安卓推流 SDK 是在安卓平台上使用的软件开发工具包 (Soft 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。

功能说明

- 支持推流到主流 rtmp 服务器
- 支持 H264 和 AAC 编码
- 支持美颜特效
- 支持对焦
- 支持缩放操作
- 支持闪光灯开关
- 支持添加水印操作 且支持水印添加至任意位置
- 支持镜像操作
- 支持摄像头的切换
- 支持固定横竖屏推流
- 支持静音推流

阅读对象

- 具有基本的 Android 开发能力的开发者
- 要求开发者对直播推流的基本功能有一定的了解

特别说明

- 仅支持 anroid4.3 及以上系统版本
- 仅支持智能设备 CPU为 Armv7 或者 Armv8 架构、X86、MIPS 或者其他架构 CPU 暂不支持
- 视频编码器采用 Android 硬件编码
- 推流 SDK 为单实例，包大小为 2M 左右
- SDK 建议使用 Android Studio 进行开发集成

准备工作

- 获取 **推流 SDK**：下载地址，其中 SDK 包含以下文件：
    - demo: 调用 SDK 的示例工程，开发者可参考 demo 代码进行开发集成
    - libs: SDK 开发包，包含 jar 包，需要在工程中进行引用
    - doc: 相关接入文档。
- 确保已开通视频直播服务，可参阅文档：快速开始

- 准备 Android Studio 开发环境

开发环境配置

> 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 将原应用迁移到Android Studio 后再集成SDK。

sdk/jniLibs: 底层动态链接库，需要将 jniLibs 整个拷贝到 moudle-name/src/main 中，然后在 module 的 build.gradle 中加入以下配置（具体可参考 Demo）： ```javascript splits { abi { enable true reset() include 'armeabi-v7a' }

```
}
```

```
> 注意：如果使用其他的第三方库也有对 .so 的依赖，则需要选取其他第三方库的 *armeabi-v7a* 的 .so，一并放入 *moudle-name/src/main/jniLibs/armeabi-v7a* 中，如果其他第三方库没有给出 armeabi-v7a 的 .so，则可以使用其 armeabi 的 .so 替代。



##  系统框架

### 推流接口框架图

![推流接口](http://docs-aliyun.cn-hangzhou.oss.aliyun-inc.com/assets/pic/45265/cn_zh/1490854206895/%E7%9B%B4%E6%92%AD%E6%8E%A8%E6%B5%81%E6%8E%A5%E5%8F%A3%E6%B5%81%E7%A8%8B%E5%9B%BE.png)

##开发接入##
1\. 权限声明

 在 AndroidManifest.xml 中，声明以下权限：
```javascript
<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" />
```
```

2. 创建推流器实例

创建 SurfaceView 和 SurfaceView 的 Callback

```
_CameraSurface = (SurfaceView) findViewById(R.id.camera_surface);
_CameraSurface.getHolder().addCallback(_CameraSurfaceCallback);
private final SurfaceHolder.Callback _CameraSurfaceCallback = new SurfaceHolder.Ca
llback() {
    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        holder.setKeepScreenOn(true);
        mPreviewSurface = holder.getSurface();
    }

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

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

创建实例初始化

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

3\. 推流

开始预览

```javascript
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);
```

开始推流
```javascript
//pushUrl是通过以下步骤得到的Url：
//1.阿里云控制台的域名管理页面
//2.点击“管理”
//3.复制基本信息页中的推流地址
//4.鉴权配置中做过鉴权的Url
mMediaRecorder.startRecord(pushUrl);
```

4\. 媒体控制

开启关闭美颜滤镜
```javascript
mMediaRecorder.addFlag(AlivcMediaFormat.FLAG_BEAUTY_ON);//开启美颜
mMediaRecorder.removeFlag(AlivcMediaFormat.FLAG_BEAUTY_ON);//关闭美颜
```
切换摄像头

```javascript
//Demo默认为前置摄像头
mMediaRecorder.switchCamera();
```
对焦

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

```javascript
mMediaRecorder.setZoom(scaleGestureDetector.getScaleFactor());
```

添加水印

