SDK对接文档 - 安卓 4.3.7

关注 Print / PDF

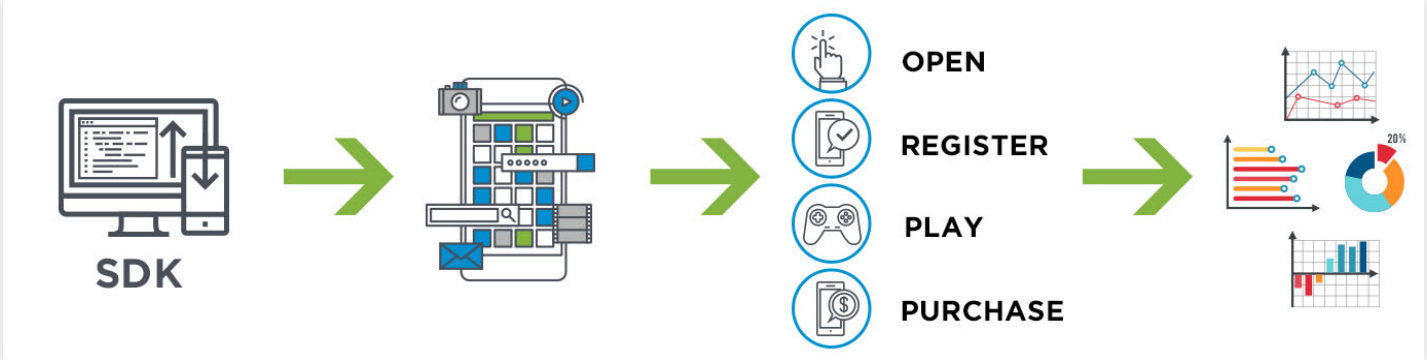
Jamie Weider
最近更新时间: 今天 05:36PM

页面内容:

请注意，这不是最新的SDK对接文档；如需参考最新的文档，请点击此处查找

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)本版本更新提示, 2)SDK包下载, 3)AppsFlyer SDK接入强制项目, 4)其他非强制项目接入和5) SDK测试五个部分。其中第二，第三和第五部分为必选接入项，完成后即可实现对安装的跟踪和留存的计算。其中，第四部分的应用内事件追踪虽为可选项，但强烈推荐接入。

1. 此版本较上个版本更新须知 - 4.3.7

- 修复了bug

2. AppsFlyer Android SDK下载点击此处

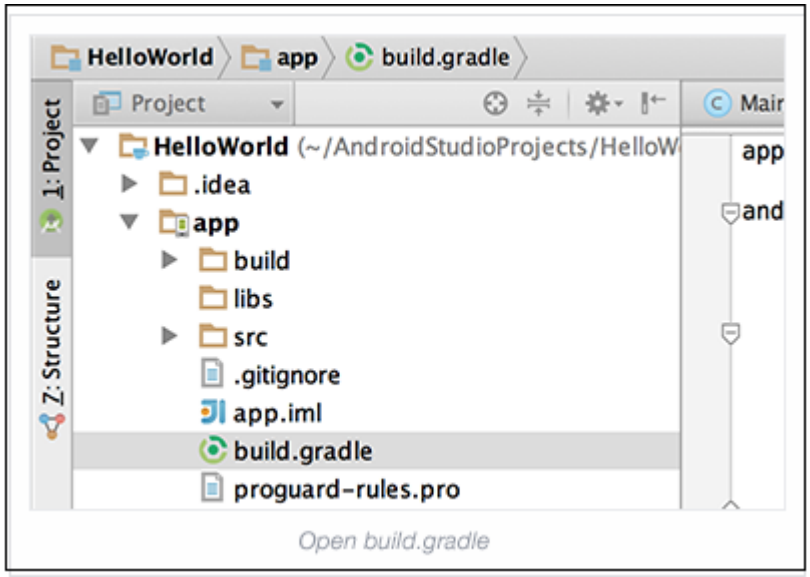
也可以在此处参考接入样本

3. AppsFlyer Android SDK强制接入项目

3.1 将AppsFlyer SDK放入到您的项目中

通过Gradle's Dependency Management将SDK导入是比较便捷的方法，具体方法如下:

- 进入您的安卓项目, 打开对应的build.gradle



- 将下面代码添加到Module-level /app/build.gradle文件中，并放到dependencies模块之前:

