快手广告开放平台 Android SDK接入文档

快手广告开放平台 Android SDK接入文档

- 1.接入准备
- 2. 接入SDK
  - 2.1 添加依赖
  - 2.2 添加权限
  - 2.3 混淆配置
- 3 SDK初始化
  - 3.1 初始化接口说明
  - 3.2 初始化配置参数说明
  - 3.3 个性化广告展示开关
  - 3.4 程序化广告展示开关
  - 3.5 隐私信息控制开关
- 4. 加载广告
  - 4.1 获取KsLoadManager对象
  - 4.2 构建场景KsScene
  - 4.3 请求激励视频广告
    - 4.3.1 请求示例
    - 4.3.2 KsRewardVideoAd接口说明
    - 4.3.3 激励视频的服务端回调支持
    - 4.3.4 激励视频内部广告监听
  - 4.4 请求全屏广告
    - 4.4.1 请求示例
    - 4.3.2 KsFullScreenVideoAd接口说明
  - 4.5 请求信息流广告
    - 4.5.1 请求示例
    - 4.5.1 请求示例
    - 4.5.2 KsFeedAd接口说明
  - 4.6 请求Draw竖屏信息流广告
    - 4.6.1 请求示例
    - 4.6.2 KsDrawAd接口说明
  - 4.7 请求原生广告数据
    - 4.7.1 请求示例
    - 4.7.2 KsNativeAd接口说明
    - 4.7.3 MaterialType素材类型
    - 4.7.4 InteractionType转化类型
    - 4.7.5 AdSourceLogoType广告角标类型
    - 4.7.6 DownloadListener广告下载监听
  - 4.8 请求开屏广告数据
    - 4.8.1 开屏小窗模式
    - 4.8.2 请求示例
      - 1.请求开屏数据
      - 2.添加开屏View
      - 3.接入开屏V+处理
    - 4.8.3 KsSplashScreenAd接口说明

4.9 请求插屏广告数据
4.9.1 请求示例
4.9.2 KsInterstitialAd接口说明
4.10 返回按钮
1 从快手返回
4.11 退出App安装提示
5. SDK错误码
6. 常见问题

1.接入准备

接入快手广告SDK前，请您联系快手广告平台申请您的AppId，广告位id等。

2. 接入SDK

SDK接入推荐使用aar的方式进行接入，请解压提供的广告SDK，在压缩包中找到ks_adsdk_xxx.aar。

接入过程中可参考压缩包中的**SDKDemo**进行使用，具体说明如下。

2.1 添加依赖

方式一：导入Android Studio(推荐)， 找到您的App工程下的libs文件夹，将上面提到的aar拷贝到该目录下，然后在项目根build.gradle文件中，以libs目录作为仓库地址添加本地仓库，然后在需要依赖的module的build.gradle文件中，添加SDK的dependencies依赖，代码如下：

```
allprojects {
  repositories {
    //本地文件仓库依赖
    flatDir { dirs 'libs'
  }
}
```

```
dependencies {
  // 快手SDK aar包，请将提供的aar包拷贝到libs目录下，添加依赖。根据接入版本修改SDK包名
  implementation files('libs/ks_adsdk_x.y.z.aar')
  def support_version = "28.0.0" //建议使用的26以上的support库版本，建议使用28最新的即可。
  // supprot库依赖，SDK内部依赖如下support，请确保添加
  implementation "com.android.support:appcompat-v7:$support_version"
  implementation "com.android.support:recyclerview-v7:$support_version"
}
```

ps：SDK也支持jar包的方式接入，如果需要采用jar包方式接入，请解压sdk的aar找到jar包，同时将所依赖的xml资源拷贝到您的工程中，即可使用。

AndroidX依赖

如果您的工程使用的是AndroidX的环境，请参考官网升级AndroidX，在`gradle.properties`文件中新增如下配置。

```
## Android 插件会使用对应的 AndroidX 库而非支持库。
android.useAndroidX=true
## Android 插件会通过重写现有第三方库的二进制文件，自动将这些库迁移为使用 AndroidX。
android.enableJetifier=true
```

```
dependencies {
  // 快手SDK aar包，请将提供的aar包拷贝到libs目录下，添加依赖。根据接入版本修改SDK包名
  implementation files('libs/ks_adsdk_x.y.z.aar')
  def version = ""
  // supprot库依赖，SDK内部依赖如下support，请确保添加
  implementation "androidx.appcompat:appcompat:$version"
  implementation "androidx.recyclerview:recyclerview:$version"
}
```

2.2 添加权限

接入此SDK，需要相关关系，请在您的App的AndroidManifest.xml文件中，添加如下权限：

```
<!--检测当前网络状态是2G、3G、4G还是WiFi-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!--联网权限-->
<uses-permission android:name="android.permission.INTERNET" />
<!--获取设备标识IMEI。用于标识用户-->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!--读写存储权限-->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!--获取MAC地址，用于标识用户-->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!--安装应用权限-->
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
<!--定位权限，不强制要求-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<!--建议添加“query_all_package”权限，SDK将通过此权限在AndroidR系统上判定广告对应的应用是否在用户的app上安装，避免投放错误的广告，以此提高用户的广告体验。若添加此权限，需要在您的用户隐私文档中声明-->
<uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
```

SDK要求最低系统版本为API 16 ，对于适配了Android6.0以上(API >= 23)的App，建议开发者在获得了动态权限之后，调用SDK的初始化代码，否则SDK可能受影响。

特别说明：请求广告数据前，请务必申请IMEI权限，否则会造成下发下载类广告的数据无法正常下发，影响广告计费追踪。所以强烈建议媒体在通过SDK请求广告前，先申请获取IMEI权限。相关权限说明，见下表：

权限名称	用途	缺失导致问题	应用场景	是否必须
ACCESS_NETWORK_STATE	访问网络状态：检测当前网络状态是2G、3G、4G还是WiFi	无法获取网络状况。严重影响视频广告填充率和播放效果	广告主投放广告时使用网络条件精准定向；后端广告返回/播放策略根据网络条件定向优化	是
INTERNET	网络访问权限	无法访问网络。拉取不到广告	广告请求时联网	是
READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE	磁盘读写权限	无法访问磁盘空间。下载类广告无法安装。严重影响广告价格	广告主根据安装情况进行效果评估	是
REQUEST_INSTALL_PACKAGES	应用安装权限	下载类广告无法安装。严重影响广告价格	广告主根据安装情况进行效果评估	是
ACCESS_WIFI_STATE	设备信息：MAC获取权限	无法获取MAC地址。严重影响广告填充率和广告效果	广告主投放广告时精准定向用户画像	否
READ_PHONE_STATE	设备信息：IMEI获取权限	无法获取设备标识IMEI。严重影响广告填充率和广告效果	广告主投放广告时精准定向用户画像	是
ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION	定位权限	无定位信息，影响广告填充率和定向推荐效果	广告主投放广告时精准地理位置定向	否

2.3 混淆配置

请确保您的应用打包混淆时，请在混淆配置文件添加如下配置：

```
-keep class org.chromium.** {*;}
-keep class org.chromium.** { *; }
-keep class aegon.chrome.** { *; }
-keep class com.kwai.**{ *; }

-dontwarn com.kwai.**
-dontwarn com.kwad.**
-dontwarn com.ksad.**
-dontwarn aegon.chrome.**
```

如果您的应用启用了资源混淆或资源缩减，您需要保留SDK的资源，SDK的资源名都是以ksad_开头的。您可以在资源混淆配置文件添加如下配置：

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
    tools:keep="@layout/ksad_*,@id/ksad_*,@style/ksad_*,
  @drawable/ksad_*,@string/ksad_*,@color/ksad_*,@attr/ksad_*,@dimen/ksad_*"
/>
```

3 SDK初始化

请在您应用的`Application`的`onCreate()`方法中调用以下代码来初始化快手广告sdk。

```
public class DemoApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        initKSSDK(this);
    }

    public static void initKSSDK(Context appContext) {
        KsAdSDK.init(appContext, new SdkConfig.Builder()
                .appId("90009") // 测试aapId，请联系快手平台申请正式AppId，必填
                .appName("test-android-sdk")  // 测试appName，请填写您应用的名称，
非必填
                .showNotification(true) // 是否展示下载通知栏
                .debug(true) // 是否开启sdk 调试日志  可选
                .build());
    }
}
```

3.1 初始化接口说明

```
/**
 * 快手广告sdk初始化入口
 *
 * @param context Application context,  必填
 * @param config 初始化配置，必填
 * @return true: 初始化成功  false : 初始化失败
 */
 public static synchronized boolean init(Context context, SdkConfig config)
```

3.2 初始化配置参数说明

```
public static class SdkConfig.Builder {
  private boolean enableDebug; // 可选参数，是否开启debug日志，默认false
  @Nullable
  private String appId; // 必填参数，应用申请的AppId
  @Nullable
  private String appName; // 可选参数，您的应用名称
  private boolean showNotification = true; // 可选参数，是否展示下载通知栏，默认为
true
}
```

3.3 个性化广告展示开关

个性化推荐广告开关：关闭后，看到的广告数量不变，相关度将降低。 是否允许开启广告的个性化推荐（false-关闭，true-开启），由开发者通过SDK以下接口来设置。不设置的话则默认为true。

```
  /**
   * 设置是否启用个性化推荐
   *
   * @param enablePersonalRecommend true为启用，false 不启用
   */
   @KsAdSdkDynamicApi
   @Keep
   void setPersonalRecommend(boolean enablePersonalRecommend);
```

3.4 程序化广告展示开关

程序化推荐广告开关：关闭后，看到的广告数量不变，但将不会为你推荐程序化广告。 是否允许开启广告的程序化推荐（false-关闭，true-开启），由开发者通过SDK以下接口来设置。不设置的话则默认为true。

```
/**
 * 设置是否启用程序化广告推荐
 *
 * @param enableProgrammaticRecommend true为启用，false 不启用
 */
@KsAdSdkDynamicApi
@Keep
void setProgrammaticRecommend(boolean enableProgrammaticRecommend);
```

3.5 隐私信息控制开关

注意**隐私数据传入时，请传原始值，不需要加密。**

```
public abstract class KsCustomController {

  /**
   * 是否允许SDK主动使用地理位置信息
   *
   * @return true可以获取，false禁止获取。默认为true
   */
  @KsAdSdkApi
  @Keep
  public boolean canReadLocation() {
    return true;
  }

