金山云移动广告Android SDK快速接入文档 V4.1.0

广告SDK接入文档

更新日志

版本 4.1.0 [2018/11/02]
1.新增展示类奖励视频

版本 4.0.9 [2018/10/25]
1.新增开屏展示类广告和多种交互类型

版本 4.0.8 [2018/9/5]
1.新增开屏类型广告位支持，以及对应API接口及文档说明

版本 4.0.7 [2018/8/20]
1.修改file_paths名称，改为ksyun_file_paths
2.添加Tracking事件上报回调接口

版本 4.0.5 [2018/4/23]
1.添加部分统计信息维度
2.去除原FileProvider依赖的xml文件中，需要手动填写包名的限制
3.新增Android 8.0 允许安装未知来源应用权限处理
4.新增接入流程中测试验证过程的详细说明

版本 4.0.4 [2018/4/9]
1.添加奔溃统计，修复部分Bug
2.调整奖励视频观看途中退出后的交互逻辑，由直接退出改成展示落地页
3.新增自动缓存广告位支持，通过setAutoCacheAd(String adSlotId)接口设置，使用后无需客户再手动调用load接口加载，SDK会自动拉取广告

版本 4.0.3 [2018/3/9]

1.修改preloadAd接口，改名为loadAd接口
2.去除原有的hasLocalAd接口，后续统一使用hasAd接口

版本 4.0.2 [2018/3/5]
1.修复偶现的关闭广告按钮不出现问题
2.修复部分机型上，Home键退出后，返回APP时奖励视频播放退出问题。
3.修改FileProvider包名，避免集成其它SDK时发生冲突

版本 4.0.1 [2018/1/26]
1.新增hasLocalAd接口
2.新增沙盒环境

版本 4.0.0 [2017/12/15]
1.初版更新

目录

1、文件清单

SDK包含的文件，如下所示：

- SDK demo工程
- jar包形式SDK，以及assets文件夹（包含SDK初始插件apk）
- aar形式SDK（包含SDK初始插件apk）
- SDK快速接入文档.pdf
- SDK接口说明文档.pdf
- SDK常见问题FAQ.pdf

2、提供形式

SDK主要使用以下两种形式提供给客户：

- AAR文件形式（**推荐**）
- Jar包+Asset资源形式

客户可以根据自己工程的实际情况，任选其中一种方式进行集成即可

3、请求环境及接入流程说明

3.1、请求环境说明

SDK请求环境，分为沙盒环境(SANDBOX_ENV，用于测试)及线上环境(RELEASE_ENV，用于生产环境)；

初始化前可通过设置SDK配置项，变更SDK请求环境。

```
1.      KsyunAdSdkConfig config = new KsyunAdSdkConfig();
2.      setSdkEnvironment(KsyunAdSdkConfig.RELEASE_ENV);
```

```
1.  切记不要使用沙盒环境的配置上线，上线时务必使用线上环境
```

3.2、接入流程说明

3.2.1、整体测试流程

1、先使用沙盒环境对应的AppId(媒体Id)及广告位Id进行开发，集成SDK，并确认整个广告加载、展示流程通畅

2、跑通一次完整流程（见3.2.2、3.2.3），然后登陆运营管理平台，手动确认已通过平台测试验证环节，然后让运营同学帮忙打开线上环境广告位Id开关

3、修改SDK配置，设置为线上环境，使用对应的线上AppId(媒体Id)、广告位Id，进入生产环节测试

3.2.2、奖励视频详细测试流程

正常拉取到广告，点击播放视频并等待广告播放完成，然后落地页点击下载，至此为一个完整流程，不同的广告位，都需要进行此项流程，才能算通过测试验证

3.2.3、开屏广告详细测试流程

背景：
1.目前开屏广告支持三种类型：直接下载，跳转落地页下载，纯展示
2.从沙盒环境拉取到的开屏广告随机下发上面三种类型，需要每一种类型都测试通过之后，该广告位才能通知运营同学上线；