```javascript
 //创建水印信息对象
 mWatermark = new AlivcWatermark.Builder()
                    .watermarkUrl(bundle.getString(WATERMARK_PATH)) //水印图片地址
                    .paddingX(bundle.getInt(WATERMARK_DX))                    //水印图
片在x轴的偏移
                    .paddingY(bundle.getInt(WATERMARK_DY))                    //水印图
```

片在y轴的偏移
                    .site(bundle.getInt(WATERMARK_SITE))   //
水印图片位置
                    .build();

mConfigure.put(AlivcMediaFormat.KEY_WATERMARK, mWatermark); //配置水印信息Watermark)
; //配置水印信息
```

5\. 事件监听

网络状态事件通知-- OnNetworkStatusListener:
 ```javascript
private OnNetworkStatusListener OnNetworkStatusListener =new OnNetworkStatusListen
er(){
/**
 * 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){}

  /**
   * 重连失败
   * @return false：停止重连 true：继续重连
   * 说明：ｓｄｋ检测到检测到需要重连时将会自动执行重连，直到重连成功
   *或者重连尝试超时，
   * 超时时间可以通过{@link AlivcMediaFormat#KEY_RECONNECT_TIMEOUT}设置
   * 默认为５ｓ，超时后将触发此回调，若返回true表示继续开始新的一轮尝试，
   *返回false，
   * 表示不再尝试
   */
  boolean onNetworkReconnectFailed(){}
}

```

错误事件通知中:
```javascript
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 connecti
on error-->" + errorCode);
                 break;
             default:
        }
    }
};
```

6\. 结束推流
```javascript
mMediaRecorder.stopRecord();     //结束推流
mMediaRecorder.reset();          //释放预览资源
mMediaRecorder.release();        //释放推流资源
```

进阶开发
分辨率、码率、帧率配置
影响画质的因素：分辨率 码率 帧率
- 分辨率：直播推流SDK提供多分辨率选择：240P、360P、480P、540P、720P、1080P
- 码率：编码器每秒编码的数据量，推流器SDK码率单位为kbps
- 帧率：FPS是图像领域中的定义，是指画面每秒传输帧数，通俗来讲就是指视频每秒的画面数

针对视频清晰度和码率分辨率设置提供一份参考，开发者可以通过应用属性来平衡该修改。

码率类型	计算公式	240P	360P	480P	540P	720P
极低码率	(宽X高X3)/4	60kbps	120kbps	250kbps	500kbps	1024kbps
低码率	(宽X高X3)/2	120kbps	250kbps	500kbps	1024kbps	2048kbps
中码率	(宽X高X3)	250kbps	500kbps	1024kbps	2048kbps	4096kbps
高码率	(宽X高X3)X2	500kbps	1024kbps	2048kbps	4096kbps	6144kbps
超高码率	(宽X高X3)X4	1024kbps	2048kbps	4096kbps	6144kbps	8192kbps

> 1080P等高分辨率需要足够的显示面积，手机等设备由于显示屏较小，不推荐使用1080P分辨率。

在开始推流前需要调用prepare(Map params, Surface surface)，配置分辨率、码率、帧率：

```javascript
mConfigure.put(AlivcMediaFormat.KEY_FRAME_RATE, 25);
                //配置帧率
mConfigure.put(AlivcMediaFormat.KEY_OUTPUT_RESOLUTION, AlivcMediaFormat.OUTPUT_RES
OLUTION_480P); //配置分辨率
mConfigure.put(AlivcMediaFormat.KEY_MAX_VIDEO_BITRATE, 500000);
                //配置码率 500kbps
mConfigure.put(AlivcMediaFormat.KEY_INITIAL_VIDEO_BITRATE, 500000);
                //配置码率 500kbps
mMediaRecorder.prepare(mConfigure, mPreviewSurface);
```

更多配置参数可查阅：[接口说明](~~50075~~)

限定符和类型	字段	说明
static int	KEY_FRAME_RATE	帧率
static int	KEY_I_FRAME_INTERNAL	关键帧间隔
static int	KEY_INITIAL_VIDEO_BITRATE	初始码率 （单位：bps）
static int	KEY_MAX_VIDEO_BITRATE	最大码率 （单位：bps）
static int	KEY_MAX_ZOOM_LEVEL	最大缩放级别 > 0
static int	KEY_MIN_VIDEO_BITRATE	最小码率 （单位：bps）
static int	KEY_OUTPUT_RESOLUTION	输出分辨率 value： OUTPUT_RESOLUTION_240P OUTPUT_RESOLUTION_360P OUTPUT_RESOLUTION_480P OUTPUT_RESOLUTION_540P OUTPUT_RESOLUTION_720P OUTPUT_RESOLUTION_1080P