  /**
   * 当canReadLocation=false时，可传入地理位置信息，sdk使用您传入的地理位置信息
   *
   * @return 地理位置参数
   */
  @KsAdSdkApi
  @Keep
  public Location getLocation() {
    return null;
  }

  /**
   * 是否允许SDK主动使用手机硬件参数，如：imei，android_id，meid，imsi，iccid
   *
   * @return true可以使用，false禁止使用。默认为true
   */
  @KsAdSdkApi
  @Keep
  public boolean canUsePhoneState() {
    return true;
  }
```

```
    /**
     * 当canUsePhoneState=false时，可传入原始的imei信息，sdk使用您传入的原始imei信息。
     * 注意：请传入原始imei值，无需使用md5加密，sdk整体信息加密传输已满足合规需要
     */
    @KsAdSdkApi
    @Keep
    public String getImei() {
      return "";
    }

    /**
     * 当canUsePhoneState=false时，可传入原始的imei信息，sdk使用您传入的原始imei信息。传入的是数组，imei个数和手机卡个数相同
     * 注意：请传入原始imei值，无需使用md5加密，sdk整体信息加密传输已满足合规需要
     */
    @KsAdSdkApi
    @Keep
    public String[] getImeis() {
      return null;
    }

    /**
     * 当canUsePhoneState=false时，可传入android_id信息，sdk使用您传入的android_id
     */
    @KsAdSdkApi
    @Keep
    public String getAndroidId() {
      return "";
    }

    /**
     * 是否允许SDK主动使用oaid
     *
     * @return true可以使用，false禁止使用。默认为true
     */
    @KsAdSdkApi
    @Keep
    public boolean canUseOaid() {
      return true;
    }

    /**
     * 当canUseOaid=false时，可传入oaid信息，sdk使用您传入的oaid信息
     * 注意：请传入原始oaid，无需使用md5加密，sdk整体信息加密传输已满足合规需要
     */
    @KsAdSdkApi
    @Keep
    public String getOaid() {
      return "";
```

```
    }


    /**
     * 是否允许SDK主动使用mac_address
     *
     * @return true可以使用，false禁止使用。默认为true
     */
    @KsAdSdkApi
    @Keep
    public boolean canUseMacAddress() {
      return true;
    }

    /**
     * 当canUseMacAddress=false时，可传入mac地址信息，sdk使用您传入的mac地址信息
     */
    @KsAdSdkApi
    @Keep
    public String getMacAddress() {
      return "";
    }


    /**
     * 是否允许SDK主动使用ACCESS_NETWORK_STATE权限
     *
     * @return true可以使用，false禁止使用。默认为true
     */
    @KsAdSdkApi
    @Keep
    public boolean canUseNetworkState() {
      return true;
    }

    /**
     * 是否允许SDK主动使用存储权限
     *
     * @return true可以使用，false禁止使用。默认为true
     */
    @KsAdSdkApi
    @Keep
    public boolean canUseStoragePermission() {
      return true;
    }

    /**
     * 是否允许SDK主动读取app安装列表
     *
```

```
     * @return true可以使用，false禁止使用。默认为true
     */
    @KsAdSdkApi
    @Keep
    public boolean canReadInstalledPackages() {
      return true;
    }

    /**
     * 当canReadInstalledPackages=false时，可传入package list信息，sdk使用您传入的
package list
     */
    @KsAdSdkApi
    @Keep
    public List<String> getInstalledPackages() {
      return null;
    }
  }
```

4. 加载广告

4.1 获取KsLoadManager对象

`KsLoadManager`对象是整个快手SDK的广告请求的接口，用于获取激励视频广告、全屏视频广告、信息流视频广告等。获取方式：

```
KsLoadManager adRequestManager = KsAdSDK.getLoadManager();
```

KsLoadManager接口说明：

```
public interface KsLoadManager {

    /**
     * 异步请求全屏视频广告，结果通过{@link FullScreenVideoAdListener}回调
     *
     * @param scene    广告场景
     * @param listener 结果回调接口
     */
    void loadFullScreenVideoAd(KsScene scene, @NonNull
FullScreenVideoAdListener listener);

    /**
     * 异步请求激励视频广告，结果通过{@link RewardVideoAdListener}回调
     *
     * @param scene    广告场景
     * @param listener 结果回调接口
     */
```

```
    @MainThread
    void loadRewardVideoAd(KsScene scene, @NonNull RewardVideoAdListener
listener);


    /**
     * 异步请求Feed广告，结果通过{@link FeedAdListener}回调
     *
     * @param sceneData 请求配置信息
     * @param listener  加载结果回调
     */
    @MainThread
    void loadFeedAd(KsScene sceneData, @NonNull FeedAdListener listener);

    /**
     * 异步请求Draw广告，结果通过{@link DrawAdListener}回调
     *
     * @param sceneData 请求配置信息
     * @param listener  加载结果回调
     */
    @MainThread
    void loadDrawAd(KsScene sceneData, @NonNull DrawAdListener listener);


    /**
     * 异步请求原生广告，结果通过{@link NativeAdListener}回调
     *
     * @param sceneData 请求配置信息
     * @param listener  加载结果回调
     */
    @MainThread
    void loadNativeAd(KsScene sceneData, @NonNull NativeAdListener listener);

  /**
   * 异步请求开屏广告，结果通过{@link SplashScreenAdListener}回调
   *
   * @param scene 广告场景
   * @param listener 结果回调接口
   */
  @MainThread
  void loadSplashScreenAd(@NonNull KsScene scene, @NonNull
SplashScreenAdListener listener);

    /**
     * 全屏视频广告加载监听
     */
    interface FullScreenVideoAdListener {

        /**
```

```
         * 加载失败的回调
         *
         * @param code 错误状态码
         * @param msg  错误文案
         */
        @Keep
        void onError(int code, String msg);

        /**
         * 请求完毕，收到服务器的返回，
         * 因为有预加载的逻辑，服务器广告结果返回后，物料可能仍在下载中，
         * @param adNumber 表示填充广告的数目
         * */
        void onRequestResult(int adNumber);

        /**
         * 广告加载成功回调
         *
         * @param adList 加载的视频广告
         */
        @MainThread
        void onFullScreenVideoAdLoad(@Nullable List<KsFullScreenVideoAd>
adList);
    }

    /**
     * 激励视频广告加载监听
     */

    interface RewardVideoAdListener {

        /**
         * 加载失败的回调
         *
         * @param code 错误状态码
         * @param msg  错误文案
         */
        @MainThread
        void onError(int code, String msg);

        /**
         * 请求完毕，收到服务器的返回，
         * 因为有预加载的逻辑，服务器广告结果返回后，物料可能仍在下载中，
         * @param adNumber 表示填充广告的数目
         * */
        void onRequestResult(int adNumber);

        /**
         * 广告加载成功回调
```

```
         *
         * @param adList 加载的视频广告
         */
        @MainThread
        void onRewardVideoAdLoad(@Nullable List<KsRewardVideoAd> adList);
    }

    /**
     * 信息流广告加载监听
     */
    @KsAdSdkApi
    @Keep
    interface FeedAdListener {

        /**
         * 加载失败的回调
         *
         * @param code 错误状态码
         * @param msg  错误文案
         */
        @MainThread
        @KsAdSdkApi
        @Keep
        void onError(int code, String msg);

        /**
         * 广告加载成功的回调
         *
         * @param adList 返回的广告列表
         */
        @MainThread
        @KsAdSdkApi
        @Keep
        void onFeedAdLoad(@Nullable List<KsFeedAd> adList);
    }

    /**
     * Draw信息流广告加载监听
     */
    @KsAdSdkApi
    @Keep
    interface DrawAdListener {

        /**
         * 加载失败的回调
         *
         * @param code 错误状态码
         * @param msg  错误文案
         */
```

```
        @MainThread
        @KsAdSdkApi
        @Keep
        void onError(int code, String msg);

        /**
         * 广告加载成功的回调
         *
         * @param adList 返回的广告列表
         */
        @MainThread
        @KsAdSdkApi
        @Keep
        void onDrawAdLoad(@Nullable List<KsDrawAd> adList);
    }

    /**
     * 原生广告加载监听
     */
    interface NativeAdListener {

        /**
         * 加载失败的回调
         *
         * @param code 错误状态码
         * @param msg  错误文案
         */
        @MainThread
        void onError(int code, String msg);

        /**
         * 广告加载成功的回调
         *
         * @param adList 返回的广告列表
         */
        @MainThread
        @KsAdSdkApi
        @Keep
        void onNativeAdLoad(@Nullable List<KsNativeAd> adList);
    }

/**
 * 开屏视频广告加载监听
 */
@KsAdSdkApi
@Keep
interface SplashScreenAdListener {

    /**
```

```
     * 加载失败的回调
     *
     * @param code 错误状态码
     * @param msg 错误文案
     */
    @MainThread
    @KsAdSdkApi
    @Keep
    void onError(int code, String msg);

    /**
     * 请求完毕，收到服务器的返回，
     * 因为有预加载的逻辑，服务器广告结果返回后，物料可能仍在下载中，
     * @param adNumber 表示填充广告的数目
     * */
    void onRequestResult(int adNumber);

    /**
     * 广告加载成功回调
     *
     * @param splashScreenAd 加载的开屏广告
     */
    @MainThread
    void onSplashScreenAdLoad(@Nullable KsSplashScreenAd splashScreenAd);
  }
}
```

4.2 构建场景KsScene

`KsScene`对象用于标识场景，不同的场景使用不同的posId，posId请联系快手平台进行申请。

```
 KsScene scene = new KsScene.Builder(posId)// posId 为平台申请广告位id  必填
      .setBackUrl("ksad://returnback") // 返回链接backUrl的设置见 **4.10** 小节 非必填
      .build();
```

4.3 请求激励视频广告

使用前请申请激励视频广告对应posId，接入方可调用`KsAdSDK.getLoadManager().loadRewardVideoAd(KsScene scene, RewardVideoAdListener listener)`异步请求激励视频广告，`KsScene`是请求场景，`RewardVideoAdListener`是结果回调接口。

4.3.1 请求示例

详细示例参考附件demo的`TestRewardVideoActivity`

