安卓 - 4.6.2

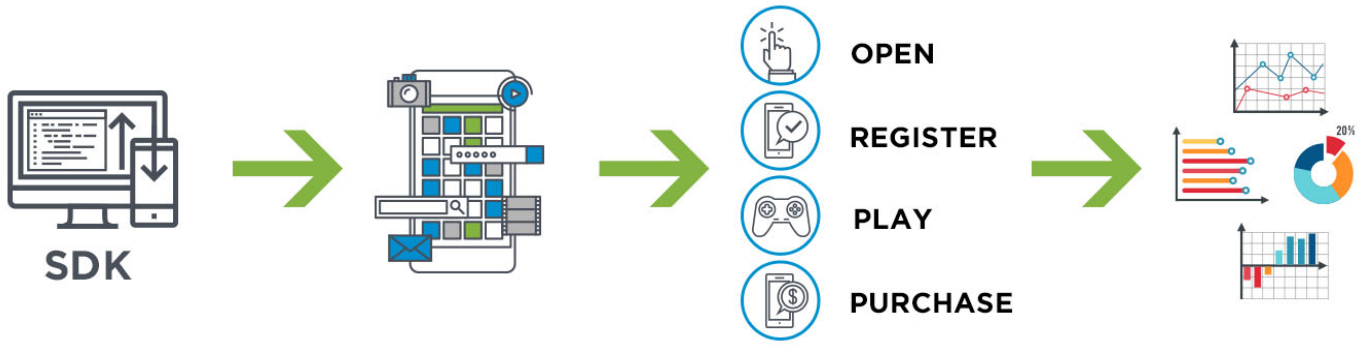
关注 Print / PDF

Jamie Weider
最近更新时间: 今天 05:42PM

页面内容:

1. 注意事项 - 现有最新版本：4.6.2

AppsFlyer的SDK提供应用安装和应用内事件的监控。我们的SDK经过多次迭代，目前已非常稳定，安全，轻量，并且可以很轻松的完成嵌入。



你可以监控安装，应用更新和应用打开（需要完成基本嵌入），也可以进一步监控应用内事件（包括购买，游戏过关等），从而可以评估渠道的ROI以及用户粘性等深层次指标

本指南分为1) 注意事项, 2)本版本更新提示, 3)SDK包下载, 4)AppsFlyer SDK接入强制项目, 5)其他非强制项目接入和6) SDK测试六个部分。**其中前第四和第六为必选接入项**，完成后即可实现对安装的跟踪和留存的计算。其中，第五部分的应用内事件追踪虽为可选项，但强烈推荐接入。

1. 注意事项 - 现有最新版本：4.6.2

1.1 注意：此版本因为涉及的更新较多， 请一定确保认真阅读文章所有内容。如果您需要从3.3.X版本更新至4.3.8版本，需要额外关注此文档。

1.2 如果该应用使用的是Unity进行最后打包，请使用AppsFlyer的Unity Plugin进行对接。

https://support.appsflyer.com/hc/en-us/articles/213766183-Unity

1.3 同时，请注意此版本仅针对安卓2.3版本或者以上的操作系统有效。

1.4 新SDK对设备唯一标识符的抓取有以下特定:

- 请一定保证AppsFlyer至少可以获取一个设备唯一标识符(advertising id, android id or IMEI)以进行归因。

- 使用这个版本的SDK, 如果您接了Google Play Services SDK, KitKat4.4和以上操作系统的设备唯一标识符Android ID和IMEI将默认不会被AppsFlyer SDK自动抓取。开发者仍可以通过以下方法向AppsFlyer上报Android ID和IMEI, 注意这两个方法需要放在AppsFlyerLib.getInstance().startTracking(this.getApplication(),"[Dev_Key]")之前

```
AppsFlyerLib.getInstance().setImeiData("IMEI_DATA_HERE")
AppsFlyerLib.getInstance().setAndroidIdData("ANDROID_ID_DATA_HERE")
```

