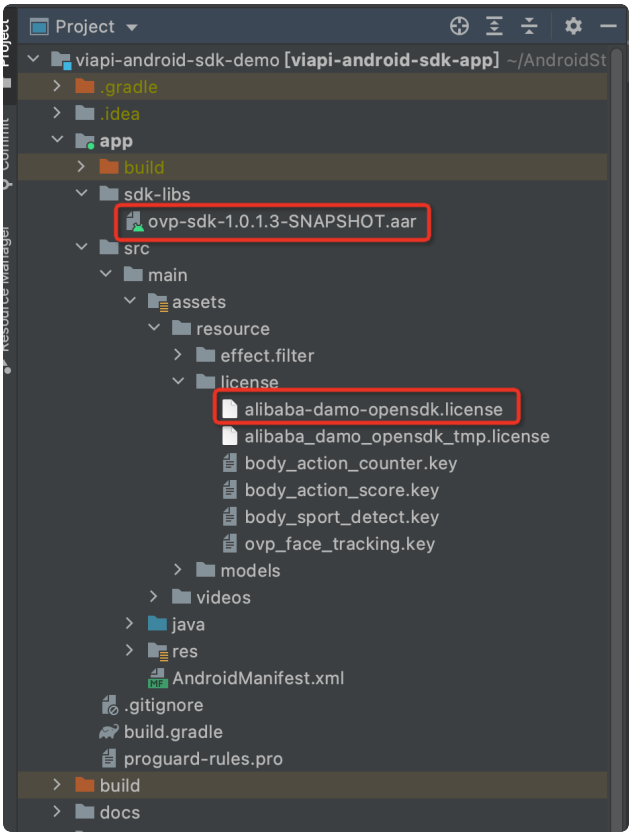
Android端opensdk滤镜使用集成说明文档

一、Android stuido配置工程

1、获取相关资源压缩包（由官网线上购买申请或阿里云相关人员提供下载链接）后，解压压缩包，可看到如下资源文件，demo示例工程、支持相关能力的aar及支持相关能力的license文件。如下图：



注意：alibaba-damo-opensdk.license为正式证书（官网下载获取的都是正式证书），_tmp结尾的为临时证书一般线下提供，临时证书不能改名，正式license可以改名字，但是不能与tmp license重名。两个证书只需要调用其中一个进行鉴权就可以。

二、SDK调用步骤，功能实现

2.1 SDK初始化

接口描述：

算法API使用前先调用SDK初始化接口，初始化之后，各功能才可以正常使用，否则会引起鉴权等异常，初始化建议放在app进程启动时Application onCreate中进行。

初始化接口：

Plain Text | 复制代码

```
VIAPICreateApi.getInstance().getVIAPISdkCore().init(Context context,boolean isDebug);
```

参数说明：

Context context 应用上下文。

boolean isDebug SDK调试开关。

返回值 ：

int类型，返回0为初始化成功，其它返回为初始化失败。

具体代码示例如下：

Plain Text | 复制代码

```
  private void initSDK() {
    int status = VIAPICreateApi.getInstance().getVIAPISdkCore().init(this,false);
    if (status != 0) {
        Toast.makeText(this, VIAPIStatusCode.getErrorMsg(status), Toast.LENGTH_LONG).show();
    } else {
        Toast.makeText(this, "初始化成功！", Toast.LENGTH_LONG).show();
    }
}
```

2.2 滤镜API使用

2.2.1 滤镜算法实例对象

OVFilter：在需要用到滤镜算法的地方，创建OVFilter滤镜实例，OVFilter是滤镜API接口对象，通过此对象可以完成camera实时视频流滤镜能力的使用。

算法实例化接口：

Plain Text | 复制代码

```
//滤镜
OVFilter mOVFilter = new OVFilter();
```

参数说明：

无

返回值:

无

具体代码示例如下：

```
Plain Text | 复制代码
1  private final OVFilter mOVFilter;
2  mOVFilter = new OVFilter();
```

2.2.2 单个能力license鉴权

接口描述：

license证书验签接口，验签通过后才能成功调用算法。

license证书验签接口：

```
Plain Text | 复制代码
1  mOVFilter.nativeCheckLicense(String licensePath);
```

获取证书路径(具体实现可以参考demo)：

```
Plain Text | 复制代码
1  String licensePath = AssetsProvider.getReleaseLicenseFilePath();
```

参数说明：

String licensePath 传入全局证书license路径或自定义的单个能力license路径。

注：如果所有能力使用同一个全局证书默认传入全局证书路径即可，如果接入方有针对此能力的单独证书，则需要传入单独的证书文件的绝对路径。

返回值：

int类型，返回0为验签成功，其它返回为验签失败。

具体代码示例如下：

```
Plain Text | 复制代码
1  int status = mOVFilter.nativeCheckLicense(licensePath);
```