```
// 1.请求激励视频广告，获取广告对象，KsRewardVideoAd
```

```
    public void requestRewardAd(View view) {
        mRewardVideoAd = null;
        KsScene scene = new
KsScene.Builder(TestPosId.POSID_REWARD.posId).build(); // 此为测试posId，请联系快
手平台申请正式posId
        KsAdSDK.getLoadManager().loadRewardVideoAd(scene, new
KsLoadManager.RewardVideoAdListener() {
            @Override
            public void onError(int code, String msg) {
                ToastUtil.showToast(mContext, "激励视频广告请求失败" + code +
msg);
            }

            @Override
            public void onRewardVideoResult(@Nullable List<KsRewardVideoAd>
adList) {
              ToastUtil.showToast(mContext, "激励视频广告数据请求成功");
            }

            @Override
            public void onRewardVideoAdLoad(@Nullable List<KsRewardVideoAd>
adList) {
                if (adList != null && adList.size() > 0) {
                    mRewardVideoAd = adList.get(0);
                    ToastUtil.showToast(mContext, "激励视频广告数据请求且资源缓存成
功");
                }
            }
        });
    }

// 竖屏播放（默认）
public void showPortrait(View view) {
  showRewardVideoAd(null);
}

// 横屏播放
public void showLandscape(View view) {
  KsVideoPlayConfig videoPlayConfig = new KsVideoPlayConfig.Builder()
          .showLandscape(true) // 横屏播放
          .build();
  showRewardVideoAd(videoPlayConfig);
}

// 2.展示激励视频广告，通过步骤1获取的KsRewardVideoAd对象，判断缓存有效，则设置监听并展示
private void showRewardVideoAd(VideoPlayConfig videoPlayConfig) {
  if (mRewardVideoAd != null && mRewardVideoAd.isAdEnable()) {
    mRewardVideoAd
```

```
        .setRewardAdInteractionListener(new
KsRewardVideoAd.RewardAdInteractionListener() {
          @Override
          public void onAdClicked() {
            ToastUtil.showToast(mContext, "激励视频广告点击");
          }

          @Override
          public void onPageDismiss() {
            ToastUtil.showToast(mContext, "激励视频广告关闭");
          }

          @Override
          public void onVideoPlayError(int code, int extra) {
            ToastUtil.showToast(mContext, "激励视频广告播放出错");
          }

          @Override
          public void onVideoPlayEnd() {
            ToastUtil.showToast(mContext, "激励视频广告播放完成");
          }

          @Override
          public void onVideoPlayStart() {
            ToastUtil.showToast(mContext, "激励视频广告播放开始");
          }

          @Override
          public void onRewardVerify() {
            ToastUtil.showToast(mContext, "激励视频广告获取激励");
          }

          @Override
          public void onRewardStepVerify(int taskType, int currentTaskStatus)
{
            ToastUtil.showToast(mContext, "激励视频广告分阶段获取激励");
          }
        });

    // 设置 "再看一个" 的回调接口，和激励视频的普通接口不能是同一个对象
    mRewardVideoAd
        .setRewardPlayAgainInteractionListener(new
KsRewardVideoAd.RewardAdInteractionListener() {
          @Override
          public void onAdClicked() {
            ToastUtil.showToast(mContext, "再看一个-激励视频广告点击");
          }

          @Override
```

```
            public void onPageDismiss() {
              ToastUtil.showToast(mContext, "再看一个-激励视频广告关闭");
            }

            @Override
            public void onVideoPlayError(int code, int extra) {
              ToastUtil.showToast(mContext, "再看一个-激励视频广告播放出错");
            }

            @Override
            public void onVideoPlayEnd() {
              ToastUtil.showToast(mContext, "再看一个-激励视频广告播放完成");
            }

            @Override
            public void onVideoPlayStart() {
              ToastUtil.showToast(mContext, "再看一个-激励视频广告播放开始");
            }

            @Override
            public void onRewardVerify() {
              ToastUtil.showToast(mContext, "再看一个-激励视频广告获取激励");
            }

            @Override
            public void onRewardStepVerify(int taskType, int currentTaskStatus)
{
              ToastUtil.showToast(mContext, "再看一个-激励视频广告分阶段获取激励");
            }
          });

      mRewardVideoAd.showRewardVideoAd(this, videoPlayConfig);
    } else {
      ToastUtil.showToast(mContext, "暂无可用激励视频广告，请等待缓存加载或者重新刷新");
    }
}
```

4.3.2 KsRewardVideoAd接口说明

`KsRewardVideoAd`是激励视频广告请求的回调结果接口，用于设置广告交互监听，显示视频。

```
public interface KsRewardVideoAd {
  /**
   * 注册激励视频广告回调
   *
   * @param listener 交互监听器
   */
  void setRewardAdInteractionListener(RewardAdInteractionListener listener);
```

```
    /**
     * 展示激励视频广告
     *
     * @param activity 宿主activity,用于判断宿主是否finishing
     * @param videoPlayConfig 视频展示配置
     */
    void showRewardVideoAd(Activity activity, VideoPlayConfig videoPlayConfig);

    /**
     * 判断该视频是否可用（主要是素材是否已缓存成功）
     *
     * @return true表示视频可用，false表示不可用
     */
    boolean isAdEnable();

    /**
     * 获取ecpm，单位：分，默认为0（使用该功能需同步商务申请使用权限）
     */
    int getECPM();

    /**
     * 媒体回传二价ecpm，竞价成功后，必须在展示前回传（使用该功能需同步商务申请使用权限）
     *
     * @param bidEcpm 单位：分
     */
    void setBidEcpm(int bidEcpm);

    /**
     * 广告曝光失败后上报失败原因
     * @param adExposureFailureCode 曝光失败类型
     *        当失败类型为AdExposureFailureCode.BID_FAILED时，
     *        曝光失败原因请上报胜出的AdExposureFailedReason.winEcpm值
     * @param adExposureFailedReason 曝光失败原因描述
     */
    void reportAdExposureFailed(@AdExposureFailureCode int
adExposureFailureCode,
            AdExposureFailedReason adExposureFailedReason);

    /**
     * 广告素材类型:视频，单图，组图
     */
    @MaterialType
    int getMaterialType();

    /**
     * 广告操作类型:h5,下载
     */
    @InteractionType
```

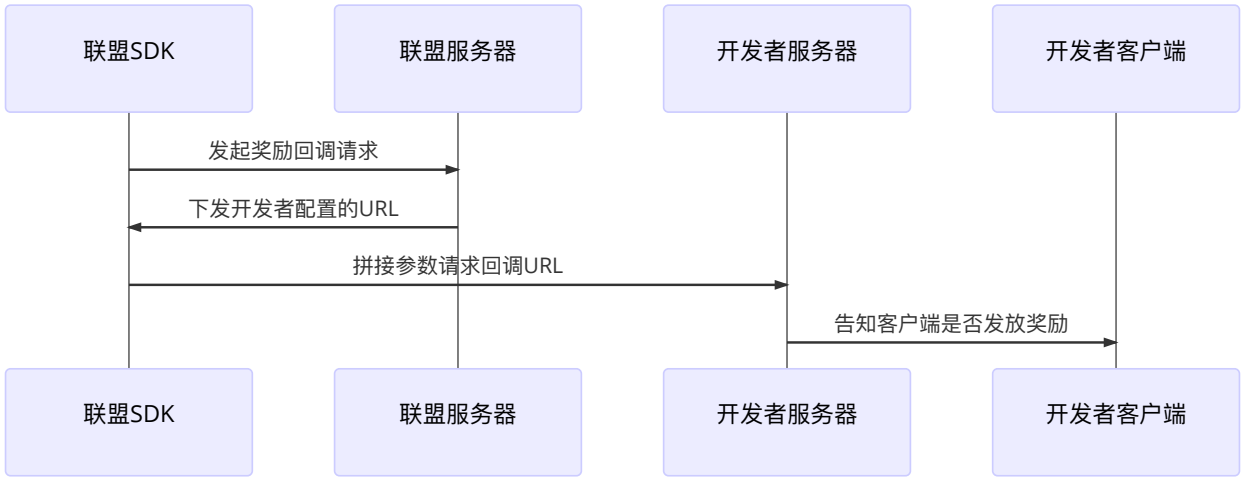
```
  int getInteractionType();

  /**
   * 激励视屏广告交互监听器
   */
  interface RewardAdInteractionListener {

    /**
     * 广告点击回调
     */
    void onAdClicked();

    /**
     * 视频页面关闭
     */
    void onPageDismiss();

    /**
     * 视频播放出错
     */
    void onVideoPlayError(int code, int extra);

    /**
     * 视频播放完成
     */
    void onVideoPlayEnd();

    /**
     * 视频播放开始
     */
    void onVideoPlayStart();

    /**
     * 视频激励有效性回调
     */
    void onRewardVerify();

    /**
     *  视频激励分阶段回调（激励广告新玩法，相关政策请联系商务或技术支持）
     * @param taskType 当前激励视频所属任务类型
     *                 RewardTaskType.LOOK_VIDEO 观看视频类型              属于浅度奖励类型
     *                 RewardTaskType.LOOK_LANDING_PAGE 浏览落地页N秒类型  属于深度奖励类型
     *                 RewardTaskType.USE_APP 下载使用App N秒类型          属于深度奖励类型
     * @param currentTaskStatus  当前所完成任务类型，@RewardTaskType中之一
     */
    void onRewardStepVerify(int taskType, int currentTaskStatus);
```

```
    /**
     * 额外奖励的回调，在触发激励视频的额外奖励的时候进行通知
     * AD_3.3.25 新增
     * @param extraRewardType 额外奖励的类型，定义在 KsExtraRewardType 中
     */
    void onExtraRewardVerify(@KsExtraRewardType int extraRewardType);
  }
}
```

4.3.3 激励视频的服务端回调支持

激励视频的有效性回调，支持开发者服务端回调，SDK会在激励视频有效的时候，GET请求回调 url（该url需要开发者在SSP平台中进行配置），这样开发者可以在自己的服务端对激励视频的有效性进行再次验证。

整体的调用流程如下：



在SSP平台配置回调url的时候，需要按照以下格式进行配置：

```
https://your_callback_url?
userId=__UID__&transId=__TRANSID__&sign=__SIGN__&amount=__RAMOUNT__&name=__RNA
ME__&extra=__EXTRA__
```

其中 “https://your_callback_url” 为客户的服务端回调url的地址，后面为各个参数的设置，客户可以按照需要进行配置。例如：