注意：请一定对应替换"IMEI_DATA_HERE" 和 "ANDROID_ID_DATA_HERE"部分。

- 如果您的应用没有接Google Play Services SDK, 那么IMEI and Android ID默认会被抓取。您也可以通过以下方法禁止Android ID和IMEI被AppsFlyer SDK抓取:

```
AppsFlyerLib.getInstance().setCollectIMEI(false)
AppsFlyerLib.getInstance().setCollectAndroidID(false)
```

2. 此版本较上个版本更新须知

- bug修复

3. AppsFlyer Android SDK下载点击此处

也可以在此处参考接入样本

4. AppsFlyer Android SDK强制接入项目

为成功监测安装和会话，请严格按照以下步骤操作。

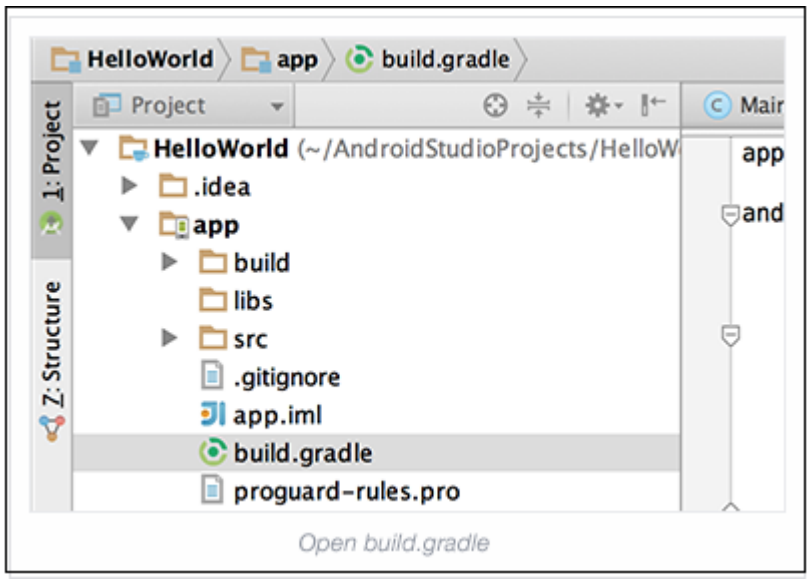
您可以使用Gradle's Dependency Management或者通过SDK.jar手动嵌入AppsFlyer SDK。

4.1 将AppsFlyer SDK放入到您的项目中

嵌入AppsFlyer SDK至您的项目中的最简单的办法为使用Gradle's Dependency Management。

添加AppsFlyer Android SDK Dependency:

1. 打开您的项目（或创建新项目），之后打开 `your_app | build.gradle`



2. 将其添加至 Module-level `/app/build.gradle` （dependencies**之前**）：

```
repositories {
        mavenCentral()
}
```

3. 添加带有AppsFlyer最新版本的SDK的compile dependency至 `build.gradle`文件：

```
dependencies {
   compile 'com.appsflyer:af-android-sdk:4+@aar'
}
```

版本信息可在此处查看。

现在您可以在您的项目中输出AppsFlyer SDK并按照指南进行嵌入。

```
import com.appsflyer.AppsFlyerLib
```

如果您不使用Gradle，请下载并添加AF-Android-SDK.jar 至项目class path。

4.2 在AndroidManifest.xml中设置以下权限

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

注意：其中READ_PHONE_STATE permission为可选项.

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

添加该权限以激活IMEI追踪(如果现在接入的安卓包是针对除Google Play以外的其他应用商店，那么此权限一定需要声明)

4.3 在AndroidManifest.xml中，设置AppsFlyer Install Referrer监听器

针对安卓应用, 同一个intent-filter内不能够同时放置多个监听器，以下为两种放置监听器的方式。

4.3.1 使用多个监听器

为了确保所有Install Referrer监听器可以成功监听由系统播放的referrer参数，请一定在AndroidManifest.xml中将AppsFlyer的监听器置于所有同类监听器第一位，并保证receiver tag在application tag中。

- 如果AppsFlyer是唯一需要的Install Referrer监听器，那么
-

```
<receiver android:name="com.appsflyer.MultipleInstallBroadcastReceiver" android:expor
    <intent-filter>
        <action android:name="com.android.vending.INSTALL_REFERRER" />
    </intent-filter>
</receiver>
```

