快手开放平台 Android 接入文档（V2.3.0）

一、设计目标

开放平台 sdk 的设计目标是：第三方应用使用开放平台 sdk 能够方便、快速的调起快手主 app 的功能页面，从而将第三方内容发布到快手平台上或者使用快手 APP 提供的其他开放功能。

账号授权	快手主站授权	快手极速版授权					
社交功能	分享私信	分享私信到指定人	打开指定用户主页				
生产功能	单图编辑	单图发布	单视频编辑	单视频裁剪	单视频发布	多视频图片裁剪	智能裁剪视频

二、项目描述

外网访问 git 库：https://github.com/KwaiSocial/KwaiSDK-Demo-Android

三、第三方接入说明

1、接入 aar

版本要求　minsdkversion:19
快手外网引用 aar：
外网版本仅提供带 auth 认证的 aar

```
dependencies {
    implementation "com.github.kwaisocial:kwai-opensdk-withauth:2.3.0"   // 版本号建议设置成最新的版本
}
```

混淆配置

```
-keep class com.kwai.opensdk.sdk.** {*;}
-keep class com.kwai.opensdk.auth.** {*;}
```

2、api 使用说明

（1）应用配置

接入方应用需要在 build.gradle 中配置如下信息

```
android {
    defaultConfig {
        applicationId "com.kwai.chat.demo" // 接入方的包名
        manifestPlaceholders = [
            "KWAI_APP_ID": "ks703687443040312600" // 申请的 appId
        ]
    }
}
```

（2） 登录认证并获取用户的 openId（需要使用 kwai-opensdk-withauth 依赖或者 kwai-opensdk+kwai-auth）

接入方 app 需要通过渠道获取分配给接入方应用的 appId，当使用需要 openId 的功能时，请先获取必须的参数 openId，当前只有分享私信到人的功能需要必须参数 openId

预先初始化

```
public class MyApplication extends Application {

  @Override
  public void onCreate() {
    super.onCreate();
    KwaiOpenSdkAuth.init(this);
  }
}
```

授权登录获取 openId

```
    private IKwaiOpenSdkAuth mKwaiOpenSdkAuth =  new KwaiOpenSdkAuth(); // 初始化

  // 设置授权结果监听
  IKwaiAuthListener kwaiAuthListener = new IKwaiAuthListener() {
      @Override
      public void onSuccess(InternalResponse response) {
        new Thread(new Runnable() {
          public void run() {
            String result = null;
            int retry = 0;
            while (null == result && retry < NETWORK_MAX_RETRY_TIMES) {
              result = getOpenIdByNetwork(response.getCode());
              retry++;
              LogUtil.i(TAG, "retry=" + retry);
            }
            final String openId = result;
            Handler mainHandler = new Handler(Looper.getMainLooper());
            mainHandler.post(new Runnable() {
              @Override
              public void run() {
                mOpenId = openId;
                if (TextUtils.isEmpty(mOpenId)) {
                  mOpenIdTv.setText("当前 openId:" + "get openId error");
                } else {
                  mOpenIdTv.setText("当前 openId:" + mOpenId);
                }
              }
            });
          }
        }).start();
      }

      @Override
      public void onFailed(String state, int errCode, String errMsg) {
        mOpenIdTv.setText("code error is " + errCode + " and msg is " + errMsg);
      }

      @Override
      public void onCancel() {
        mOpenIdTv.setText("login is canceled");
      }
    };

// 请求授权，支持两个平台 KwaiConstants.Platform.KWAI_APP（快手主站）、KwaiConstants.Platform.NEBULA_APP（快手极速版）
// 未设置的默认通过快手主站授权
mKwaiOpenSdkAuth.sendAuthReqToKwai(getActivity(), Config.SCOPE, Config.STATE, kwaiAuthListener, new String[]{KwaiConstants.Platform.KWAI_APP});

// 服务器使用接口，获取 openId 的网络请求，为了安全性建议放在第三方客户端的服务器中，由接入方服务器实现这个请求接口后将 openid 返回接入方方客户端
  private String getOpenIdByNetwork(final String code) {
    String url = getRequestOpenIdUrl("code", APP_ID, APP_SECRET, code);
    String result = NetworkUtil.get(url, null, null);
    String openId = null;
    try {
```

```
      LogUtil.i(TAG, "result=" + result);
      JSONObject obj = new JSONObject(result);
      openId = obj.getString("open_id");
      LogUtil.i(TAG, "openId=" + openId);
    } catch (Throwable t) {
      LogUtil.e(TAG, "getOpenId exception");
    }
    return openId;
  }

  private String getRequestOpenIdUrl(String grantType, String appId, String appKey, String code) {
    StringBuilder builder = new StringBuilder();
    builder.append(URL_HOST);
    builder.append("/oauth2/access_token?");
    builder.append("grant_type=" + grantType);
    builder.append("&app_id=" + appId);
    builder.append("&app_secret=" + appKey);
    builder.append("&code=" + code);
    return builder.toString();
  }
```