```
https://your_callback_url?userId=__UID__&transId=__TRANSID__
```

这样配置的话，则SDK在调用客户服务端url的时候，只会包含 `userId` 和 `transId` 这两个参数，而不包含其他的。

> 注意：在Android targetSdkVersion=28 的应用中，默认不再支持http请求，所以建议回调url都使用 https 协议。

在SSP配置完成后，开发者使用该功能的时候，需要在请求激励视频的时候，通过KsScene对象设置相关的参数，参数示例如下：

```
// 此为测试posId，请联系快手平台申请正式posId
KsScene.Builder builder = new KsScene.Builder(posId)
        .screenOrientation(screenOrientation);
// 激励视频服务端回调的参数设置
Map<String, String> rewardCallbackExtraData = new HashMap<>();
// 开发者系统中的用户id，会在请求客户的回调url中带上
rewardCallbackExtraData.put("thirdUserId", "your-uerid");
// 开发者自定义的附加参数，会在请求客户的回调url中带上
rewardCallbackExtraData.put("extraData", "your-extra-data");

builder.rewardCallbackExtraData(rewardCallbackExtraData);
```

联盟SDK在调用开发者指定的回调url的时候，会附加一些参数（形式为GET请求的参数，配置方法在上文中）供开发者服务端使用，参数示例如下：

```
http://your_callback_url?userId=your-uerid&transId=test_trans_id&sign=11121222&amount=0&name=name&extra=your-extra-data
```

GET请求的各个参数说明：

参数名称	参数类型	参数说明
extra	String	开发者在SDK中请求激励视频时设置的自定义附加参数“extraData”
userId	String	开发者在SDK中请求激励视频时设置的 “thirdUserId” 参数
transId	String	完成观看的唯一交易ID
sign	String	签名
name	String	奖励名称
amount	int	奖励数量

> 上表中提到的sign为唯一签名，计算方式为:

> sign = md5(appSecurityKey:transId) ，所有字母均为小写。
>
> 其中 appSecurityKey 为在SSP平台设置回调url时获得，transId为请求中的参数。

SDK在请求开发者的指定url的时候，开发者需要按照一定的格式返回给SDK结果，返回数据为json的格式，详情如下：

字段名称	字段定义	字段类型	备注
isValid	校验结果	bool	判定结果，是否发放奖励。

```
{
  "isValid" : true
}
```

4.3.4 激励视频内部广告监听

激励视频的内部，会在一些场景中请求并展示广告（目前有聚合页和回流页，相关功能的开通请联系技术支持人员），SDK支持设置对内部广告的监听

```
public interface KsRewardVideoAd {
    /**
     * 设置激励视频回流页场景中 聚合广告的监听接口
     *
     * @param innerAdInteractionListener 交互监听接口
     */
    @KsAdSdkApi
    @Keep
    void setInnerAdInteractionListener(KsInnerAd.KsInnerAdInteractionListener
            innerAdInteractionListener);

}
```

其中监听接口`KsInnerAd.KsInnerAdInteractionListener`的定义如下：

```
  /**
   * 聚合广告的交互监听接口，会在不同的广告场景中出现聚合广告
   */
  @KsAdSdkApi
  @Keep
  interface KsInnerAdInteractionListener {

    /**
     * 广告点击的回调
     */
```

```
    @KsAdSdkApi
    @Keep
    void onAdClicked(KsInnerAd ksInnerAd);

    /**
     * 广告展示回调 每个广告仅回调一次
     */
    @KsAdSdkApi
    @Keep
    void onAdShow(KsInnerAd ksInnerAd);
  }
```

示例代码如下(见Demo 工程的TestRewardVideoActivity 类):

```
    // 设置激励视频内部广告的监听
    mRewardVideoAd.setInnerAdInteractionListener(
       new KsInnerAd.KsInnerAdInteractionListener() {
           @Override
           public void onAdClicked(KsInnerAd ksInnerAd) {
           ToastUtil.showToast(mContext, "激励视频内部广告点击: " +
ksInnerAd.getType());
           }

           @Override
           public void onAdShow(KsInnerAd ksInnerAd) {
           ToastUtil.showToast(mContext, "激励视频内部广告曝光: " +
ksInnerAd.getType());
       }
    });
```

4.4 请求全屏广告

使用前请申请全屏广告对应posId，接入方可调用`KsAdSDK.getLoadManager().loadFullScreenVideoAd(KsScene scene, FullScreenVideoAdListener listener)`异步请求全屏广告，KsScene是请求场景，FullScreenVideoAdListener是结果回调接口。

4.4.1 请求示例

详细示例参考附件demo的`TestFullScreenVideoActivity`

```
// 1.请求全屏视频广告，获取广告对象，KsFullScreenVideoAd
public void requestFullScreenAd(View view) {
  mFullScreenVideoAd = null;
  KsScene scene = new
KsScene.Builder(TestPosId.POSID_FULLSCREEN.posId).build(); // 此为测试posId，请联
系快手平台申请正式posId
```

```
  KsAdSDK.getLoadManager().loadFullScreenVideoAd(scene,
      new KsLoadManager.FullScreenVideoAdListener() {
        @Override
        public void onError(int code, String msg) {
          ToastUtil.showToast(mContext, "全屏视频广告请求失败" + code + msg);
        }

        @Override
        public void onFullScreenVideoResult(@Nullable
List<KsFullScreenVideoAd> adList) {
           ToastUtil.showToast(mContext, "全屏视频广告数据请求成功");
        }

        @Override
        public void onFullScreenVideoAdLoad(@Nullable
List<KsFullScreenVideoAd> adList) {
          if (adList != null && adList.size() > 0) {
            mFullScreenVideoAd = adList.get(0);
            ToastUtil.showToast(mContext, "全屏视频广告数据请求且资源缓存成功");
          }
        }
      });
}

// 竖屏播放（默认）
public void showLandscape(View view) {
  showFullScreenVideoAd(null);
}

// 横屏播放
public void showPortrait(View view) {
  KsVideoPlayConfig videoPlayConfig = new KsVideoPlayConfig.Builder()
      .showLandscape(true) // 横屏播放
      .build();
  showFullScreenVideoAd(videoPlayConfig);
}

// 2.展示全屏视频广告，通过步骤1获取的KsFullScreenVideoAd对象，判断缓存有效，则设置监听并
展示
private void showFullScreenVideoAd(VideoPlayConfig videoPlayConfig) {
  if (mFullScreenVideoAd != null && mFullScreenVideoAd.isAdEnable()) {
    mFullScreenVideoAd
        .setFullScreenVideoAdInteractionListener(new
FullScreenVideoAdInteractionListener() {
          @Override
          public void onAdClicked() {
            ToastUtil.showToast(mContext, "全屏视频广告点击");
          }
```

```
            @Override
            public void onPageDismiss() {
              ToastUtil.showToast(mContext, "全屏视频广告关闭");
            }

            @Override
            public void onVideoPlayError(int code, int extra) {
              ToastUtil.showToast(mContext, "全屏视频广告播放出错");
            }

            @Override
            public void onVideoPlayEnd() {
              ToastUtil.showToast(mContext, "全屏视频广告播放完成");
            }

            @Override
            public void onVideoPlayStart() {
              ToastUtil.showToast(mContext, "全屏视频广告播放开始");
            }

            @Override
            public void onSkippedVideo() {
              ToastUtil.showToast(mContext, "全屏视频广告播放跳过");

            }
          });
      mFullScreenVideoAd.showFullScreenVideoAd(this, videoPlayConfig);
    } else {
      ToastUtil.showToast(mContext, "暂无可用全屏视频广告，请等待缓存加载或者重新刷新");
    }
}
```

4.3.2 KsFullScreenVideoAd接口说明

`KsFullScreenVideoAd`是全屏广告请求的回调结果接口，用于设置广告交互监听，显示视频。

```
public interface KsFullScreenVideoAd {
  /**
   * 注册全屏视频广告回调
   *
   * @param listener 交互监听器
   */
  void
setFullScreenVideoAdInteractionListener(FullScreenVideoAdInteractionListener
listener);

  /**
   * 展示全屏视频广告
   *
```

```
     * @param activity 宿主activity,用于判断宿主是否finishing
     * @param videoPlayConfig 视频展示配置
     */
    void showFullScreenVideoAd(Activity activity, VideoPlayConfig
videoPlayConfig);

    /**
     * 判断该视频是否可用（主要是素材是否已缓存成功）
     *
     * @return true表示视频可用，false表示不可用
     */
    boolean isAdEnable();

    /**
     * 获取ecpm，单位：分，默认为0（使用该功能需同步商务申请使用权限）
     */
    int getECPM();

    /**
     * 媒体回传二价ecpm，竞价成功后，必须在展示前回传（使用该功能需同步商务申请使用权限）
     *
     * @param bidEcpm 单位：分
     */
    void setBidEcpm(int bidEcpm);

    /**
     * 广告曝光失败后上报失败原因
     * @param adExposureFailureCode 曝光失败类型
     *        当失败类型为AdExposureFailureCode.BID_FAILED时，
     *        曝光失败原因请上报胜出的AdExposureFailedReason.winEcpm值
     * @param adExposureFailedReason 曝光失败原因描述
     */
    void reportAdExposureFailed(@AdExposureFailureCode int
adExposureFailureCode,
            AdExposureFailedReason adExposureFailedReason);

    /**
       * 广告素材类型
       */
      @MaterialType
      int getMaterialType();

      /**
       * 广告操作类型
       */
      @InteractionType
      int getInteractionType();

    /**
```

```
     * 全屏视频广告交互监听器
     */
    interface FullScreenVideoAdInteractionListener {

        /**
         * 全屏视频广告点击回调
         */
        void onAdClicked();

        /**
         * 全屏视频广告页面关闭
         */
        void onPageDismiss();

        /**
         * 全屏视频广告播放出错
         */
        void onVideoPlayError(int code, int extra);

        /**
         * 全屏视频广告播放完成
         */
        void onVideoPlayEnd();

        /**
         * 全屏视频广告播放开始
         */
        void onVideoPlayStart();

