金山云移动广告Android SDK快速接入文档 V4.0.2

更新日志

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、环境说明

SDK分为沙盒环境(SANDBOX_ENV)及线上环境(RELEASE_ENV)，默认会使用沙盒环境。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_COARSE_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

```
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"/>
```

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/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,修改其中的包名，如下：

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

```
<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"/>
```

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/file_paths"/>
</provider>
```

4、将SDK文件目录下，jar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：

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

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、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初始化成功后，建议尽早调用preloadAd接口，进行预加载操作。
                //此接口会预加载当前AppId对应的所有广告位的广告资源
                KsyunAdSdk.getInstance().preloadAd(new IKsyunAdPreloadListener() {
                    @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) {
                       //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
            }

            @Override
            public void onFailure(int errCode, String errMsg) {
                //SDK初始化失败处理
            }
        });
    }

}
```

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。

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

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。

```
            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) {

                }
            });
```

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

```
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);
}
```

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。

```
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);
}
```

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

```
public interface IKsyunAdPreloadListener {
    //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
    void onAdInfoSuccess();

    //预加载广告配置获取失败
    void onAdInfoFailed(int errCode, String errMsg);

```

```
        //预加载广告视频资源下载完成，参数为广告位Id
        void onAdLoaded(String adSlotId);
    }
```

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，支持仅播放本地视频(hasLocalAd) & 播放本地+在线广告(hasAd)两种形式
- 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。
- 如果hasLocalAd接口返回false，需要客户调用preloadAd单个广告位广告接口，来进行进行广告预加载操作
- 建议在刚进入游戏场景时，调用preloadAd接口进行预加载操作。即将进入奖励视频入口前，调用hasLocalAd判断是否存在广告，并以此作为展示奖励视频入口依据。