

金山云移动广告SDK快速接入文档 V4.0.1

2018-1-8 V1.0.0 版本 - 汤洛

1.初版更新

目录

金山云移动广告SDK快速接入文档 V4.0.1

目录

- 1、文件清单
- 2、提供形式
- 3、环境说明
- 4、Unity导出Android工程接入说明
- 5、SDK动态权限申请说明
- 6、快速集成
- 7、快速使用
 - 7.1、初始化及预加载
 - 7.2、展示广告
- 8、高级用法
 - 8.1、SDK配置项
 - 8.2、广告事件监听
 - 8.3、广告资源预加载事件监听
 - 8.4、关于视频广告播放及预加载机制说明

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

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、环境说明

SDK分为测试环境(TEST_ENV)及线上环境(RELEASE_ENV)，默认会使用测试环境。初始化前可通过SDK配置项变更请求环境。

建议客户先使用测试环境及测试AppId开发联调，确认接口打通和数据无误后，再切换线上环境和对应线上AppId，进入生产环节。

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、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_FINE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION(**非必须** , 用于地理位置相关)

6、快速集成

SDK库文件导入

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

```
1.  android {
2.      ...
3.      repositories {
4.          flatDir {
5.              dirs 'libs'
6.          }
7.      }
8.  }
9.
10. dependencies {
11.     //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
12.     compile(name: 'sdk-xxx.aar', ext: 'aar')
13. }
```

3、添加Manifest权限

```
1.  <uses-permission
2.      android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
3.  <uses-permission
4.      android:name="android.permission.READ_EXTERNAL_STORAGE"/>
5.  <uses-permission
6.      android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
7.  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
8.  <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
9.  <uses-permission
10.     android:name="android.permission.ACCESS_NETWORK_STATE"/>
11.  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

```

8.     <uses-permission android:name="android.permission.INTERNET"/>
9.     <uses-permission
10.        android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
11.     <uses-permission
12.        android:name="android.permission.ACCESS_COARSE_LOCATION"/>
13.     <uses-permission
14.        android:name="android.permission.ACCESS_FINE_LOCATION"/>

```

4、注册SDK内部组件

```

1.     //此处添加provider, 是为兼容7.0之后的应用自动安装问题
2.     <provider
3.         android:name="android.support.v4.content.FileProvider"
4.         android:authorities="${applicationId}.fileprovider"
5.         android:exported="false"
6.         android:grantUriPermissions="true">
7.         <meta-data
8.             android:name="android.support.FILE_PROVIDER_PATHS"
9.             android:resource="@xml/file_paths"/>
10.    </provider>

```

5、将SDK文件目录下，res文件下xml目录，复制到工程的相应目录下,修改其中的包名，如下：

```

1.     //注意下方的path取值，需要填写用户自己的包名
2.     <external-path path="Android/data/com.xxx.xxx.xxx/"
3.         name="files_root" />
4.     <external-path path="cache/apk/." name="external_storage_root" />

```

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

```

1.     <uses-permission
2.        android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
3.     <uses-permission
4.        android:name="android.permission.READ_EXTERNAL_STORAGE"/>

```

```

3.     <uses-permission
4.         android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
5.     <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
6.     <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
7.     <uses-permission
8.         android:name="android.permission.ACCESS_NETWORK_STATE"/>
9.     <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
10.    <uses-permission android:name="android.permission.INTERNET"/>
11.    <uses-permission
12.        android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION"/>
13.    <uses-permission
14.        android:name="android.permission.ACCESS_COARSE_LOCATION"/>
15.    <uses-permission
16.        android:name="android.permission.ACCESS_FINE_LOCATION"/>

```

3、注册SDK内部组件

```

1.     //奖励视频展示Activity
2.     <activity
3.         android:name="com.ksc.ad.sdk.ui.AdProxyActivity"
4.         android:hardwareAccelerated="true"
5.         android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
6.         android:configChanges="keyboardHidden|orientation|screenSize" />
7.     //动态权限申请，透明浮层Activity
8.     <activity
9.         android:name="com.ksc.ad.sdk.ui.AdPermissionProxyActivity"
10.        android:configChanges="keyboardHidden|orientation|screenSize"
11.
12.        android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"
13.    />
14.    <service android:name="com.ksc.ad.sdk.service.AdProxyService" />
15.    //此处添加provider，是为兼容7.0之后的应用自动安装问题
16.    <provider
17.        android:name="android.support.v4.content.FileProvider"
18.        android:authorities="${applicationId}.fileprovider"
19.        android:exported="false"
20.        android:grantUriPermissions="true">
21.        <meta-data
22.            android:name="android.support.FILE_PROVIDER_PATHS"
23.            android:resource="@xml/file_paths"/>
24.    </provider>

```

4、将SDK文件目录下，res文件下xml目录，复制到工程的相应目录下,修改其中的包名，如

下：

```
1. //注意下方的path取值，需要填写用户自己的包名
2. <external-path path="Android/data/com.xxx.xxx.xxx/"
3.     name="files_root" />
4. <external-path path="cache/apk/." name="external_storage_root" />
```

5、在project视图中选中app module < 右键 < open module setting, 在接下来的project structure视图中，左下角选中app module < dependency选项卡 < 点击左下角加号 < 选择jar dependency < 选择刚导入的jar包

6、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、快速使用

7.1、初始化及预加载

建议在App启动后，第一个页面onCreate时进行SDK初始化及预加载操作。

如果没有调用KsyunAdSdkConfig的setSdkEnvironment()方法，设置SDK请求环境，默认则为测试环境。