（3）开放平台 api 使用

开放 api 使用方式

```
private IKwaiOpenAPI mKwaiOpenAPI; // 声明使用接口
mKwaiOpenAPI = KwaiOpenAPIFactory.createKwaiOpenAPI(this, SOCIAL_SHARE_FT.getAppId()); // 初始化

// 使用 sdk 提供的 loading 界面，设置 false 第三方应用可以自定义实现 loading
mKwaiOpenAPI.setShowDefaultLoading(true);

// 业务请求回调结果监听
mKwaiOpenAPI.addKwaiAPIEventListerer(new IKwaiAPIEventListener() {

  @Override
  public void onRespResult(@NonNull BaseResp resp) {
    Log.i(TAG, "resp=" + resp);
    if (resp != null) {
      Log.i(TAG, "errorCode=" + resp.errorCode + ", errorMsg="
          + resp.errorMsg + ", cmd=" + resp.getCommand()
          + ", transaction=" + resp.transaction);
      mCallbackTv.setText("CallBackResult: errorCode=" + resp.errorCode + ", errorMsg="
          + resp.errorMsg + ", cmd=" + resp.getCommand()
          + ", transaction=" + resp.transaction);
    } else {
      mCallbackTv.setText("CallBackResult: resp is null");
    }
  }
});

// 移除对回调结果的监听，请及时移除不用的监听避免内存泄漏问题
mKwaiOpenAPI.removeKwaiAPIEventListerer();
```

(4) 社交方向相关的业务请求示例代码：

选择人或者群组分享私信

```
// 通过选择人或者群组分享私信
  public void shareMessage() {
    // base params
    ShareMessage.Req req = new ShareMessage.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "sharemessage";

    // business params
    req.message = new KwaiMediaMessage();
    req.message.mediaObject = new KwaiWebpageObject();
    ((KwaiWebpageObject) req.message.mediaObject).webpageUrl =
        "https://blog.csdn.net/a249900679/article/details/51386660";
    req.message.title = "test";
    req.message.description = "webpage test share, hahahah";
    Bitmap b = BitmapFactory.decodeResource(getResources(), R.mipmap.ic_launcher);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
```

```
    b.compress(Bitmap.CompressFormat.PNG, 100, baos);
    req.message.thumbData = baos.toByteArray();

    // send request
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

分享私信给个人

```
// 通过 TargetOpenId 分享私信给个人，openId 是必须参数
  public void shareMessageToBuddy() {
    if (TextUtils.isEmpty(HistoryOpenIdActivity.sTargetOpenId)) {
      Toast.makeText(getActivity(), "sTargetOpenId is null, 请先设置", Toast.LENGTH_SHORT).show();
      return;
    }

    ShareMessageToBuddy.Req req = new ShareMessageToBuddy.Req();
    req.openId = mOpenId;
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "sharemessageToBuddy";

    req.targetOpenId = HistoryOpenIdActivity.sTargetOpenId;
    req.message = new KwaiMediaMessage();
    req.message.mediaObject = new KwaiWebpageObject();
    ((KwaiWebpageObject) req.message.mediaObject).webpageUrl =
        "https://blog.csdn.net/a249900679/article/details/51386660";
    req.message.title = "test";
    req.message.description = "webpage test share, hahahah";
    Bitmap b = BitmapFactory.decodeResource(getResources(), R.mipmap.ic_launcher);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    b.compress(Bitmap.CompressFormat.PNG, 100, baos);
    req.message.thumbData = baos.toByteArray();
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

打开个人主页

```
// 打开 TargetOpenId 指向的个人主页
  public void showProfile() {
    if (TextUtils.isEmpty(HistoryOpenIdActivity.sTargetOpenId) && getActivity() != null && !getActivity().isFinishing()) {
      Toast.makeText(getActivity(), "sTargetOpenId is null, 请先设置", Toast.LENGTH_SHORT).show();
      return;
    }

    ShowProfile.Req req = new ShowProfile.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "showProfile_1";

    req.targetOpenId = HistoryOpenIdActivity.sTargetOpenId;

    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

(5) 生产方向相关的业务请求示例代码：

- 生产方向的操作具有权限控制，请在 demo 中查询接入方是否获取了对应操作的权限
- 设置封面时需要封面图与视频大小保持一致

单图发布

```
//发布图片
  public void publishPicture(File file) {
    SinglePicturePublish.Req req = new SinglePicturePublish.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "SinglePicturePublish";

    req.mediaInfo = new PostShareMediaInfo();
    ArrayList<String> imageFile = new ArrayList<>();
    imageFile.add(file.getAbsolutePath());
    req.mediaInfo.mMultiMediaAssets = imageFile;
    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
```

```
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

单图发布

```
//编辑图片
  public void editPicture(File file) {
    SinglePictureEdit.Req req = new SinglePictureEdit.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "SinglePictureEdit";

    req.mediaInfo = new PostShareMediaInfo();
    ArrayList<String> imageFile = new ArrayList<>();
    imageFile.add(file.getAbsolutePath());
    req.mediaInfo.mMultiMediaAssets = imageFile;
    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

单视频发布