        /**
         * 跳过全屏视频广告播放
         */
        void onSkippedVideo();
    }
}
```

4.5 请求信息流广告

为降低媒体接入成本，快手广告SDK默认提供了几种视频和图片的信息流广告模板，媒体可直接通过请求信息流广告数据，获取对应模板控件直接填充，使用前请联系快手平台，申请对应posId，并配置相应的模板。如果SDK提供的模板不能满足媒体自身样式的要求，可以通过请求原生广告进行自渲染。

4.5.1 请求示例

4.5.1 请求示例

请求旧版信息流接口 loadFeedAd， 改接口已经废弃，推荐使用loadConfigFeedAd 接口。 详细示例参考附件demo中的`TestFeedListActivity`、`TestFeedRecyclerActivity`

```
//1.请求广告
private void requestAd(long posId) {
  KsScene scene = new KsScene.Builder(posId)
        .adNum(3).build(); // 此为测试posId，请联系快手平台申请正式posId
  KsAdSDK.getLoadManager().loadFeedAd(scene, new
KsLoadManager.FeedAdListener() {
    @Override
    public void onError(int code, String msg) {
      if (mListView != null) {
        mListView.setLoadingError();
      }
      ToastUtil.showToast(mContext, "广告数据请求失败" + code + msg);
    }

    @Override
    public void onFeedAdLoad(@Nullable List<KsFeedAd> adList) {
      if (mListView != null) {
        mListView.setLoadingFinish();
      }
      if (adList == null || adList.isEmpty()) {
        ToastUtil.showToast(mContext, "广告数据为空");
        return;
      }
      int loadCount = 10;// 模拟每次展示的Item刷新个数
      for (int i = 0; i < loadCount; i++) {
        mFeedList.add(null);
      }
      int totalCount = mFeedList.size();
      for (KsFeedAd ksFeedAd : adList) {
        if (ksFeedAd == null) {
          continue;
        }
        int random = (int) (Math.random() * loadCount) + totalCount -
loadCount;
        mFeedList.set(random, ksFeedAd);
      }
      mFeedListAdapter.notifyDataSetChanged();
    }
  });
}

//2.获取SDK广告模板
private View getAdItemView(View convertView, ViewGroup parent, final KsFeedAd
ksFeedAd) {
      AdViewHolder adViewHolder;
      if (convertView == null) {
        convertView =
LayoutInflater.from(mContext).inflate(R.layout.feed_list_item_ad_container,
            parent, false);
```

```
            adViewHolder = new AdViewHolder(convertView);
            convertView.setTag(adViewHolder);
        } else {
            adViewHolder = (AdViewHolder) convertView.getTag();
        }
        // 设置监听
        // ksFeedAd.setVideoSoundEnable(false);//视频播放是否，默认静音播放
        ksFeedAd.setAdInteractionListener(new KsFeedAd.AdInteractionListener() {
            @Override
            public void onAdClicked() {
                ToastUtil.showToast(mContext, "广告点击回调");
            }

            @Override
            public void onAdShow() {
                ToastUtil.showToast(mContext, "广告曝光回调");
            }

            @Override
            public void onDislikeClicked() {
                ToastUtil.showToast(mContext, "广告不喜欢回调");
                mFeedList.remove(ksFeedAd);
                notifyDataSetChanged();
            }

            @Override
            public void onDownloadTipsDialogShow() {
                ToastUtil.showToast(mContext, "广告展示下载合规弹窗");
            }

            @Override
            public void onDownloadTipsDialogDismiss() {
                ToastUtil.showToast(mContext, "广告关闭下载合规弹窗");
            }
        });
        View videoView = ksFeedAd.getFeedView(mContext);
        if (videoView != null && videoView.getParent() == null) {
            adViewHolder.mAdContainer.removeAllViews();
            adViewHolder.mAdContainer.addView(videoView);
        }
        return convertView;
    }
```

推荐使用: 自定义信息流模版的广告请求接口。 详细示例参考附件demo中的 `configFeed.TestFeedListActivity`、`configFeed.TestFeedRecyclerActivity`

```
private void requestAd(long posId, int width) {
```

```
  KsScene scene = new KsScene.Builder(posId) // 此为测试posId，请联系快手平台申请正式posId
      .width(width)
      .adNum(3) // 支持返回多条广告，默认1条，最多5条，参数范围1-5
      .build();

  KsAdSDK.getLoadManager().loadConfigFeedAd(scene, new KsLoadManager.FeedAdListener() {
      @Override
      public void onError(int code, String msg) {
        if (mListView != null) {
          mListView.setLoadingError();
        }
        ToastUtil.showToast(mContext, "广告数据请求失败" + code + msg);
      }

      @Override
      public void onFeedAdLoad(@Nullable List<KsFeedAd> adList) {
        if (mListView != null) {
          mListView.setLoadingFinish();
        }
        if (adList == null || adList.isEmpty()) {
          ToastUtil.showToast(mContext, "广告数据为空");
          return;
        }
        int loadCount = 10;// 模拟每次展示的Item刷新个数
        for (int i = 0; i < loadCount; i++) {
          mFeedList.add(null);
        }
        int totalCount = mFeedList.size();
        for (KsFeedAd ksFeedAd : adList) {
          if (ksFeedAd == null) {
            continue;
          }
          int random = (int) (Math.random() * loadCount) + totalCount - loadCount;
          mFeedList.set(random, ksFeedAd);
        }
        mFeedListAdapter.notifyDataSetChanged();
      }
    });
}

//2.获取SDK广告模板
private View getAdItemView(View convertView, ViewGroup parent, final KsFeedAd ksFeedAd) {
      AdViewHolder adViewHolder;
      if (convertView == null) {
```

```
        convertView =
LayoutInflater.from(mContext).inflate(R.layout.feed_list_item_ad_container,
            parent, false);
        adViewHolder = new AdViewHolder(convertView);
        convertView.setTag(adViewHolder);
      } else {
        adViewHolder = (AdViewHolder) convertView.getTag();
      }
      // 设置监听
      // ksFeedAd.setVideoSoundEnable(false);//视频播放是否，默认静音播放
      ksFeedAd.setAdInteractionListener(new KsFeedAd.AdInteractionListener() {
        @Override
        public void onAdClicked() {
          ToastUtil.showToast(mContext, "广告点击回调");
        }

        @Override
        public void onAdShow() {
          ToastUtil.showToast(mContext, "广告曝光回调");
        }

        @Override
        public void onDislikeClicked() {
          ToastUtil.showToast(mContext, "广告不喜欢回调");
          mFeedList.remove(ksFeedAd);
          notifyDataSetChanged();
        }

        @Override
        public void onDownloadTipsDialogShow() {
          ToastUtil.showToast(mContext, "广告展示下载合规弹窗");
        }

        @Override
        public void onDownloadTipsDialogDismiss() {
          ToastUtil.showToast(mContext, "广告关闭下载合规弹窗");
        }
      });
      View videoView = ksFeedAd.getFeedView(mContext);
      if (videoView != null && videoView.getParent() == null) {
        adViewHolder.mAdContainer.removeAllViews();
        adViewHolder.mAdContainer.addView(videoView);
      }
      return convertView;
    }
```

4.5.2 KsFeedAd接口说明

```
public interface KsFeedAd {
```

```
  /**
   * 获取SDK渲染的模板View
      */
    @Nullable
    View getFeedView(Context context);

  /**
   * 注册页面交互回调接口
      *
   * @param listener 交互监听器
      */
    void setAdInteractionListener(AdInteractionListener listener);


  /**
   * 视频播放是否有声
      *
   * @param enable 默认false静音
      */
    void setVideoSoundEnable(boolean enable);

  /**
   * 获取ecpm，单位：分，默认为0（使用该功能需同步商务申请使用权限）
      */
    int getECPM();

  /**
   * 媒体回传二价ecpm，竞价成功后，必须在展示前回传（使用该功能需同步商务申请使用权限）
      *
   * @param bidEcpm 单位：分
      */

  void setBidEcpm(int bidEcpm);

  /**
   * 广告曝光失败后上报失败原因
   * @param adExposureFailureCode 曝光失败类型
   *        当失败类型为AdExposureFailureCode.BID_FAILED时，
   *        曝光失败原因请上报胜出的AdExposureFailedReason.winEcpm值
   * @param adExposureFailedReason 曝光失败原因描述
   */
  void reportAdExposureFailed(@AdExposureFailureCode int
adExposureFailureCode,
        AdExposureFailedReason adExposureFailedReason);

   /**
   * 配置播放器
   *
   * @param videoPlayConfig 播放器配置
```

```
     */
    void setVideoPlayConfig(@Nullable KsAdVideoPlayConfig videoPlayConfig);

    /**
     * 广告素材类型
     */
    @MaterialType
    int getMaterialType();

    /**
     * 广告操作类型
     */
    @InteractionType
    int getInteractionType();

    /**
     * 广告交互回调接口
        */
      interface AdInteractionListener {

        /**
         * 广告点击的回调，点击后的动作由sdk控制
         */
        void onAdClicked();

        /**
         * 广告展示回调 每个广告仅回调一次
         */
        void onAdShow();

        /**
         * 不喜欢按钮点击回调
         */
        void onDislikeClicked();

        /**
         * 下载类广告展示下载合规弹窗
         */
        void onDownloadTipsDialogShow();

        /**
         * 下载类广告关闭下载合规弹窗
         */
        void onDownloadTipsDialogDismiss();
    }
}
```

4.6 请求Draw竖屏信息流广告

SDK为媒体提供了竖屏信息流广告样式，该广告样式时候适用于全屏的竖屏视频中使用，类似于快手APP的大屏版模式。

4.6.1 请求示例

详细示例参考附件demo中的`TestDrawVideoActivity`

```
// 1.请个Draw信息流广告，获取广告对象KsDrawAd
private void requestAd(long posId) {
   KsScene scene = new KsScene.Builder(posId) // 此为测试posId，请联系快手平台申请正式posId
    .adNum(3) // 支持返回多条广告，默认1条，最多5条，参数范围1-5
    .build();
  KsAdSDK.getLoadManager().loadDrawAd(scene, new KsLoadManager.DrawAdListener() {
    @Override
    public void onError(int code, String msg) {
      ToastUtil.showToast(mContext, "广告数据请求失败" + code + msg);
    }