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

- 使用最新的AppsFlyer SDK, 在dependencies模块中加入下列代码

```
compile 'com.appsflyer:af-android-sdk:4.3.7@aar'
```

- 进行AppsFlyer SDK导入

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

而如果您不是使用Gradle，可将AppsFlyer SDK jar包拖入到对应路径中。

3.2 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" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

注意：其中READ_PHONE_STATE permission is optional. 而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。

3.3 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，因此为了确保所有Install Referrer监听器可以成功监听由系统播放的referrer参数，请一定将AppsFlyer的监听器置于所有同类监听器第一位。

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

```
<receiver android:name="com.appsflyer.MultipleInstallBroadcastReceiver" android:exported=
   <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，请参照下方案例:

```
<!—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

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。

3.4 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。

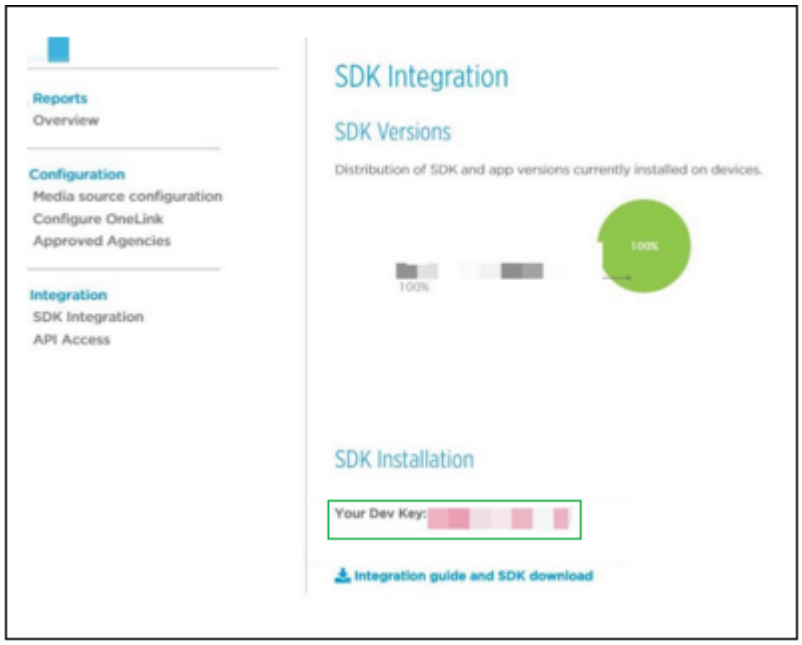
- 从此处下载Google Play Services SDK, 而AppsFlyer的追踪实际所需要的只是mobile ads包;
- 在接入此SDK的过程中，推荐您使用Proguard可以拿掉不必要的部分，以达到有效控制包大小的效果;
- 将以下条目添加到 AndroidManifest.xml,作为应用的最后一项 (在</application>前面):

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

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

3.5 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]");
```

注意: *原来sendTracking()方法都已经在此版本移除。新SDK可以自动识别在前台以及后台运行的应用开启。*

4. 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

4.1 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。

4.1.1 语法:

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

- 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
```

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
```

4.1.2 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
```

4.2 货币单位设置

- **可以通过下方的API,设置全局的货币单元**

- `AppsFlyerLib.getInstance().setCurrencyCode("GBP");`
  也可以在事件层级设置货币单元，对应使用的下方的代码

- `AFInAppEventParameterName.CURRENCY`
  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。

4.3 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：

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

4.4 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。*

4.5 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。

4.6 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
```

4.7 基于深度链接Re-targeting广告追踪

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:

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

4.8 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 AppsFlyerInAppPurchaseValic
            public void onValidateInApp() {
                  Log.d(TAG, "Purchase validated successfully");
            }
            public void onValidateInAppFailure(String error) {
                  Log.d(TAG, "onValidateInAppFailure called: " + error);
            }
      });
```

4.9 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);
```

4.10 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"/>
```

对应的追踪链接配置和SDK对接测试，请参考此文档。

4.11 追踪卸载

因为该卸载监测方法依赖push notifications，需要在Manifest文件中声明以下权限：

```
<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> 关闭标签前，添加一下监听器

```
<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>
```

在应用第一个activity中设置GCM项目编号：将以下代码添加在应用第一个activity的startTracking()方法之后：

```
AppsFlyerLib.getInstance().setGCMProjectID(this,” 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. 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