```
//发布单个视频
  public void publishSingleVideo(File file) {
    SingleVideoPublish.Req req = new SingleVideoPublish.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "SingleVideoPublish";

    req.mediaInfo = new PostShareMediaInfo();
    ArrayList<String> imageFile = new ArrayList<>();
    imageFile.add(file.getAbsolutePath());
    req.mediaInfo.mMultiMediaAssets = imageFile;
    if (!TextUtils.isEmpty(mSingleVideocoverPath.getText())) {
      req.mCover = mSingleVideocoverPath.getText().toString();
    }
    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

单视频编辑

```
//编辑单个视频
  public void editSingleVideo(File file) {
    SingleVideoEdit.Req req = new SingleVideoEdit.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "SingleVideoEdit";

    req.mediaInfo = new PostShareMediaInfo();
    ArrayList<String> imageFile = new ArrayList<>();
    imageFile.add(file.getAbsolutePath());
    req.mediaInfo.mMultiMediaAssets = imageFile;

    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

单图发布

```
//裁剪单个视频
  public void clipSingleVideo(File file) {
```

```
    SingleVideoClip.Req req = new SingleVideoClip.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "SingleVideoClip";

    req.mediaInfo = new PostShareMediaInfo();
    ArrayList<String> imageFile = new ArrayList<>();
    imageFile.add(file.getAbsolutePath());
    req.mediaInfo.mMultiMediaAssets = imageFile;

    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

多图和视频裁剪

```
// clipMultiMedia
  public void clipMultiMedia(ArrayList<String> multiMedia) {
    MultiMediaClip.Req req = new MultiMediaClip.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "MultiMediaClip";

    req.mediaInfo = new PostShareMediaInfo();
    req.mediaInfo.mMultiMediaAssets = multiMedia;

    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

智能剪辑 API

```
//智能剪辑的 api 使用
  public void aiCutMedias(ArrayList<String> multiMedia) {
    AICutMedias.Req req = new AICutMedias.Req();
    req.sessionId = mKwaiOpenAPI.getOpenAPISessionId();
    req.transaction = "AICutMedias";

    req.mediaInfo = new PostShareMediaInfo();
    req.mediaInfo.mMultiMediaAssets = multiMedia;

    if (!TextUtils.isEmpty(mTagList.getText().toString())) {
      req.mediaInfo.mTag = mTagList.getText().toString();
    }
    req.mediaInfo.mDisableFallback = mDisableFallBack.isChecked();
    if (!TextUtils.isEmpty(mExtraEdit.getText().toString())) {
      req.mediaInfo.mExtraInfo = mExtraEdit.getText().toString();
    }
    mKwaiOpenAPI.sendReq(req, getActivity());
  }
```

3、错误码定义

相关错误码定义

```
public interface KwaiOpenSdkErrorCode {

  //这些值不要随意改变，和 ios 统一了，和主站保持一致了
  int ERR_KWAI_APP_NOT_LOGIN = -1011;
  int ERR_INVALID_PARAMETERS = -1010;
  int ERR_KWAI_APP_UNSUPPORT = -1006;
```

```
  int ERR_NO_KWAI_APP = -1005;
  int ERR_CANCEL = -1;
  int ERR_OK = 1;
  // add new
  int ERR_NO_AUTH_AND_OPENID = -1012;
  int ERR_FALL_BACK_REJECT = -1013;
  int ERR_TEENAGER_MODE = -1014;
  int ERR_NETWORK = -1015;
  // post share
  int ERR_NO_PERMISSION = -1016;
  int ERR_COMPRESS_PICTURE = -1017;

  //server api 返回的errorcode
  int ERR_UNAUTHORIZED_CMD = 20088;  //未授权的cmd
  int ERR_UNACCESSIBLE_USERID = 20089; //无法获取openId对应的userId
  int ERR_INVALID_CMD = 20090; //无效的cmd
  int ERR_INCONSISTEN_OPENID_LOGIN_USERID  = 20091; //登陆userId 和 openId对应的userId不一致
  int ERR_TARGET_NOT_BUDDY  = 20092; //targetOpenId不是好友
  int ERR_INVALID_TARGET_OPEN_ID = 20094; // 无效的targetOpenId

  /**
   * 100200100, //请求缺少参数或参数类型错误
   * 100200101, //未授权的 client，无效的 app 或 developer
   * 100200102, //请求被拒绝，可能是无效的 token 等
   * 100200103, //请求的 responseType 错误
   * 100200104, //请求的 grantType 不支持
   * 100200105, //请求的 code 错误
   * 100200106, //请求的 scope 错误
   * 100200107, //无效的 openid
   * 100200108, // access_token 或者 refresh_token 过期
   * 100200109, // 用户取消该 app 授权
   * 100200110, // 用户授权过期
   * 100200111, // 用户未授权过
   */
  int ERR_SERVER_CHECK_INVALID_PARAMETER  = 100200100; //SERVER端检查发现无效的参数
}
```