- 如果需要使用多个Install Referrer监听器，Manifest.xml必须按照以下方式出现:

```
<!—The AppsFlyer Install Receiver is first and will broadcast to all receivers placed be
<receiver android:name="com.appsflyer.MultipleInstallBroadcastReceiver" android:exported=
   <intent-filter>
       <action android:name="com.android.vending.INSTALL_REFERRER" />
   </intent-filter>
</receiver>
<!—All other receivers should follow right after -->
<receiver android:name="com.google.android.apps.analytics.AnalyticsReceiver" android:expo
  <intent-filter>
       <action android:name="com.android.vending.INSTALL_REFERRER" />
  </intent-filter>
</receiver>
<receiver android:name="com.admob.android.ads.analytics.InstallReceiver" android:exported
        <intent-filter>
            <action android:name="com.android.vending.INSTALL_REFERRER" />
```

```
        </intent-filter>
</receiver>
```

注意：上文中的com.google.android.apps.analytics.AnalyticsReceiver和 com.admob.android.ads.analytics.InstallReceiver只是样例

4.3.2 使用单监听器

在AndroidManifest.xml中，请将以下监听器作为INSTALL_REFERRER中第一个监听器，或在其他多监听器后，并确保监听器在application tag中：

```
<receiver android:name="com.appsflyer.SingleInstallBroadcastReceiver"
android:exported="true">

    <intent-filter>

      <action android:name="com.android.vending.INSTALL_REFERRER" />

    </intent-filter>

</receiver>
```

如需了解更多关于如何在您的应用context内设置多个Install Referrer监听器，可参考此文档。

4.4 接入Google Play Services SDK - 此步骤是为了确保AppsFlyer可以获取设备的广告ID(Advertising ID)以进行归因，请一定完成。

- 从此处下载Google Play Services SDK, 而AppsFlyer的追踪实际所需要的只是mobile ads包；
- 在接入此SDK的过程中，推荐您使用Proguard可以拿掉不必要的部分，以达到有效控制包大小的效果；
- 将以下条目添加到 AndroidManifest.xml,作为应用的最后一项 (在</application>前面):

```
<meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version" />
```

注意：

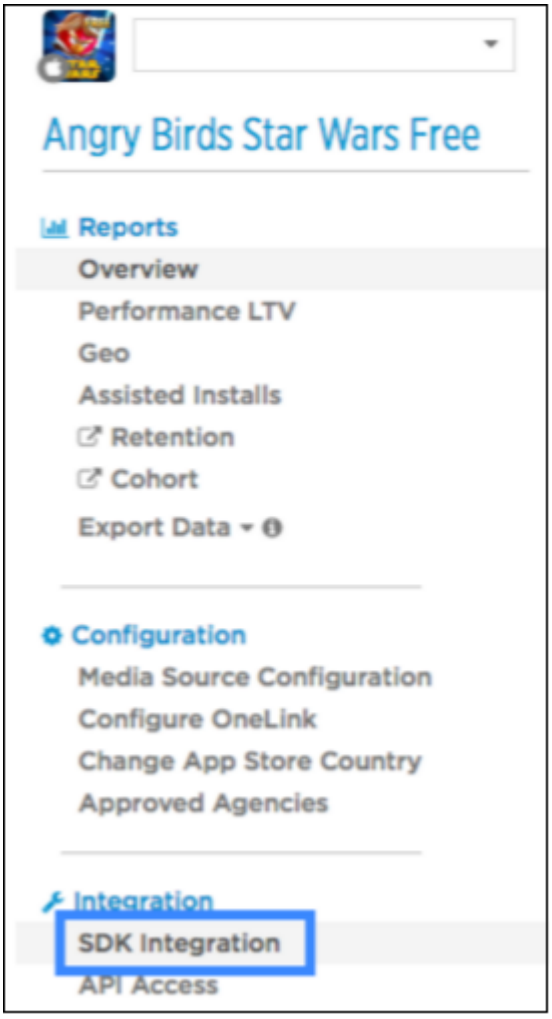
- AppsFlyer使用Google Play Services library 以获取Google Advertising ID。
- Google Play Service 7.5为使用GCM Registration Token(用于卸载监测)的最低版本要求。AppsFlyer建议您使用最新版本。