```
1. class MainActivity extends Activity {
2.
3.     @Override
4.     protected void onCreate(Bundle savedInstanceState) {
5.         super.onCreate(savedInstanceState);
6.
7.         KsyunAdSdkConfig config = new KsyunAdSdkConfig();
8.         //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，默认则为测试环境
9.         config.setSdkEnvironment(KsyunAdSdkConfig.RELEASE_ENV);
10.        KsyunAdSdk.getInstance().init(MainActivity.this, "your_release_app_id", config, new IKsyunAdInitResultListener() {
11.            @Override
12.            public void onSuccess(Map<String, String> map) {
13.                //SDK初始化成功后，建议尽早调用preloadAd接口，进行预加载操作。
14.                //此接口会预加载当前AppId对应的所有广告位的广告资源
15.                KsyunAdSdk.getInstance().preloadAd(new
IKsyunAdPreloadListener() {
```

```

16.         @Override
17.         public void onAdInfoSuccess() {
18.             //加载广告配置信息成功
19.         }
20.
21.         @Override
22.         public void onAdInfoFailed(final int errCode, final
String errMsg) {
23.             //加载广告配置信息失败
24.         }
25.
26.         @Override
27.         public void onAdLoaded(final String adSlotId) {
28.             //该方法根据加载到的广告数量及广告位数量, 可能会被多次调
用
29.         }
30.
31.         @Override
32.         public void onFailure(int errCode, String errMsg) {
33.             //SDK初始化失败处理
34.         }
35.     });
36. }
37.
38. }

```

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。

```

1.         //判断某广告位是否存在广告
2.         boolean isExist = KsyunAdSdk.getInstance().hasAd(adslot_id);
3.         //hasLocalAd判断某广告位是否存在本地已加载广告
4.         //boolean isExist =
KsyunAdSdk.getInstance().hasLocalAd(adslot_id);
5.         if(isExist){
6.             //广告存在, 调用showAd接口
7.         }else{
8.             //广告不存在, 需要调用preloadAd单个广告位接口进行预加载
9.         }

```

8、高级用法

8.1、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。

```
1.         KsyunAdSdkConfig config = new KsyunAdSdkConfig();
2.         //设置SDK请求环境为线上环境。SDK如果不设置config，默认则为测试环境
3.         config.setSdkEnvironment(KsyunAdSdkConfig.RELEASE_ENV);
4.         //设置奖励视频展示过程中，允许出现关闭按钮
5.         config.setShowCloseBtnOfRewardVideo(true);
6.         //设置奖励视频展示过程中，出现关闭按钮的时间点
7.         config.setCloseBtnComingTimeOfRewardVideo(5);
8.
9.         KsyunAdSdk.getInstance().init(MainActivity.this, appId, config,
new IKsyunAdInitResultListener() {
10.             @Override
11.             public void onSuccess(Map<String, String> map) {
12.
13.             }
14.
15.             @Override
16.             public void onFailure(int errCode, String errMsg) {
17.
18.             }
19.         });
```

8.2、广告事件监听

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

```
1.     public interface IKsyunAdListener {
2.         //广告展示成功时回调
3.         void onShowSuccess(String adSlotId);
4.         //广告展示失败时回调
5.         void onShowFailed(String adSlotId, int errCode, String errMsg);
6.         //广告内容播放完成，一般用于视频类广告
7.         void onADComplete(String adSlotId);
8.         //广告被点击
9.         void onADClick(String adSlotId);
10.        //广告被关闭
11.        void onADClose(String adSlotId);
```



```
12.     }
```

对于奖励视频类型的广告，通过设置setRewardVideoAdListener接口，可以监听奖励条件是否达成回调。

```
1.     public interface IKsyunRewardVideoAdListener {
2.         //奖励条件达成
3.         void onAdAwardSuccess(String adSlotId);
4.
5.         //奖励条件未达成
6.         void onAdAwardFailed(String adSlotId, int errCode, String errMsg);
7.     }
```

8.3、广告资源预加载事件监听

通过设置preloadAd(IKsyunAdPreloadListener preloadListener)接口的回调参数preloadListener，可以监听广告资源预加载相应的事件

```
1.     public interface IKsyunAdPreloadListener {
2.         //预加载广告配置获取成功，注意此回调仅仅是广告配置获取成功，不代表广告视频资源
           下载完成
3.         void onAdInfoSuccess();
4.
5.         //预加载广告配置获取失败
6.         void onAdInfoFailed(int errCode, String errMsg);
7.
8.         //预加载广告视频资源下载完成，参数为广告位Id
9.         void onAdLoaded(String adSlotId);
10.    }
```

8.4、关于视频广告播放及预加载机制说明

- SDK播放视频广告，支持播放本地预加载广告与在线播放广告两种形式，默认会优先使用本地已预加载的广告内容
- 如果客户只想播放本地已缓存好的视频，请在调用ShowAd方法前，使用hasLocalAd方法进行判断，此方法只有对应广告位存在已缓存好的视频时才会返回true。
- 一般情况下，客户只需要进入App初始化成功后，调用一次preloadAd方法出发预加载行为即可。后续的预加载行为，SDK内部会自己执行，客户只需监听对应preloadAd方法设置的回调参数结果即可

。

播放在线内容时，优先

SDK初始化成功后，建议尽早调用preloadAd接口，进行预加载操作，以保证播放奖励视频的体验。

后续SDK内部会在合适的时机触发预加载广告操作，无需用户再调用preloadAd接口。