    @Override
    public void onDrawAdLoad(@Nullable List<KsDrawAd> adList) {
      if (adList == null || adList.isEmpty()) {
        ToastUtil.showToast(mContext, "广告数据为空");
        return;
      }
      List<TestItem.NormalVideo> normalVideoList = getTestVideo();
      for (TestItem.NormalVideo normalVideo : normalVideoList) {
        mDrawList.add(new TestItem(normalVideo, null));
      }
      for (KsDrawAd ksDrawAd : adList) {
        if (ksDrawAd == null) {
          continue;
        }
        int random = (int) (Math.random() * 100);
        int index = random % normalVideoList.size();
        if (index == 0) {
          index++;
        }
        mDrawList.add(index, new TestItem(null, ksDrawAd));
      }
      mRecyclerAdapter.notifyDataSetChanged();
    }
  });
}

// 2.
KsDrawAd ksDrawAd = item.ksDrawAd;
ksDrawAd.setAdInteractionListener(new KsDrawAd.AdInteractionListener() {
```

```
  @Override
  public void onAdClicked() {
    ToastUtil.showToast(mContext, "广告点击回调");
  }

  @Override
  public void onAdShow() {
    ToastUtil.showToast(mContext, "广告曝光回调");
  }
});

View drawVideoView = ksDrawAd.getDrawView(mContext);
if (drawVideoView != null && drawVideoView.getParent() == null) {
  drawViewHolder.mVideoContainer.removeAllViews();
  drawViewHolder.mVideoContainer.addView(drawVideoView);
}
```

4.6.2 KsDrawAd接口说明

```
public interface KsDrawAd {

    /**
     * 返回ecpm值，单位：分，默认为0，对外文档不可见
     */
    int getECPM();

    /**
     * 媒体返回二价ecpm，在展示前调用。
     *
     * @param bidEcpm 单位：分
     */
    void setBidEcpm(int bidEcpm);

    /**
     * 广告曝光失败后上报失败原因
     * @param adExposureFailureCode 曝光失败类型
     *        当失败类型为AdExposureFailureCode.BID_FAILED时，
     *        曝光失败原因请上报胜出的AdExposureFailedReason.winEcpm值
     * @param adExposureFailedReason 曝光失败原因描述
     */
    void reportAdExposureFailed(@AdExposureFailureCode int
adExposureFailureCode,
        AdExposureFailedReason adExposureFailedReason);

    /**
     * 获取SDK渲染的模板View
     */
    @Nullable
    View getDrawView(Context context);
```

```
/**
 * 注册页面交互回调接口
 *
 * @param listener 交互监听器
 */
void setAdInteractionListener(AdInteractionListener listener);


/**
 * 广告素材类型
 */
@MaterialType
int getMaterialType();

/**
 * 广告操作类型
 */
@InteractionType
int getInteractionType();

/**
 * 广告交互回调接口
 */
interface AdInteractionListener {

    /**
     * 广告点击的回调，点击后的动作由sdk控制
     */
    void onAdClicked();

    /**
     * 广告展示回调 每个广告仅回调一次
     */
    void onAdShow();


  /**
   * 视频开始播放
   */
  void onVideoPlayStart();

  /**
   * 视频暂停播放
   */
  void onVideoPlayPause();

  /**
   * 视频恢复播放
```

```
         */
        void onVideoPlayResume();


        /**
         * 视频播放结束
         */
        void onVideoPlayEnd();

        /**
         * 视频播放错误
         */
        void onVideoPlayError();
    }
}
```

4.7 请求原生广告数据

在SDK提供的广告样式不满足媒体自身样式的场景下，媒体可以通过请求原生广告数据，自行渲染广告界面，目前只建议媒体自渲染信息流广告，使用前联系快手申请对应场景posid。

4.7.1 请求示例

详细示例参考附件demo中的`TestNativeAdActivity`

```
//1.请求广告
public void requestNativeAd(View view) {
    int posId = 90009004;// 此为测试posId，请联系快手平台申请正式posId
     KsScene scene = new KsScene.Builder(posId) // 此为测试posId，请联系快手平台申请正式posId
    .adNum(1) // 支持返回多条广告，默认1条，最多5条，参数范围1-5
    .build();
    KsAdSDK.getLoadManager().loadNativeAd(scene, new KsLoadManager.NativeAdListener() {
        @Override
        public void onError(int code, String msg) {
          Toast.makeText(TestNativeAdActivity.this, "广告数据请求失败" + code + msg, Toast.LENGTH_SHORT)
              .show();
        }

        @Override
        public void onNativeAdLoad(@Nullable List<KsNativeAd> adList) {
          if (adList == null || adList.isEmpty()) {
            Toast.makeText(TestNativeAdActivity.this, "广告数据为空", Toast.LENGTH_SHORT).show();
            return;
          }
```

```
        Toast.makeText(TestNativeAdActivity.this, "广告数据加载成功",
Toast.LENGTH_SHORT).show();
        showAd(adList);
      }
    });
}

//2.展示广告
private void showAd(@NonNull List<KsNativeAd> adList) {
    mNativeAdContainer.removeAllViews();
    for (KsNativeAd ksNativeAd : adList) {
      View view;
      // 判断广告素材类型
      switch (ksNativeAd.getMaterialType()) {
        case MaterialType.VIDEO:
          // 视频素材，渲染自定义的视频广告
          view = getVideoItemView(mNativeAdContainer, ksNativeAd);
          break;
        case MaterialType.SINGLE_IMG:
          // 单图素材，渲染自定义的单图广告
          view = getSingleImageItemView(mNativeAdContainer, ksNativeAd);
          break;
        case MaterialType.GROUP_IMG:
          // 组图素材，渲染自定义的组图广告
          view = getGroupImageItemView(mNativeAdContainer, ksNativeAd);
          break;
        case MaterialType.UNKNOWN:
        default:
          view = getNormalItemView(mNativeAdContainer);
      }
      if (adView != null && adView.getParent() == null) {
          mNativeAdContainer.addView(adView);
      }
    }
}
```

4.7.2 KsNativeAd接口说明

```
public interface KsNativeAd {

  /**
   * 广告描述
   */
  String getAdDescription();

  /**
   * 获取非下载广告的产品名称
   */
  String getProductName();
```

```
/**
 * 广告来源，可能为空
 */
@Nullable
String getAdSource();

/**
 * 获取广告角标的logo
 * @param logoType  角标样式类型
 * @return
 */
@Nullable
String getAdSourceLogoUrl(@AdSourceLogoType int logoType);

/**
 * 广告图片集合，单图和组图类型广告素材有返回，视频类素材返回为空
 */
@Nullable
List<KsImage> getImageList();

/**
 * 下载类型的AppIcon，非下载返回为空
 */
@Nullable
String getAppIconUrl();

/**
 * 下载类型的AppName，非下载返回为空
 */
@Nullable
String getAppName();

/**
 * 应用下载次数文案，非下载返回为空
 * eg：1000W此下载
 */
@Nullable
String getAppDownloadCountDes();

/**
 * 应用下载评分，取值0-5.0；非下载返回为0
 */
float getAppScore();

/**
 * 获取开发者主体
 */
@Nullable
```

```
    String getCorporationName();

    /**
     * 获取应用权限信息
     */
    @Nullable
    String getPermissionInfo();

    /**
     * 获取应用权限信息链接
     */
    @Nullable
    String getPermissionInfoUrl();

    /**
     * 获取隐私条款链接
     */
    @Nullable
    String getAppPrivacyUrl();

    /**
     * 获取隐私条款链接
     */
    @Nullable
    String getAppVersion();

    /**
     * 获取App包名
     */
    @Nullable
    String getAppPackageName();

    /**
     * 获取下载包大小
     */
    long getAppPackageSize();

    /**
     * 获取视频view
     *
     * @param videoSoundEnable 是否静音播放
     */
    @Nullable
    View getVideoView(Context context, boolean videoSoundEnable);

    /**
     * 视频类型广告素材，返回视频资源地址，非视频类素材返回为空
     */
    @Nullable
```

```
String getVideoUrl();

/**
 * 视频类型广告素材，返回视频封面素材，非视频类素材返回为空
 */
@Nullable
KsImage getVideoCoverImage();

/**
 * 视频时长，单位秒，非视频类素材返回0
 */
int getVideoDuration();

/**
 * 广告默认转化推荐的文案
 */
String getActionDescription();

/**
 * 广告素材类型:视频，单图，组图
 */
@MaterialType
int getMaterialType();

/**
 * 广告操作类型:h5,下载
 */
@InteractionType
int getInteractionType();

/**
 * 获取ecpm，单位：分，默认为0（使用该功能需同步商务申请使用权限）
 */
int getECPM();

/**
 * 媒体回传二价ecpm，竞价成功后，必须在展示前回传（使用该功能需同步商务申请使用权限）
 *
 * @param bidEcpm 单位：分
 */
void setBidEcpm(int bidEcpm);

/**
 * 广告曝光失败后上报失败原因
 * @param adExposureFailureCode 曝光失败类型
 *        当失败类型为AdExposureFailureCode.BID_FAILED时，
 *        曝光失败原因请上报胜出的AdExposureFailedReason.winEcpm值
 * @param adExposureFailedReason 曝光失败原因描述
 */
```

```
  void reportAdExposureFailed(@AdExposureFailureCode int
adExposureFailureCode,
        AdExposureFailedReason adExposureFailedReason);

   /**
    * 注册可点击的View，sdk内部判断回调click或show事件
    * 已废弃，使用 {@link KsNativeAd#registerViewForInteraction(Activity,
ViewGroup, List, AdInteractionListener)} 代替
    *
    * @param container  渲染广告最外层的ViewGroup
    * @param clickViews 可转化点击的View的列表，转化操作由sdk执行
    */
  @Deprecated
  void registerViewForInteraction(@NonNull ViewGroup container, @NonNull
List<View> clickViews,
                                  AdInteractionListener listener);


  /**
    * 注册可点击的View，sdk内部判断回调click或show事件
    *
    * @param activity   渲染广告所依赖的Activity容器，需要是一个当前可见的Activity。
    *                   需要依赖该Activity对象来显示弹窗，不传可能会导致弹窗无法展示
    * @param container  渲染广告最外层的ViewGroup
    * @param clickViews 可转化点击的View的列表，转化操作由sdk执行
    */
  void registerViewForInteraction(Activity activity
      , @NonNull ViewGroup container, @NonNull List<View> clickViews,
      AdInteractionListener listener);