其他相关资源：https://developers.google.com/android/guides/setup

来源：https://developer.android.com/google/play-services/setup.html

4.5 SDK初始化 - 是AppsFlyer能追踪到激活和应用会话数据的基础

- 进入任何一个应用的AppsFlyer数据后台>> SDK Integration获取您账户的Dev_Key备用



- SDK的初始化，需要将以下项目添加应用加载的第一个Activity/onCreate函数中, 并将 [Dev_Key] 替换成您AppsFlyer账户的实际Dev_Key

```
AppsFlyerLib.getInstance().startTracking(this.getApplication(),"[Dev_Key]");
```

该API能够是的AppsFlyer监测到安装、会话和更新。

注意:

- *如果您的应用有re-trageting活动或者含有多个activity，并支持deeplinking，请按照5.7进行操作，以保证监测到回话和其他应用内活动。*
- *如果您的应用为后台运行的工具类应用，请按照5.12进行操作，以保证使用AppsFlyer监测会话和留存。*

5. 其他非强制项目接入 - 请与投放团队协商哪些部分需要接入

5.1 应用内事件追踪接入(In-App Events Tracking API) - 建议接入

之前的强制步骤仅仅只满足了对应用激活和开启的追踪, 而应用内事件是安装之后用户与应用交互的行为（如注册，登录，付费等），对它的追踪可以更好的判断用户质量（包括反作弊），从而可以对广告投放进行优化。

5.1.1 语法：

```
public static void trackEvent(Context context, String eventName, Map eventValues)
```

- context - 使用 getApplicationContext()

- eventName： 字符串格式，定义了事件的名称，此名称也可以自定义。事件的命名字符个数需要限制在45个以内，如果长于45个字符那么在AppsFlyer后台将无法显示，只有在对应csv报告中或者数据接口中能正常显示。

- eventValues：对应MAP格式的事件赋值(即富应用内事件格式), 一个事件可由多个key进行赋值。
- 如果事件需要统计收益(revenue), 那么请务必使用af_revenue作为指定的key(下方)上报对应的值。 格式上，可使用任何numeric类型进行赋值。

AFInAppEventParameterName.REVENUE

您可以将任何数值作为收益，正负均可。

注意：而使用af_price统计的值不会作为收益(Revenue)汇总在AppsFlyer后台。*
AFInAppEventParameterName.PRICE

5.1.2 应用内事件追踪对接案例 - 更多关于富应用内事件追踪接入，请参考此文档。

请注意: AppsFlyer所有的预设参数库除了在追踪收益时必须使用af_revenue, 其他都是可选，而且可以自定义。因此，下方案例仅作为参考。*

Example 1: 过关事件 - Level Achieved In-App Event

```
Map<String, Object> eventValue = new HashMap<String, Object>();
eventValue.put(AFInAppEventParameterName.LEVEL,9);
eventValue.put(AFInAppEventParameterName.SCORE,100);
AppsFlyerLib.getInstance().trackEvent(content,AFInAppEventType.LEVEL_ACHIEVED,eventValue)
```

对应的事件会以下面的数据格式上报给AppsFlyer

{af_level: 9, af_score: 100}

Example 2: 购买事件 - Purchase Event

```
Map<String, Object> eventValue = new HashMap<String, Object>();
eventValue.put(AFInAppEventParameterName.REVENUE,200);
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE,"category_a");
eventValue.put(AFInAppEventParameterName.CONTENT_ID,"1234567");
eventValue.put(AFInAppEventParameterName.CURRENCY,"USD");
AppsFlyerLib.getInstance().trackEvent(content,AFInAppEventType.PURCHASE,eventValue);
```

对应的事件会以下面的数据格式上报给AppsFlyer , 同时$200这笔金额因为正确使用了REVENUE这个key，会被AppsFlyer作为总收益的一部分对应统计在数据后台。

```
{af_content_id: “1234567”, af_content_type: “category_a”, af_revenue: 200, af_currenc
```

