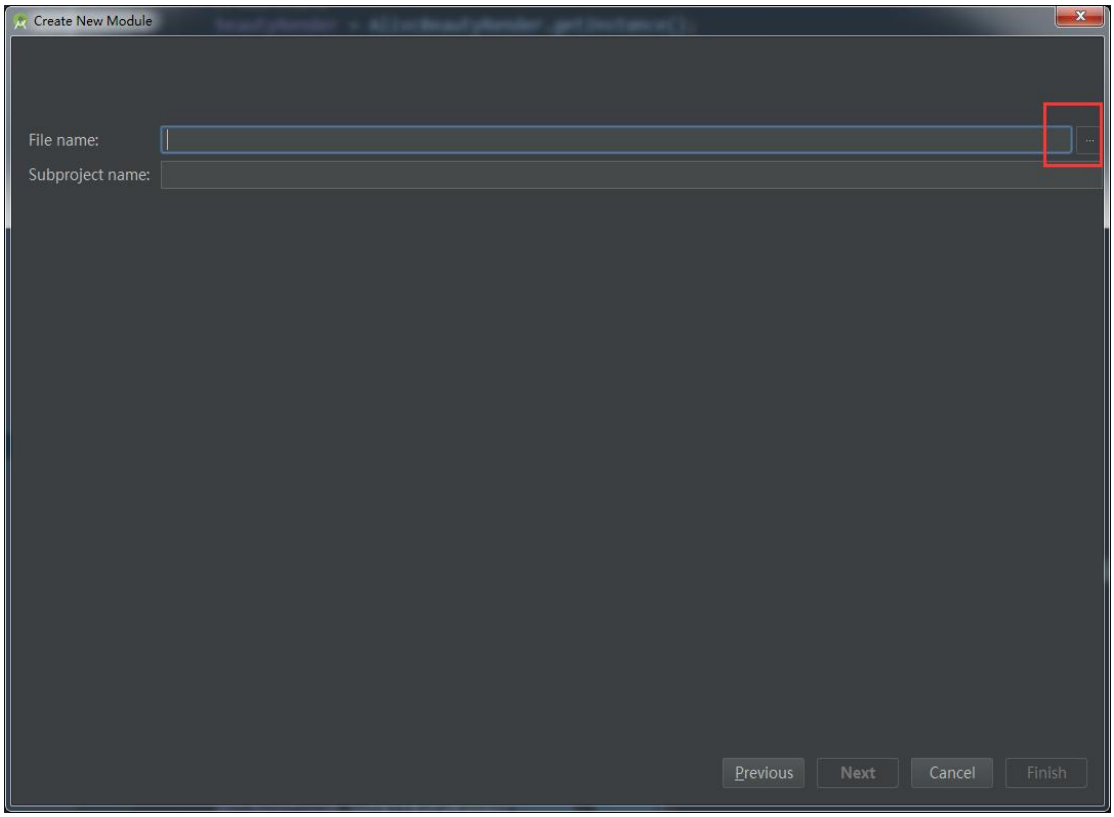
2. libs：其中有两个AAR分别是推流的SDK，和点赞的交互SDK，SDK开发包, SDK建议使用Android Studio进行集成，对于使用eclipse的开发者可以参考Google提供的迁移方法https://developer.android.com/studio/intro/migrate.html (此地址需要翻墙，也可自行百度解决）,将原应用迁移到Android Studio后再集成SDK。
   目前SDK包以AAR的形式给出。Android Studio用户只需采用导入Module的方式添加依赖AAR就可以了.
   在需要集成的应用中新建一个Module，如下图:



选择AAR方式导入，点击next。



导 入 AlivcLiveRecord-release1.0.0 ， 选 择 >> AlivcLiveRecord-release1.0.0.aar 文件，点击Finish。



