SDK for Android集成手册举例

该文档只适用于SDK开发参考，可以集成在移动应用中监测相关数据，以下提到的所有事件和自定义事件要求上传的参数都需要根据业务实际需要修改，字段也可以根据开发的实际情况增减。

一、综述

1. 使用用范围

该SDK为移动应用提供数据统计分析服务，通过在移动应用中接入该SDK，可在SDK开发方后台查阅相关数据

此SDK适用Android XX版本 及以上设备。

2. 统计标准

新增用户：

该SDK中的用户指用户的一台唯一设备。

用户的一次使用：

指用户从打开应用的界面至离开界面的完整过程。如用户在离开界面后30分钟内重新回到应用中，将被认为是上次使用被打扰后的延续，记为一次完整使用。

页面：

指Android应用的每个Activity或Fragment

自定义事件：

指用户在应用中进行了特定的操作或是达成了特定的条件。例如：用户点击了广告，用户进行了支付行为等。

自自定义事件用于收集任意您期望跟踪的数据。

二、接入数据系统

1. 为应用申请AppKey

在后台网站中创建一款应用，将获得一串32位的AppKey，用于唯一标识一款应用。

是否支持跨平台应用(统一的App ID)?

2. 向工程中导入SDK

下载数据统计SDK后解压至本地目录，将其中的XXSDK-*.jar导入至工程中； 在Eclipse中（其他开发工工具参阅其自自带说明）右键点击工工程根目录→选择Properties --> Java Build Path -->Libraries → 点击Add External JARs...找到本地目录下的XXSDK-*.jar文件，点击 打开按钮 即导入成功。

3. 配置ANDROIDMANIFEST.XML文文件

3.1 权限配置：

3.2 服务配置：

在<application>标签下添加如下元素：

<service

android:name= “com.test1.businesstrack.core.eventReceiver.EventService">

</service>

3.3 meta-data配置：

权限用用途

INTERNET 允许程序联网和发送统计数据的权限。

ACCESS_NETWORK_STATE 允许应用检测网络连接状态，在网络异常状态下避免数据发送，节省流量和电量。

READ_PHONE_STATE 允许应用以只读的方式访问手机设备的信息，通过获取的信息来唯一标识用户。

ACCESS_WIFI_STATE 获取设备的MAC地址，同样用来标识唯一用户。

示例代码：

```
<?xml version="1.0" encoding="utf-8"?>

<manifest ......>

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

<uses-permission

android:name="android.permission.ACCESS_NETWORK_STATE"/>

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

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

<application ......>

<activity ......>

......

</activity>

......

<service

android:name= “com.test.businesstrack.core.eventReceiver.EventService” >

</service>

<meta-data android:name="XX_appkey" android:value="Your_app_id"/>

<meta-data android:name="xx_channelid" android:value= “Your_channel_id"/>

<meta-data android:name="xx_account_id" android:value="Your_account_id"/>
```

</application>

</manifest>

4. 添加代码

4.1 添加SDK初始化代码

在应用启动的时候（比如在启动应用的activity的onCreate方法中，或者在自定义的

application的

onCreate方方法中，越早越好）调用以下方法：

如果没有在AndroidManifest里里的meta-data声明XX_appkey和XX_channelid，请调用

用:

XXAppAnalytics.init(final Context context, final String appKey, final String

channelId)

//如果已经在AndroidManifest里里的meta-data申明了XX_appkey和XX_channelid，请

调用用:

XXAppAnalytics.init(final Context context)

4.2 添加必要的统计方方法

4.2.1 页面统计方方法

使用数据统计系统需要至少添加以下调用方法，这些调用用于准确跟踪用户每次的应用使用，

是准确统计启动、活跃、留存数据的基础：

在应用用的每个Activity的onResume方法里调用XXAppAnalytics.onResume(Activity

context)，

传入的参数为当前Activity实例，用于跟踪用户使用中的打开应用和页面跳转的数据。

@Override

protect void onResume() {

meta-data 用用途

XX_appkey

用来唯一标识应用，将创建应用时获得的32个字符组成的AppKey写meta-data

中，替换value中的“Your_app_id”即可。

XX_channelid

用来标注应用推广渠道，区分用户的来源来查看统计。可自定义渠道名来替换value中的

“Your_channel_id”。 格式：32个字符内，支持中文、英文、数字、空格、英文点和下

划线，请勿使用其他符号。

XXzamplus_account_id 用来标注客户的字段。

super.onResume();

XXAppAnalytics.onResume(this); }

在应用的每个Activity中的onPause方法里调用XXAppAnalytics.onPause(Activity

context)，传入的参数为当前的Activity实例，用于跟踪用户离开页面和退出应用的数据。

@Override

protect void onPause() {

super.onPause();

XXAppAnalytics.onPause(this); }

注意1： 确保在每个activity中都调用了以上方法，否则可能造成应用的使用时长、页面访

问时长数据错误。

注意2：对有继承关系的Activity，只需要对上层Activity基类添加调用即可，否则可能会重

复计量，造成数据错误。

注意3：对于会内嵌其他Activity的ActivityGroup(比如TabActivity)，建议不要在ActivityGroup里添加调用，而是在所有内嵌的Activity里里添加。

4.2.2 子页面统计方法

随着Android平台引入Fragment的支持，部分应用需要改用Fragment(或者类似的技术)而不是Activity来实现应用的页面。在这种情况下，不能利用XXAppAnalytics.onResume/onPause方法来跟踪页面。为此引入了两个新的API，用于跟踪不是用Activity来实现的页面。不过需要注意的是，