注意：应用内事件名称不得超过45个字段。超过的事件将不显示在控制面板中，只会在raw data、Pull和Push API中显示。

5.2 货币单位设置

- 可以通过下方的API,设置全局的货币单元

```
AFInAppEventParameterName.CURRENCY
```

- 也可以在事件层级设置货币单元，对应使用的下方的代码

```
AppsFlyerLib.getInstance().setCurrencyCode("GBP");
```

- USD是默认的货币单元，AppsFlyer接受的是ISO格式，具体可参看此文档。

5.3 获取AppsFlyer Unique ID

AppsFlyer ID是基于AppsFlyer专利技术生成的设备唯一识别符，是AppsFlyer归因和统计的重要依据。如果您需要获取该ID, 可通过以下API实现：

利用以下API获取AppsFlyer Unique ID:

```
public String getAppsFlyerUID(Context context);
```

示例：

```
String appsFlyerId = AppsFlyerLib.getInstance().getAppsFlyerUID(this);
```

5.4 设置Customer User ID (用户账户ID)

有些应用会给每个独立用户指定一个的ID(玩家ID或者登陆邮箱等)作为标识其身份唯一性的标志。可将此ID上报给AppsFlyer，由此账户ID便可以和其他设备ID建立映射关系。

对应API是：

```
AppsFlyerLib.getInstance().setCustomerUserId("myId");
```

注意：

- *该接口的调用时机可根据具体情况自定义，基本原则是确保该ID可以和其他的设备ID有效地形成映射关系。一般建议在应用一打开的时候调用, 或者如果这个账户ID仅仅只和某个事件相关，也可以在该事件对应位置进行调用。*

如果您需要将AppsFlyer追踪的数据回传给类似Mixpanel或者Swrve这类用户行为分析平台, 这个API是必须要接的。

5.5 实时获取转化数据 - 带来了激活转化的媒体源数据

可以在SDK层级实时获取AppsFlyer追踪到的媒体源数据。一个很重要的应用是延迟深度链接，您可以通过分析媒体源的信息，让用户在激活后被直接引导至匹配媒体源的页面而非应用首页。具体设置，请参考此文档。

注意：该功能相关函数会在每次应用打开的时候调用。

5.6 向AppsFlyer上报用户邮箱

您也可以给AppsFlyer上报每个设备对应关联的邮箱，AppsFlyer也接受通过Sha1, MD5或者plain方法加密的邮箱值。

例如：

```
AppsFlyerLib.getInstance().setUserEmails(AppsFlyerProperties.EmailsCryptType.MD5, "email1
```

注意：个人验证信息（PII）如邮箱地址不会被AppsFlyer保存，且不会出现在任何报告中。手机该信息的目的仅为向媒体平台回传所用。

5.7 基于深度链接Re-targeting广告追踪

Deeplinking为re-targeting活动追踪的很重要的部分，如果有re-targeting活动，AppsFlyer强烈建议您使用Deeplinking。

如果您需要通过AppsFlyer, 追踪通过深度链接打开应用的Re-targeting活动投放数据，那么请确保每个相关的activity对应OnCreate()方法中都添加了以下的代码（主活动除外，即SDK初始化处除外）：

```
AppsFlyerLib.getInstance().sendDeepLinkData(this);
```

*通过Deeplinking开启的activity需在manifest文件中含有以下activity定义：

```
<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="your unique scheme" />
</intent-filter>
```

5.8 应用内购买认证

AppsFlyer可为您提供Google Play服务器端的购买认证。请在onActivityResult方法中, 调用validateAndTrackInAppPurchase方法, 此方法的成功调用会自动向AppsFlyer上报一个af_purchase事件。

```
public static void validateAndTrackInAppPurchase(Context context,
String publicKey, String signature, String purchaseData, String price,
String currency, HashMap<String, String> additionalParameters);
```

调用的callback blocks有两种结果, 一个是“Success”另一个是“Failure”(操作失败的原因有多种，常见的包括认证无效)。

```
AppsFlyerLib.getInstance().registerValidatorListener(this,new AppsFlyerInAppPurchaseValid
            public void onValidateInApp() {
                        Log.d(TAG, "Purchase validated successfully");
                }
                public void onValidateInAppFailure(String error) {
                        Log.d(TAG, "onValidateInAppFailure called: " + error);
                }
          });
```