3. doc：存放SDK相关接入文档。

1.3 推流器性能

1. 目前推流SDK推流采用的是硬编。

2. 推流采用FFMPEG推流

3. SDK的大小：去除ffmpeg动态库之后SDK对应用的大小增加在1M左右.加上ffmepg在2M左右

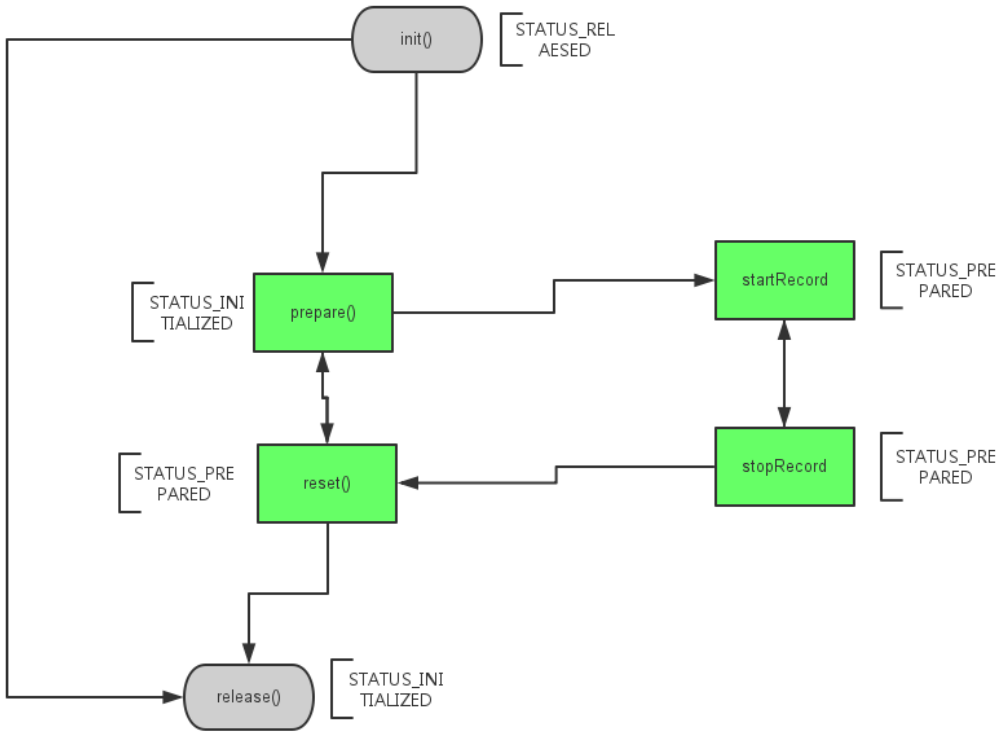
1.4 注意事项

1. 推流器SDK目前只支持单实例。不能够同时开2个推流实例，同时只能存在一个实例，需要另开实例的时候，需要关闭之前存在的实例。

2. 操作系统版本要求android4.3以上。

2. 系统框架

2.1 系统框架图



2.2 类框架图

无

3. 使用说明

3.1 开发环境配置

1. 需要准备Android运行环境,以及硬件CPU支持ARMv7或ARMv7s的安卓设备。

2. 权限开通，在阿里云上申请推流SDK开发权限。

3.2 开发步骤

首先在安卓应用程序中，需要声明以下权限：

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
   <uses-permission  android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_SETTINGS" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission  android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
```

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

使用用媒体推流器SDK的调用顺序为：

0. 调用AlivcMediaRecorderFactory.createMediaRecorder获取实例
1. 初始化mMediaRecorder.init(this);
2. 调用mMediaRecorder.prepare(mConfigure, mPreviewSurface);
3. 调用mMediaRecorder.startRecord(pushUrl);开始推流
4. 调用mMediaRecorder.stopRecord();停止推流
5. 调用mMediaRecorder.reset();释放预览资源,对应的是prepare
6. 调用mMediaRecorder.release();释放资源,对应的是init

3.3 Demo 示例

在 SDK 中提供了 Demo，此 Demo 是用推流 SDK 开发了一个完整的推流器，用户可以参考 Demo 进行推流的开发。

下面给出了部分重要的 Demo 中调用 SDK 的代码。

一、创建 SurfaceView 和 SurfaceView 的 Callbac

```
_CameraSurface = (SurfaceView) findViewById(R.id.camera_surface);
_CameraSurface.getHolder().addCallback(_CameraSurfaceCallback);
Private final SurfaceHolder.Callback _CameraSurfaceCallback = new
SurfaceHolder.Callback() {
@Override
```

```
        public void surfaceCreated(SurfaceHolder holder) {
            holder.setKeepScreenOn(true);
            mPreviewSurface = holder.getSurface();
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
int height) {
            mMediaRecorder.setPreviewSize(width, height);
            mPreviewWidth = width;
            mPreviewHeight = height;
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            mPreviewSurface = null;
            mMediaRecorder.stopRecord();
            mMediaRecorder.reset();
        }
    };
         mPreviewSurface = holder.getSurface();
```

二、创建推流器，准备推流：

1.创建实例初始化

```
        mMediaRecorder = AlivcMediaRecorderFactory.createMediaRecorder();
        mMediaRecorder.init(this);
        mMediaRecorder.setOnRecordStatusListener(mRecordStatusListener);
        mMediaRecorder.setOnNetworkStatusListener(mOnNetworkStatusListener);
        mMediaRecorder.setOnRecordErrorListener(mOnErrorListener);
```

2.开始预览