页面应该是一个接着一个的，进入新页面，必须离开旧页面，否则可能导致展示的数据异常。

在进入页面的时候加入如下的代码调用：

XXAppAnalytics.onPageStart(Context context, String pageName);

其中pageName为自定义的页面名称。

在离开页面的时候加入如下的代码调用：

XXAppAnalytics.onPageEnd(Context context, String pageName);

其中pageName为自定义的页面名称，需要和onPageStart中传入的一致；并且onPageStart和onPageEnd必须成对调用。

4.2.3 自定义事件统计

自定义事件用于统计任何您期望去跟踪的数据，如：点击某功能按钮、填写某个输入框、触发了某个广告等；同时，自定义事件还支持添加一些描述性的属性参数，使用多对Key-Value的方式来进行发送（非必须使用），用来对事件发生时的状况做详尽分析。

一些事件会具有层级关系，如果给每个都命名不同的Event ID，会使ID总数量过多，带来管理和查阅数据的麻烦，我们提供了Label标签的用法。如果一系列事件都属于同一类，可使用相同的Event ID，而选用不同的Label即可。这个用法类似于Event ID是大目录，Label

是具体事件内容。

同时 ,自定义事件还支持添加一些描述性的属性参数 ,可使用Key-Value的方式来进行发送。

通过调用以下方法发送自定义事件：

XXAppAnalytics.onEvent(Context context,

String eventId, String eventLabel,

HashMap<String, String> params);

如：记录一个商品列表页面上某个商品被点击进入详情页的事件：

HashMap<String

,

String>

params

=

new

HashMap<String

,

String>();

params.put(“商品名”,

“ELIXIER 樱花睡梦面膜”);

params.put(“商品价格”,

“248”);

XXAppAnalytics.onEvent(context,

“商品在列表中被选中”,

“搜索结果列表”，

params);

4.2.4 常用事件统计

对于一些常用的事件，SDK提供了一些快捷方法来简化了调用。这些方法中的isConversion

参数用于

定义是否传递转化代码，如果是true，则会将此事件作为转化事件记录。

例如用户注册事件：

XXAppAnalytics.onRegister(Context

context,

String

userId,

boolean

isConversion);

第二个参数为注册的账号。

用户登录事件：

XXAppAnalytics.onLogin(Context

context,

String

userId,

boolean

isConversion);

第二个参数为登录的账号。

用户将商品加入到购物车：

XXAppAnalytics.onCart(Context

context,

String

produectId,

int

amount,

boolean

isConversion);

第二个参数是商品的ID，第三个参数是对应商品放入购物车的数量。

用户确定购物车，进入订单创建流程：

XXAppAnalytics.onCartSubmit(Context

context,

String

userId,

boolean

isConversion)；

第二个参数为用用户的账号。

4.2.5 地理位置信息统计

如果您的应用会使用用户的位置信息，可通过接口将信息提交至SDK数据中，可使您获得更

加精准的数据报表。调用以下方法来提交您获取的精确信息：

XXAppAnalytics.onGeo(Context context, double longitude, double latitude, String

coor);

第二,三个参数是经纬度数据,第四个参数是坐标系标识，可传“WGS84”，“GCJ02”，“BD09”等。如使

用系统自带的定位api或不清楚具体坐标系，可以传空。

4.2.6

应用异常捕获

收集应用的错误日志可帮助您来修正BUG、改善产品。报表中，我们除了提供错误次数的数据外，还提供错误的详细信息阅览，并会对错误进行合理分类。

您可通过两种方式来获取错误日志：让

SDK自动捕获异常、或您主动传送错误信息给SDK。

SDK自动获取异常信息

为了简化开发者的工作，我们提供自动获取异常信息的功能。如果你的应用中没有配置自动捕捉异常信息，可以在应用启动时的SDK初始化方法后调用以下方法并传入true，来开启自自动捕获异常：

XXAppAnalytics.init(getApplicationContext());

XXAppAnalytics,setReportUncaughtExceptions(Context context, boolean enabled);

主动传送异常信息

您也可主动调用以下方法将抛出的exception信息传入至方法中的第二个参数中：

XXAppAnalytics.onError(Context context, Throwable throwable);

示例：

```
try {
File file = new File("filePath");
```

```
if(!file.exists()){
file.createNewFile();
}
} catch (IOException e) {
XXAppAnalytics.onError(this, e);
}
```

4.2.7 营销效果追踪

如果你需要跟踪一些营销效果的相关数据，可以添加以下方法。

营销相关事件的追踪：

如果你需要追踪一个用户在应用中的某个营销活动中的行为，可以添加以下方法：

```
XXAppAnalytics.onRemarketingEvent(Context context, String pageName,
HashMap<String,
String> map);
```

第二个参数pageName是当前事件发生时所在的页面名称。

第三个参数是与此事件相关的自定义参数，比如优惠的商品名，优惠价格，优惠活动等。

如果你需要追踪一个用户在应用中的一个转化事件（如下单或支付等行为），你可以添加以下方法：

```
XXAppAnalytics.onConversion(Context context, String conversionId,
HashMap<String, String>
map);
```

第二个参数是此事件对应的转化类型。

第三个参数是与此转化事件相关的自定义参数，如订单号，支付金额等。

4.2.8 日志输出控制

在SDK集成调试时，可以开启SDK的日志输出方便查看运行情况：

XXAppAnalytics.setLogStatus(true);

XXAppAnalytics.init(getApplicationContext());

在初始化前设置日志开关。

5. 关于混淆

请在混淆配置文件proguard.cfg或proguard-project.txt的最后加上

-keep public class com.XX.businesstrack.**

-keepclassmembers class com.XX.businesstrack.** { public protected *;}

使得混淆的时候不会影响统计 SDK 的命名空间。__