流程：
展示开屏广告，点击开屏广告：
1.若是直接下载类，成功触发下载，测试流程完成；
2.若是跳转落地页下载，打开落地页后，点击下载，成功触发下载，测试流程完成；
3.若是纯展示类广告，测试流程完成;

4、Unity导出Android工程接入说明

如果您的导出Android工程目录为Android Studio形式

- 建议使用AAR集成方式
- 需要额外添加Android Support V4 支持库

如果您的导出Android工程目录为Eclipse形式

- 建议使用Jar+Assets资源形式导入方式(AAR文件形式，eclipse环境下不支持)
- 需要额外添加Android Support V4 支持库，且添加的V4库版本，应与您在Eclipse环境下，编译使用的Target SDK版本一致

5、注意事项说明

5.1、关于SDK动态权限申请说明

默认情况下，6.0以上系统，SDK内部会在初始化的时候，向APP申请以下动态权限。
如果APP方不希望SDK申请权限，后续说明中的SDK配置项中，有对应的开关可以关闭权限申请。
对应的，APP应提供SDK运行所必须的动态权限。

- Manifest.permission.READ_PHONE_STATE（**必须**，用于生成唯一ID）
- Manifest.permission.ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION(**非必须**，用于地理位置相关)

5.2、关于Android 8.0 未知应用安装权限说明

Android 8.0及以上系统，在安装应用时需要添加以下权限，否则APK下载完成之后，不能成功唤起应用安装界面

- Manifest.permission.REQUEST_INSTALL_PACKAGES(**必须**，用于Android 8.0 安装未知

来源APK 授权）

5.3、关于混淆规制的说明

插件内部已经添加了相应的混淆逻辑，用户在集成时只需要添加以下混淆规则即可

```
    -keep class com.ksc.ad.sdk.**{ *;}
    -dontwarn com.ksc.ad.sdk.**
```

6、快速集成

SDK库文件导入

6.1、方式一，基于AAR文件形式（推荐）

1、在客户的app module工程，根目录libs文件夹（如不存在，创建一个即可）下，放置对应的aar包。

2、在app module的build.gradle文件中，添加如下所示代码

```
android {
    ...
    repositories {
        flatDir {
            dirs 'libs'
        }
    }
}

dependencies {
     //此处name字段中，应填写aar文件真实名称，这里仅以sdk-xxx.aar为例
     compile(name: 'sdk-xxx.aar', ext: 'aar')
}
```

3、**添加Manifest权限**

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission
android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
```

4、**注册SDK内部组件**

```
//此处添加provider，是为兼容7.0之后的应用自动安装问题
<provider
    android:name="com.ksc.ad.sdk.util.KsyunFileProvider"
    android:authorities="${applicationId}.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/ksyun_file_paths"/>
</provider>
```

5、**将SDK文件目录下，aar文件下xml目录，复制到工程的相应目录下，内容如下：**

```
<external-cache-path name="files_root" path="apk/."/>
```

6.2、方式二，基于Jar包+Asset资源形式

1、**在客户app module工程根目录libs文件夹（如不存在，创建一个即可）下，放置对应的jar包。**

2、**添加Manifest权限**

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

```
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission
android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
```

3、注册SDK内部组件

```
//奖励视频展示Activity
<activity
    android:name="com.ksc.ad.sdk.ui.AdProxyActivity"
     android:hardwareAccelerated="true"
    android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
    android:configChanges="keyboardHidden|orientation|screenSize" />
//动态权限申请，透明浮层Activity
<activity
    android:name="com.ksc.ad.sdk.ui.AdPermissionProxyActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"

android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"
/>
<service android:name="com.ksc.ad.sdk.service.AdProxyService" />
//此处添加provider，是为兼容7.0之后的应用自动安装问题
<provider
    android:name="com.ksc.ad.sdk.util.KsyunFileProvider"
    //注意下方的authorities中com.xxx部分取值，需要填写用户自己的包名
    android:authorities="com.xxx.xxx.xxx.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
```