```
        private Map<String, Object> mConfigure = new HashMap<>();
        mConfigure.put(AlivcMediaFormat.KEY_CAMERA_FACING, cameraFrontFacing);
        mConfigure.put(AlivcMediaFormat.KEY_MAX_ZOOM_LEVEL, 3);
        mConfigure.put(AlivcMediaFormat.KEY_OUTPUT_RESOLUTION, resolution);
```

```
mMediaRecorder.prepare(mConfigure, mPreviewSurface);
```

3. 开始推流

```
mMediaRecorder.startRecord(pushUrl);
```

4. 开启关闭美颜滤镜

```
mMediaRecorder.addFlag(AlivcMediaFormat.FLAG_BEAUTY_ON);//开启美颜
mMediaRecorder.removeFlag(AlivcMediaFormat.FLAG_BEAUTY_ON);//关闭美颜
```

5. 切换摄像头

```
mMediaRecorder.switchCamera();
```

6. 对焦

```
mMediaRecorder.focusing(x, y);// demo 提供了首次对焦+手动对焦
```

7. 缩放

```
mMediaRecorder.setZoom(scaleGestureDetector.getScaleFactor(), null);
```

8. 结束推流

```
mMediaRecorder.stopRecord();
```

9. 释放资源

```
mMediaRecorder.reset();//释放预览资源
mMediaRecorder.release();//释放推流资源
```

三.网络状态事件通知-- OnNetworkStatusListener

OnNetworkStatusListener:

```
/**
 * 网络较差时的回调，此时推流 buffer 为满的状态，会执行丢包，此时数据流不能正常推送
 */
void onNetworkBusy();
/**
 * 网络空闲状态，此时本地推流 buffer 不满，数据流可正常发送
 */
void onNetworkFree();
/**
 * @param       status
 */
void onConnectionStatusChange(int status);
/**
```

```
   * 网络重连
   * @return  false:停止重连  true:允许重连
     */
  boolean onNetworkReconnect();
```

四、错误事件通知中：

```
Private      OnLiveRecordErrorListener      mOnErrorListener      =      new
OnLiveRecordErrorListener() {
       @Override
       public void onError(int errorCode) {
           switch (errorCode) {
               case AlivcStatusCode.ERROR_SERVER_CLOSED_CONNECTION:
               case AlivcStatusCode.ERORR_OUT_OF_MEMORY:
               case AlivcStatusCode.ERROR_CONNECTION_TIMEOUT:
               case AlivcStatusCode.ERROR_BROKEN_PIPE:
               case AlivcStatusCode.ERROR_ILLEGAL_ARGUMENT:
               case AlivcStatusCode.ERROR_IO:
               case AlivcStatusCode.ERROR_NETWORK_UNREACHABLE:
                   Log.i(TAG, "Live stream connection error-->" + errorCode);
                   ToastUtils.showToast(LiveCameraActivity.this,"Live    stream
connection error-->" + errorCode);

                   break;

               default:
           }
       }
   };
```

4.接口说明

SDK中提供了两个类AlivcMediaRecorder和AlivcMediaRecorderFactory，

其中AlivcMediaRecorder是推流SDK使用接口类，AlivcMediaRecorderFactory用来创建推流器器AlivcMediaRecorder。同时提供了多个事件通知接口，用来监听推流器的各种状态。

类名	功能
AlivcMediaRecorderFactory	创建推流器接口
AlivcMediaRecorder	推流功能接口类
AlivcMediaFormat	推流器可配参数类
OnNetworkStatusListener	推流状态监听接口
OnRecordStatusListener	预览状态监听接口
OnLiveRecordErrorListener	推流错误信息监听接口

4.1 AlivcMediaRecorderFactory

类名：AlivcMediaRecorderFactory

功能：创建推流器接口 AlivcMediaRecorder

成员：

成员	功能
`createMediaRecorder`	创建推流器 AlivcMediaRecorder

详细说明：