     /**
    * 注册可点击的View，sdk内部判断回调click或show事件
    *
    * @param activity   渲染广告所依赖的Activity容器，需要是一个当前可见的Activity。
    *                   需要依赖该Activity对象来显示弹窗，不传可能会导致弹窗无法展示
    * @param container  渲s染广告最外层的ViewGroup
    * @param clickViews Key为可转化点击的View，Value为对应的转化行为 转化操作由sdk执行
    * @see com.kwad.sdk.api.model.KsNativeConvertType
    *
    * 注！！！请联系联盟SDK商务配置相关App开关,否则该方法设置对应的转化行为后不生效
    */
  @KsAdSdkApi
  @Keep
  void registerViewForInteraction(Activity activity
      , @NonNull ViewGroup container, @NonNull Map<View, Integer> clickViews
      ,AdInteractionListener listener);

  /**
    * 下载类型，可根据需要设置下载监听器，非下载类设置无线
```

```
     */
    void setDownloadListener(KsAppDownloadListener downloadListener);

    /**
     * 广告交互回调接口
     */
    interface AdInteractionListener {
        /**
         * 广告点击的回调，点击后的动作由sdk控制
         *
         * @param ad KsNativeAd原生广告对象
         */
        void onAdClicked(View view, KsNativeAd ad);

        /**
         * 广告展示回调 每个广告仅回调一次
         *
         * @param ad KsNativeAd原生广告对象
         */
        void onAdShow(KsNativeAd ad);

        /**
         * 下载类广告展示下载合规弹窗
         */
        void onDownloadTipsDialogShow();

        /**
         * 下载类广告关闭下载合规弹窗
         */
        void onDownloadTipsDialogDismiss();
    }
}
```

4.7.3 MaterialType素材类型

```
@IntDef({MaterialType.UNKNOWN, MaterialType.VIDEO, MaterialType.SINGLE_IMG,
        MaterialType.GROUP_IMG})
@interface MaterialType {
    int UNKNOWN = 0;
    int VIDEO = 1;
    int SINGLE_IMG = 2;
    int GROUP_IMG = 3;
}
```

4.7.4 InteractionType转化类型

```
@IntDef({InteractionType.H5, InteractionType.DOWNLOAD,
InteractionType.UNKNOWN})
@interface InteractionType {
  int UNKNOWN = 0;
  int DOWNLOAD = 1;
  int H5 = 2;
}
```

4.7.5 AdSourceLogoType广告角标类型

```
/**
 * 广告角标类型
 */
@Keep
@IntDef({AdSourceLogoType.NORMAL, AdSourceLogoType.GREY})
public @interface AdSourceLogoType {
  /**
   * 普通模式
   */
  int NORMAL = 0;
  /**
   * 夜间模式
   */
  int GREY = 1;
}
```

4.7.6 DownloadListener广告下载监听

```
/**
 * 广告下载监听（建议使用，需配置白名单）
 */
@KsAdSdkApi
@Keep
public interface KsApkDownloadListener extends KsAppDownloadListener {
  /**
   * 下载暂停
   */
  @KsAdSdkApi
  @Keep
  void onPaused(int progress);
}

/**
 * 广告common下载监听
 */
@KsAdSdkApi
@Keep
public interface KsAppDownloadListener {
```

```
    /**
     * 未开始下载
     */
    @KsAdSdkApi
    @Keep
    void onIdle();

    /**
     * 下载中回调
     */
    @KsAdSdkApi
    @Keep
    void onDownloadStarted();

    /**
     * 下载中回调
     */
    @KsAdSdkApi
    @Keep
    void onProgressUpdate(int progress);

    /**
     * 下载完成回调
     */
    @KsAdSdkApi
    @Keep
    void onDownloadFinished();

    /**
     * 安装完成回调
     */
    @KsAdSdkApi
    @Keep
    void onInstalled();

    /**
     * 下载失败
     */
    @KsAdSdkApi
    @Keep
    void onDownloadFailed();
}
```

4.8 请求开屏广告数据

在SDK提供的开屏视频广告，在APP启动页展示，使用前联系快手申请对应场景posid。

4.8.1 开屏小窗模式

新增开屏小窗模式，可以在用户开屏视频播放一段时间后，开屏广告结束跳转到主页后，开屏视频缩小为小窗口继续播放，以提高开屏广告的转化。接入方式如下:

1 用户需要在 KsScene中配置 needShowMiniWindow(true) 打开小窗模式。

2 如果回调onSkippedAd， 需要用静态变量保存 KsSplashScreenAd

3 如果回调onAdShowEnd， 需要用静态变量保存 KsSplashScreenAd

4 如果开屏是一个独立的Activity, 而不是覆盖在主页面之上的一个页面，请在开屏Activity 切换中通过overridePendingTransition(0, 0); 禁用出场入场动画。

5 如果开屏广告是一个独立的Activity, 而不是覆盖在主页面之上的一个页面, 需要在主页面展示小窗口，需要在主页面的onAttachedToWindow() 中调用 KsSplashScreenAd#showSplashMiniWindow()。

如果如果开屏广告是覆盖在主页面之上的一个页面，不是独立的Activity, 请在合适的地方KsSplashScreenAd#showSplashMiniWindow()。

```
boolean showSplashMiniWindow(Context context,
                    KsSplashScreenAd.SplashScreenAdInteractionListener
listener, Rect rect);
```

4.8.2 请求示例

1.请求开屏数据

```
// 1.请求开屏广告，获取广告对象，KsFullScreenVideoAd
  public void requestSplashScreenAd() {
    SplashAdExtraData extraData = new SplashAdExtraData();
    extraData.setDisableShakeStatus(true);
    KsScene scene = new KsScene
      //是否需要开屏小窗展示，默认为false，设置false后将不会回调 onShowMiniWindow
       .needShowMiniWindow(true)
      //是否需要屏蔽摇一摇功能
      .setSplashExtraData(extraData)
      .Build(TestPosId.POSID_SPLASHSCREEN.posId).build(); // 此为测试posId，请联
系快手平台申请正式posId
    if (KsAdSDK.getLoadManager() != null) {
      KsAdSDK.getLoadManager().loadSplashScreenAd(scene, new
KsLoaderManager.SplashScreenAdListener() {
        @Override
        public void onError(int code, String msg) {
          mSplashAdContainer.setVisibility(View.GONE);
          mEmptyView.setVisibility(View.VISIBLE);
          showTips("开屏广告请求失败" + code + msg);
          gotoMainActivity();
        }
```

```
        @Override
        public void onRequestResult(int adNumber) {
           ToastUtil.showToast(mContext, "开屏广告广告请求填充 " + adNumber);
        }

        @Override
        public void onSplashScreenAdLoad(@NonNull KsSplashScreenAd
splashScreenAd) {
          mSplashAdContainer.setVisibility(View.VISIBLE);
          //SplashAd.ksSplashScreenAd 为静态变量， 保存splashScreenAd用户小窗模式
          SplashAd.ksSplashScreenAd = splashScreenAd;
          addView(splashScreenAd);
        }
      });
    }
  }
```

2.添加开屏View

- 使用View 方式接入

```
private void addView(final KsSplashScreenAd splashScreenAd) {
  View view =
      splashScreenAd.getView(this,
          new KsSplashScreenAd.SplashScreenAdInteractionListener() {
            @Override
            public void onAdClicked() {
              showTips("开屏广告点击");
              /**
               * 开屏广告点击会吊起h5或应用商店，并回调onAdClick(),
mGotoMainActivity控制由h5或应用商店返回后是否直接进入主界面
               * 建议当需要展示开屏小窗时设置为返回后进入主界面，其他情况设为false以继续执
行开屏的倒计时
               */
              mGotoMainActivity = mNeedShowMiniWindow;
              SplashAd.ksSplashScreenAd = null;
            }

            @Override
            public void onAdShowError(int code, String extra) {
              showTips("开屏广告显示错误 " + code + " extra " + extra);
              //出错不触发显示miniWindow
              SplashAd.ksSplashScreenAd = null;
              gotoMainActivity();
            }

            @Override
            public void onAdShowEnd() {
              showTips("开屏广告显示结束");
```

```
            gotoMainActivity();
          }

          @Override
          public void onAdShowStart() {
            showTips("开屏广告显示开始");
            mEmptyView.setVisibility(View.GONE);
          }

          @Override
          public void onSkippedAd() {
            showTips("用户跳过开屏广告");
            gotoMainActivity();
          }
        });

  if (!isFinishing()) {
    ViewGroup root = findViewById(R.id.splash_ad_container);
    root.removeAllViews();
    view.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT));
    root.addView(view);
  }
}
```

3.接入开屏V+处理

在主页面（要展示缩小窗的页面）MainActivity中

```
@Override
public void onAttachedToWindow() {
  super.onAttachedToWindow();
  if (SplashAd.ksSplashScreenAd != null) {
    Rect rect = new Rect();
    DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
    rect.right = displayMetrics.widthPixels;
    rect.left =  rect.right - displayMetrics.widthPixels / 4;
    rect.bottom = (int) (displayMetrics.heightPixels * 0.83f);
    rect.top = rect.bottom - (displayMetrics.widthPixels / 4) * 16 / 9;

    SplashAd.ksSplashScreenAd.showSplashMiniWindowIfNeeded(this,
        new KsSplashScreenAd.SplashScreenAdInteractionListener() {
          @Override
          public void onAdClicked() {
          }

          @Override
          public void onAdShowError(int code, String extra) {
```

```
            }

            @Override
            public void onAdShowEnd() {
            }

            @Override
            public void onAdShowStart() {
            }

            @Override
            public void onSkippedAd() {
            }
        }, rect);
    SplashAd.ksSplashScreenAd = null;
  }
}
```

4.8.3 KsSplashScreenAd接口说明

```
public interface KsSplashScreenAd {
  /**
   * 判断该广告是否可用（主要是素材是否已缓存成功）
      *
   * @return true表示视频可用，false表示不可用
      */
    boolean isAdEnable();

  /**
   * 获取开屏幕视频的View.
      *
   * @param listener 交互监听器
   * @return 开屏View. **注意** 每次获取都是新的实例
      */
    @KsAdSdkApi
    @Keep
    View getView(Context context, SplashScreenAdInteractionListener listener);

  /**
   * 设置 交互监听器
   * @param listener 交互监听器
   * @return Fragment，需要监听的Fragment
      */
    void setListener(KsSplashScreenFragment fragment, @Nullable
SplashScreenAdInteractionListener listener);

  /**
```

```
     * 判断该广告是为视频
     *
     * @return true表示视频，false表示图文 版本2.7.1只支持视频
     */
    boolean isVideo();