示例

```
AppsFlyerLib.getInstance().validateAndTrackInAppPurchase(context,publicKey, signature, pu
```

5.9 主动禁止对特定设备的追踪

AppsFlyer也提供免除对特定设备追踪的解决方案。这个方法是严格遵守相关数据隐私要求的。默认的设置是关闭，表示所有激活设备都可以被AppsFlyer追踪。

如果您需要使用这个方法，请在SDK初始化时调用：

```
AppsFlyerLib.getInstance().setDeviceTrackingDisabled(true);
```

5.10 追踪除在Google Play上发布的安卓包 (Out of Store Tracking)

Google Play是AppsFlyer后台默认的安卓应用商店，如果您的apk包发布在其他的安卓应用商店，便需要在完成之前的步骤后额外的进行以下的对接。

需要在AndroidManifest.xml 文件中，</application> tag前加入一句以CHANNEL为key的代码，请注意赋值是大小写敏感：

如果该应用商店是Amazon，该代码可写成：

```
<meta-data android:name="CHANNEL" android:value="Amazon" />
```

如果您对该应用商店的命名是Standalone，该代码可写成：

```
<meta-data android:name="CHANNEL" android:value="Standalone"/>
```

Verizon (预安装):

```
<meta-data android:name="CHANNEL" android:value="Verizon" />
```

注意：您在控制面板中添加该应用时，请务必加入CHANNEL值。

对应的追踪链接配置和SDK对接测试，请参考此文档。

5.11 追踪卸载

添加如下权限至manifest,请注意将下列代码的<your-package-name>部分替换成对应的安卓包名。 (使用推送通知的应用需拥有以下权限)

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

      <permission android:name="YOUR-PACKAGE-NAME.permission.C2D_MESSAGE"
            android:protectionLevel="signature" />
      <uses-permission android:name="<your-package-name>.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

在</application>前添加一下接收器:

```
<receiver
      android:name="com.google.android.gms.gcm.GcmReceiver"
      android:exported="true">
```

```
        <intent-filter>
            <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        </intent-filter>
    </receiver>
    <service android:name="com.appsflyer.InstanceIDListener" android:exported="false">
        <intent-filter>
            <action android:name="com.google.android.gms.iid.InstanceID"/>
        </intent-filter>
    </service>
```

在您的main activity中设置 **GCM Project Number**：立刻在调用startTracking()之前添加该行内容:

```
public void setGCMProjectNumber(String id);
```

举例：

```
AppsFlyerLib.getInstance().setGCMProjectNumber('1234567890');
```

AppsFlyer提供卸载数据的追踪方案，具体的对接步骤请参考此文档。

5.12 后台运行会话追踪方法

这个方法多适用于工具类应用对后台运行会话的统计。

方法声明

```
public void reportTrackSession(Context context);
```

方法调用

```
AppsFlyerLib.getInstance().reportTrackSession(context);
```

5.13 通过OneLink定制应用内深度跳转

OneLink原来已支持深度链接，以直达在OneLink内af_dp参数定义的应用内特定页面。

而此处提到的新功能，您需要接入 **onAppOpenAttribution** 回调。您可利用此回调获取OneLink中的参数信息，以自定义用户应用内深度跳转。

```
void onAppOpenAttribution(Map<String,String> attributionData); //android
```

详情，请参考此文档。

5.14 推送通知监测

AppsFlyer可以将推送通知作为re-targeting活动的一部分进行监测。

为使用该功能，请在每个点击通知开启应用的Activity的onCreate中调用一下方法：

AppsFlyerLib.getInstance().sendPushNotificationData(this);

数据负载需包括 "af"及相关key-value字符：

```
\"af\" : { \"c\" : \"test_campaign\" , \"is_retargeting\" : \"true\" , \"pid\" : \"push_p
```

示例:

```
\"data\": { \"score\": \"5x1\", \"time\": \"15:10\" , \"af\" : { \"c\" : \"test_campaign\
```

6. SDK接入测试

只有完成测试后, 才可确保对接无误。安卓SDK测试文档请参考此处。

评论

0 条评论

成为第一个写评论的人。

提交