```
        android:resource="@xml/ksyun_file_paths"/>
</provider>
```

4、将SDK文件目录下，aar文件下xml目录，复制到工程的相应目录下，内容如下：

```
<external-cache-path name="files_root" path="apk/."/>
```

5、将SDK文件目录下，assets文件夹中的内容，复制到客户app module中src/main/assets目录下

7、奖励视频快速使用

7.1、初始化及加载广告

建议在App启动后，第一个页面onCreate时进行SDK初始化操作。初始化成功之后，在合适的场景下，提前加载广告。
如果没有调用KsyunAdSdkConfig的setSdkEnvironment()方法，设置SDK请求环境，默认则为沙盒环境。

```
class MainActivity extends Activity {

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

        KsyunAdSdkConfig config = new KsyunAdSdkConfig();
        //设置SDK请求环境为线上环境。SDK的init初始化方法，如果不设置config，默认则为沙盒环境
        config.setSdkEnvironment(KsyunAdSdkConfig.RELEASE_ENV);
        KsyunAdSdk.getInstance().init(MainActivity.this, "your_release_app_id", config, new IKsyunAdInitResultListener() {
            @Override
            public void onSuccess(Map<String, String> map) {
                //SDK初始化成功，设置事件监听
                KsyunAdSdk.getInstance().setAdListener(this);
                KsyunAdSdk.getInstance().setRewardVideoAdListener(this);
            }

            @Override
            public void onFailure(int errCode, String errMsg) {
```

```
                //SDK初始化失败处理
            }
        });
    }


    //在合适的场景下，提前加载广告
    public void onLevelStart(){
        //加载当前AppId(媒体Id)对应的所有广告位的广告
        KsyunAdSdk.getInstance().loadAd(new IKsyunAdLoadListener() {
            @Override
            public void onAdInfoSuccess() {
                //加载广告配置信息成功
            }

            @Override
            public void onAdInfoFailed(final int errCode, final String errMsg) {
                //加载广告配置信息失败
            }

            @Override
            public void onAdLoaded(final String adSlotId) {
                //广告位准备就绪，可以展示
            }
    }

}
```

7.2、展示广告

在广告位入口展示前，先调用hasAd方法判断当前广告位有无对应有效广告，根据结果决定是否展示入口。

广告展现后，待用户点击时，再调用showAd方法展示广告。

```
    //即将展示奖励视频入口前
    public void onGameOver(){
        //判断某广告位是否存在广告
        if(KsyunAdSdk.getInstance().hasAd(adslot_id)){
            //广告存在，点击奖励视频入口后，调用showAd接口展示广告

//KsyunAdSdk.getInstance().showAd(MainActivity.this,"YOUR_ADSLOT_ID");
        }else{
```

```
                //广告不存在,则调用loadAd接口，重新获取一次广告
                KsyunAdSdk.getInstance().loadAd(adslot_id,new
IKsyunAdLoadListener() {
                @Override
                public void onAdInfoSuccess() {
                    //加载广告配置信息成功
                }

                @Override
                public void onAdInfoFailed(final int errCode, final String
errMsg) {
                    //加载广告配置信息失败
                }

                @Override
                public void onAdLoaded(final String adSlotId) {
                    //广告位准备就绪，可以展示
                }
            }
        }
```

8、开屏广告快速使用

8.1、初始化及展示开屏广告

开屏广告的初始化及展示方法，无需依赖SDK初始化。

您可以在自己的SplashActivity的onCreate中直接调用以下代码，进行开屏初始化及展示操作。

(注：SDK 内部对已缓存但未展示的广告做了数据本地持久化，退出APP数据依旧会存在本地，所以可以直接调用接口展示开屏)