    /**
     * 广告素材类型:视频，单图，组图
     */
    @MaterialType
    int getMaterialType();

    /**
     * 广告操作类型:h5,下载
     */
    @InteractionType
    int getInteractionType();

    /**
     * 展示开屏小窗口
     * 该方法在应在当前Activity onAttachedToWindow 调用
     * @param context 传入的context需要是一个当前展示的Activity的contetxt
     * @param listener 当前miniWindow的 用户操作监听
     * @param rect miniWindow窗口展示的区域，单位是px
     */
    @MainThread
    @KsAdSdkApi
    @Keep
    boolean showSplashMiniWindow(Context context,

KsSplashScreenAd.SplashScreenAdInteractionListener listener,
                                 Rect rect);

    /**
     * 开屏广告交互监听器
     */
    @KsAdSdkApi
    interface SplashScreenAdInteractionListener {

        /**
         * 开屏广告点击回调
         */
        void onAdClicked();

        /**
         * 开屏广告播放出错
         */
        void onAdShowError(int code, String extra);
```

```
    /**
     * 开屏广告播放完成
     */
    void onAdShowEnd();

    /**
     * 开屏广告播放开始
     */
    void onAdShowStart();

    /**
     * 跳过开屏广告播放
     */
    void onSkippedAd();

    /**
     * 开屏广告出现合规下载弹窗
     */
    void onDownloadTipsDialogShow();

    /**
     * 开屏广告的合规下载弹窗消失
     * 是用户点击了下载转化后触发的关闭
     */
    void onDownloadTipsDialogDismiss();

    /**
     * 开屏广告的合规下载弹窗消失
     * 用户取消了下载触发的关闭
     */
    void onDownloadTipsDialogCancel();
  }
}
```

4.9 请求插屏广告数据

在SDK提供的插屏广告，在媒体插屏场景下展示，使用前联系快手申请对应场景posid。

4.9.1 请求示例

```
// 1.请求插屏广告，获取广告对象，InterstitialAd
  public void requestInterstitialAd(View view) {
    mKsInterstitialAd = null;
     // 此为测试posId，请联系快手平台申请正式 posId
    KsScene scene = new
KsScene.Builder(TestPosId.POSID_INTERSTITIAL.posId).build();
    KsAdSDK.getLoadManager().loadInterstitialAd(scene,
```

```
        new KsLoadManager.InterstitialAdListener() {
          @Override
          public void onRequestResult(int adNumber) {
            ToastUtil.showToast(mContext, "插屏广告请求填充个数 " + adNumber);
          }
          @Override
          public void onInterstitialAdLoad(@Nullable List<KsInterstitialAd>
adList) {
            if (adList != null && adList.size() > 0) {
              mKsInterstitialAd = adList.get(0);
              ToastUtil.showToast(mContext, "插屏广告请求成功");
              KsVideoPlayConfig videoPlayConfig = new
KsVideoPlayConfig.Builder()
                  .build();
              showInterstitialAd(videoPlayConfig);
            }
          }
        });
  }
```

4.9.2 KsInterstitialAd接口说明

```
/**
 * 插屏广告
 * */
public interface KsInterstitialAd {

  /**
   * 返回ecpm值，单位：分，默认为0，对外文档不可见
   */
  int getECPM();

  /**
   * 媒体返回二价ecpm，在展示前调用。
   *
   * @param bidEcpm 单位：分
   */
  void setBidEcpm(int bidEcpm);

  /**
   * 广告曝光失败后上报失败原因
   * @param adExposureFailureCode 曝光失败类型
   *        当失败类型为AdExposureFailureCode.BID_FAILED时，
   *        曝光失败原因请上报胜出的AdExposureFailedReason.winEcpm值
   * @param adExposureFailedReason 曝光失败原因描述
   */
  void reportAdExposureFailed(@AdExposureFailureCode int
adExposureFailureCode,
        AdExposureFailedReason adExposureFailedReason);
```

```
  /**
   * 展示插屏广告
   *
   * @param activity   宿主activity,用于判断宿主是否finishing
   * @param videoPlayConfig 视频展示配置
   */
  void showInterstitialAd(Activity activity, KsVideoPlayConfig
videoPlayConfig);

  /**
   * 注册页面交互回调接口
   *
   * @param listener 交互监听器
   */
  void setAdInteractionListener(AdInteractionListener listener);

  /**
   * 判断该广告是为视频
   * 已废弃， 推荐使用{@link #getMaterialType()}.
   * @return true表示视频，false表示图文
   */
  @Deprecated
  boolean isVideo();

  /**
   * 广告素材类型:视频，单图，组图
   */
  @MaterialType
  int getMaterialType();

  /**
   * 广告操作类型:h5,下载
   */
  @InteractionType
  int getInteractionType();

  /**
   * 广告交互回调接口
   */
  interface AdInteractionListener {

    /**
     * 广告点击的回调，点击后的动作由sdk控制
     */
    void onAdClicked();

    /**
     * 广告展示回调 每个广告仅回调一次
```

```
     */
    void onAdShow();

    /**
     * 关闭按钮点击回调
     */
    void onAdClosed();

    /**
     * 插屏广告页面关闭
     */
    void onPageDismiss();

    /**
     * 插屏广告 视频播放出错
     */
    void onVideoPlayError(int code, int extra);

    /**
     * 插屏广告 视频播放完成
     */
    void onVideoPlayEnd();

    /**
     * 插屏广告 视频播放开始
     */
    void onVideoPlayStart();

    /**
     * 跳过插屏广告
     */
    void onSkippedAd();
}
```

4.10 返回按钮

1 从快手返回

SDK内支持点击商品详情卡片跳转到快手主站对应商品详情页，并在商品详情页显示返回按钮的功能。默认配置下，点击返回按钮后，会直接返回到到媒体侧页面的，但是无法保证所有情况下均可以返回。

要保证所有情况下点击返回按钮，均可以返回到媒体侧页面，媒体侧需要在初始化SDK时，配置backurl进行返回跳转。操作步骤如下：

1. 构建场景时，设置相应的backurl属性：

```
    private void initContentPage() {
    KsScene adScene = new
KsScene.Builder(TestPosId.POSID_CONTENT_PAGE.posId).promoteId("1013").setBackU
rl("ksad://returnback").build();
    ......
  }
```

2. 该返回链接是以DeepLink的方式进行广播的，因此媒体侧需要配置相应接受该广播消息的Activity对象，如Demo中：

```
      <activity android:name=".open.contentalliance.TestReturnBackActivity"

 android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"
          android:launchMode="singleTask">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data
                  android:scheme="ksad"
                  />
            </intent-filter>
        </activity>
```

注意scheme名称需要与backurl内的scheme保持一致。这样，当在快手内点击返回按钮后，系统即可唤醒该Activity。默认情况下，直接关闭该Activity即可重新显示跳转之前的页面：

```
public class TestReturnBackActivity extends FragmentActivity {

  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    finish();
  }

}
```

如有更多需求，可参考Deeplink的使用方式自行处理。

4.11 退出App安装提示

SDK支持媒体在退出关闭app时，尝试展示最近下载且未安装的apk，再次安装提示的弹框。使用方式，通过

`KsAdSDK.getLoadManager().showInstallDialog`尝试展示，详细可参考Demo的`MainActivity`中的示例，接口说明：

```
/**
```

```
    * 展示下载安装弹框
    * @param activity，宿主页面，非空
    * @param listener，监控关闭事件，非空
    * @return 是否展示，弹框展示返回true，否则返回false
    */
boolean showInstallDialog(Activity activity, KsExitInstallListener listener);

@KsAdSdkApi
@Keep
public interface KsExitInstallListener {

  /**
   * 点击安装按钮
   */
  void onInstallClick();

  /**
   * 弹框关闭
   */
  @KsAdSdkApi
  @Keep
  void onDialogClose();

}
```

5. SDK错误码

请求广告数据失败时会回调错误接口`void onError(int code, String msg);`，常见错误码如下:

code	说明
40001	没有网络
40002	数据解析失败
40003	广告数据为空
40004	缓存视频资源失
100001	参数有误
100002	服务器错误
100003	不允许的操作
100004	服务不可用
310001	appId未注册
310002	appId无效
310003	appId已封禁
310004	packageName与注册的packageName不一致
310005	操作系统与注册的不一致
320002	appId对应账号无效
320003	appId对应账号已封禁
330001	posId未注册
330002	posId无效
330003	posId已封禁
330004	posid与注册的appId信息不一致

6. 常见问题

1. Android 9.0开始应用默认不支持http的请求，导致广告请求失败，资源和应用下载出现如下报错，请媒体自行适配http请求

   java.io.IOException: Cleartext HTTP traffic to static.yximgs.com not permitted

2. Android 9.0问题：Caused by: java.lang.RuntimeException: Using WebView from more than one process at once with the same data directory is not supported. https://crbug.com/558377

   如果您的应用targetSdkVersion为28，且应用内多个进程使用webview会出现此问题，原因是在Android P 以及之后版本不支持同时从多个进程使用具有相同数据目录的WebView。处理方式：1.排查您应用是否需要进程使用webview，如果不需要请只在主进程使用webview，例如只在主进程初始化一些第三方的SDK，2.如果您的应用需要保持多进程使用webview，请您按照谷歌官方建

议，在你的应用Application初始化其他SDK前，先调用WebView.setDataDirectorySuffix为不同进程设置不同的webview目录。

3. 请求视频广告失败：

   1.请核对广告SDK初始化的AppID是否正确；

   2.请核对请求广告时传入的广告场景参数posId是否正确。

   3.请根据返回的错误吗，参考错误码对照表知晓问题

4. 点击视频页面的APK下载，无法下载：

   1.下载APK的链接有https的，手机代理charles等工具时无法下载；

   2.检测您的网络是否正常，网络太差环境会出现下载超时无法下载。

5. SDK广告缓存清理机制

   1.激励视频和全屏视频广告请求成功后，会对需要资源提前进行缓存，缓存资源超过200m会自动清理；

   2.通过下载的apk会存储到本地文件，下载APK文件数超过10个时会自动清理。

6. 点击视频下载的APK安全问题：

   1.下载的apk存储路径是外部sdcard的下载目录

   2.未对下载的apk进行签名和md5校验，经验证安装在手机上的恶意APP可以替换下载包，如果走HTTP下载也可能被劫持替换。