2.2.3 传入滤镜能力的文件

接口描述：

传入滤镜所需的资源文件。

传入文件接口：

```
Plain Text | 复制代码
1  mOVFilter.nativeFilterCreate(String ResourcePath);
```

参数说明：

String ResourcePath 传入滤镜算法所需的资源文件，传资源文件上层文件夹的路径。

返回值：

int类型，返回0为成功，其它返回为失败。

具体代码示例如下：

```
Plain Text | 复制代码
1  int status = mOVFilter.nativeFilterCreate(ResourcePath);
```

2.2.4 设置具体的某个滤镜能力

接口描述：

设置具体的某个滤镜能力（如活力、清新等）

传入接口：

```
Plain Text | 复制代码
1  mOVFilter.nativeFilterSetEffectPath(String filterResRootDir + File.separato
   r + filterId);
```

参数说明：

String filterResRootDir 滤镜资源的根目录；
filterId 具体滤镜对应的值 0(正常),120(活力),121(清新),122(美食),123(日系),124(美颜),125(薄荷),126(黑白),127(增强)；

返回值：

int类型，返回0为成功，其它返回为失败。

具体代码示例如下：

```
Plain Text | 复制代码
1  int status = mOVFilter.nativeFilterSetEffectPath(filterResRootDir + File.se
   parator + "127");
```

2.2.5 设置具体的某个滤镜能力（用于滤镜间切换）

接口描述：

用于具体的滤镜能力间的切换（如活力、清新等）

传入接口：

```
Plain Text | 复制代码
1  mOVFilter.nativeFilterSetEffectId(int filterId);
```

参数说明：

filterId 具体滤镜对应的值 0(正常),120(活力),121(清新),122(美食),123(日系),124(美颜),125(薄荷),126(黑白),127(增强)；

返回值：

int类型，返回0为成功，其它返回为失败。

具体代码示例如下：

```
Plain Text | 复制代码
1  int status = mOVFilter.nativeFilterSetEffectId(127);
```

2.2.6 滤镜处理接口

接口描述：

通过此接口使滤镜能力生效

传入接口：

```
Plain Text | 复制代码
1  mOVFilter.nativeFilterProcessTexture(int textureId, int width, int height,
   int tex_type, int data_type, int format, int rotate);
```

参数说明：

int id：需要处理并绑定数据的纹理id（此id指opengl es纹理绑定id）；
int width：传入帧数据的宽；
int height：传入帧数据的高；
int tex_type：纹理id的类型OV_TEXTURE_OES、OV_TEXTURE_2D、OV_TEXTURE_3D；
int data_type：传入数据的编码类型如OV_UINT8、OV_UINT16等；
int format：传入数据的格式如OV_IMG_FMT_RGBA、OV_IMG_FMT_NV21、OV_IMG_FMT_NV12、OV_IMG_FMT_I420；
int rotate：设备旋转的角度（可参考Demo查看获取方式）；

返回值：

int类型，返回美颜处理之后绑定的纹理id。

具体代码示例如下：

```
id = mOVFilter.nativeFilterProcessTexture(id, size.getWidth(), size.getHeight(), OVTextureType.OV_TEXTURE_2D.textureType, OVDataType.OV_UINT8.intType, OVImageFormat.OV_IMG_FMT_RGBA.formatType, 0);
```

2.2.7 算法销毁

接口描述：

在不需要用到算法的地方进行算法销毁。

接口示例：

```
mOVFilter.nativeFilterDestroy();
```

参数说明：

无

返回值：

int类型，返回0为删除算法相关成功，其它返回为删除算法相关处理失败。

具体代码示例如下：

```
int status = mOVFilter.nativeFilterDestroy();
```

三、支持的系统和硬件版本

1、硬件要求：要求设备上有相机模块,陀螺仪模块

2、CPU架构：armeabi-v7a、arm64-v8a

3、系统：最低支持 Android 4.0（API Level 14）需要开发者通过minSdkVersion来保证支持系统的检测

四、注意事项

1、 viapi-android-sdk的 minSdkVersion为 14。

2、demo工程Android Studio 3.4 及以上，Open GLES 2.0 及以上。

3、证书分为临时证书以_tmp结尾和正式证书。临时证书一般作为调试用，正式证书一般作为发布用。根据证书的不同代码需要做相应调整，全局搜索mIsTmpLicense字段，如果是_tmp证书，需把该字段改为true，正式证书的话把该字段改为false。

附 离线鉴权错误码定义

-2011 license没有初始化直接调用API接口。

-2012 当前的license与调用app不是绑定关系，license用在其他app中使用。

-2013 license无效。

-2014 license授权时间过期。

-2015 此license中不包含调用的算法能力（未购买此能力。

-2016 bundle id获取失败。

-2017 临时license时间校验失败。