```
SplashAd mSplashAd =
KsyunAdSdk.getInstance().initSplashAd(SplashActivity.this,
mParentViewGroup,"mAdSlotId", 6, true, mSplashAdListener);
//初始化完成后，直接调用该方法展示，若不存在广告，则会直接回调SplashAdListener的
onSplashAdShowEnd回调方法。
mSplashAd.showAd();
```

8.2、加载开屏广告

加载开屏广告，需在SDK初始化成功之后。

方式1. 调用KsyunAdSdk.getInstance().loadAd(String adSlotId, IKsyunAdLoadListener preloadListener) 方法，进行加载开屏广告操作;
方式2. 调用loaAd(IKsyunAdLoadListener preloadListener)接口，加载APPID（媒体ID）下的所有广告的方式加载；

```
KsyunAdSdk.getInstance().loadAd(adSlotId, new IKsyunAdLoadListener() {
            @Override
            public void onAdInfoSuccess() {
                Log.d(DemoConstants.LoadSingleAd, "请求广告配置成功");
            }

            @Override
            public void onAdInfoFailed(int errCode, String errMsg) {
                Log.d(DemoConstants.LoadSingleAd, "请求广告配置失败");
            }

            @Override
            public void onAdLoaded(String s) {
                Log.d(DemoConstants.LoadSingleAd, "广告加载成功");
            }
        });
```

9、高级用法

9.1、SDK配置项

在调用init初始化方法之前，可以通过设置SDK配置项，来进行环境及功能的可选配置。具体支持的配置项定义及说明详情，请参见SDK接口文档附录表。

```
        KsyunAdSdkConfig config = new KsyunAdSdkConfig();
        //设置SDK请求环境为线上环境。SDK如果不设置config，默认则为沙盒环境
        config.setSdkEnvironment(KsyunAdSdkConfig.RELEASE_ENV);
        //设置奖励视频展示过程中，允许出现关闭按钮
        config.setShowCloseBtnOfRewardVideo(true);
        //设置奖励视频展示过程中，出现关闭按钮的时间点
        config.setCloseBtnComingTimeOfRewardVideo(5);

        KsyunAdSdk.getInstance().init(MainActivity.this, appId, config,
new IKsyunAdInitResultListener() {
```

```
            @Override
            public void onSuccess(Map<String, String> map) {

            }

            @Override
            public void onFailure(int errCode, String errMsg) {

            }
        });
```

9.2、广告事件监听

9.2.1 奖励视频广告事件监听

对于奖励视频类型的广告，广告事件监听主要依赖以下两个接口。

通过设置setAdListener接口，监听广告播放过程中用户对应的行为回调

```
public interface IKsyunAdListener {
    //广告展示成功时回调
    void onShowSuccess(String adSlotId);
    //广告展示失败时回调
    void onShowFailed(String adSlotId, int errCode,String errMsg);
    //广告内容播放完成，一般用于视频类广告
    void onADComplete(String adSlotId);
    //广告被点击
    void onADClick(String adSlotId);
    //广告被关闭
    void onADClose(String adSlotId);
}
```

通过设置setRewardVideoAdListener接口，可以监听奖励条件是否达成回调。

```
public interface IKsyunRewardVideoAdListener {
    //奖励条件达成
    void onAdAwardSuccess(String adSlotId);

    //奖励条件未达成
    void onAdAwardFailed(String adSlotId, int errCode, String errMsg);
}
```

9.3、广告加载事件监听

通过设置loadAd(IKsyunAdLoadListener loadListener)接口的回调参数loadListener，可以监听广告加载相应的事件

```
public interface IKsyunAdLoadListener {
    //请求配置获取成功，注意此回调仅仅是广告配置获取成功，不代表广告加载完成
    void onAdInfoSuccess();

    //请求配置获取失败
    void onAdInfoFailed(int errCode, String errMsg);

    //广告加载完成
    void onAdLoaded(String adSlotId);
}
```