```
AlivcMediaRecorderFactory.createMediaRecorder()
```

createMediaRecorder用来创建推流器，返回AlivcMediaRecorder类。

参数:

- null

返回值：返回空为错误，正确则为有效的AlivcMediaRecorder值。

4.2 AlivcMediaRecorder

类名：AlivcMediaRecorder

功能：推流器接口类 AlivcMediaRecorder，提供推流控制

成员：

成员	功能
init	初始化推流器
prepare	开始预览
startRecord	开始推流
switchCamera	切换摄像头
stopRecord	结束推流
reset	释放预览资源,对应的是 prepare
focusing	对焦
setZoom	缩放
setPreviewSize	设置预览大小
addFlag	添加美颜
removeFlag	移除美颜
release	释放资源对应: init
setOnRecordErrorListener	设置推流错误回调
setOnRecordStatusListener	设置推流的状态回调监听

setOnNetworkStatusListener	设置网络状态的回调监听

下面详细介绍一下各个成员函数的具体使用：

1. 初始化推流器

   ```
   void init(Context context);
   ```
   参数：

   context：Android 上下文

2. 开始预览

   ```
   /**
   * 开始预览
   */
   void prepare(Map<String, Object> params, Surface surface);
   ```
   参数：

   params: 推流过程中不可动态改变的参数.

   surface:

2. 开始推流

   ```
   void startRecord(String outputUrl);
   ```
参数：

   outputUrl 表示推流地址。

备注：在 prepare 完成之后调用 startRecord 进行推流。

3. 结束推流

   ```
   void stopRecord();
   ```

4. 切换摄像头

   ```
   void switchCamera();
   ```
5. 释放资源

   ```
   void reset();
   ```
   备注：释放预览资源,对应的是 prepare

6. 手动对焦

   ```
   void focusing(float xRatio, float yRatio);
   ```
   备注： xRatio 横向坐标点所占的比例

   yRatio 纵向坐标点所占的比例

7. 缩放

```
void setZoom(float scaleFactor,CaptureRequest.OnCaptureRequestResultListener listener);
```

备注：scaleFactor 参数为缩放比例

listener 参数是传递

8. 设置预览大小

```
void setPreviewSize(int width, int height);
```

备注：width 预览宽，height 预览高

9. 添加美颜

```
void addFlag(int flag);
```

参数：flag 目前改 falg 只提供了美颜,方便将来扩展滤镜之类

10. 释放资源

```
void release();
```

11. 设置推流错误回调

```
Void setOnRecordErrorListener(OnLiveRecordErrorListener listener);
```

备注：回调具体的信息在下面介绍

12. 设置推流的状态回调监听

```
void setOnRecordStatusListener(OnRecordStatusListener listener);
```

备注：回调具体的信息在下面介绍

13. 设置网络状态的回调监听

```
void setOnNetworkStatusListener(OnNetworkStatusListener listener);
```

备注：回调具体的信息在下面介绍

4.3 OnNetworkStatusListener

当调用startRecord()后，推流器开始推流，当前的网络连接状态会返回，用户需要注册该事件,以便获取到该时间通知，在结束和开始的状态都会给予返回.

```
public interface OnNetworkStatusListener {

  /**

   * 网络较差时的回调，此时推流buffer为满的状态，会执行丢包，此时数据流不能正常推送

   */

  void onNetworkBusy();

   * 网络空闲状态，此时本地推流buffer不满，数据流可正常发送

   */

  void onNetworkFree();

  /**

   * @param       status

   */

  void onConnectionStatusChange(int status);

  /**

   * 网络重连

   * @return  false:停止重连  true:允许重连

    */
```

```
  boolean onNetworkReconnect();
}
```

4.4 OnRecordStatusListener

当prepare完成后，会发出该事件通知消息，用户需要注册该事件，采集开始得到摄像头的状态以便做一些事情。比如进入对焦一次.

```
public interface OnRecordStatusListener {
    /*** 摄像头打开成功     */
    void onDeviceAttach();
    /*** 开启预览成功     */
    void onSessionAttach();
    /**     * 停止预览     */
    void onSessionDetach();
    /*** 关闭摄像头     */
    void onDeviceDetach();
    /**摄像头打开失败**/
    void onDeviceAttachFailed(int facing)
   /** 输出分辨率设置过大，没有合适的预览分辨率选择 **/
    void onIllegalOutputResolution()
}
```

4.5 OnLiveRecordErrorListener

当开始推流，返回给开发者的错误信息。

```
public interface OnLiveRecordErrorListener {

  void onError(int errorCode);

}
```

参数：

1. errorCode：获取到的错误类型Code：

    - ERORR_OUT_OF_MEMORY：内存不足
    - ERROR_IO：I/O错误
    - ERROR_BROKEN_PIPE：管道中断
    - ERROR_ILLEGAL_ARGUMENT：非法参数
    - ERROR_NETWORK_UNREACHABLE：网络不可达
    - ERROR_SERVER_CLOSED_CONNECTION 服务端断开链接
    - ERROR_CONNECTION_TIMEOUT 网络超时

5. 注意事项

无

6. 版权声明

p

版权所有，切勿盗版