OpenStatistics 集成指南

1.集成准备

1.1 下载

下载 OpenStatistics 开源统计 SDK

1.2 导入 SDK

将下载包中的 libs 文件夹合并到本地工程 libs 子目录下；在 Eclipse 中右键工程根目录，选择 Properties -> Java Build Path -> Libraries，然后点击 Add External JARs... 选择指向 jar 的路径，点击 OK，即导入成功。(ADT17及以上不需要手动导入)

1.3 获得 Appkey

在 ShareSDKStatistics 开源统计项目的管理后台中创建 App 并获得 AppKey

2.基本功能集成

2.1 配置 manifest

```
<manifest……>

<uses-sdk android:minSdkVersion="4"></uses-sdk>

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

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

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

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

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

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

<uses-permission
android:name='android.permission.WRITE_EXTERNAL_STORAGE'/>

<application ……>

……

<activity ……/>

<meta-data android:name="SHARESDK_APPKEY" android:value="YOUR_APP_KEY"/>

<meta-data android:name="SHARESDK_CHANNEL" android:value="Channel ID"/>

<!-- service 用于保证一定能够向服务器上传数据 -->

<service android:name="cn.sharesdk.open.statistics.server.RemoteService"

    android:process=":remote">
    <intent-filter>
    <action android:name="cn.sharesdk.open.statistics.server.AIDLService"/>
    </intent-filter>
</service>

</application>

</manifest>
```

2.1.1 权限获取

权限	用途
ACCESS_NETWORK_STATE(必须)	检测联网方式，区分用户设备使用的是2G、3G 或是WiFi
READ_PHONE_STATE(必须)	获取用户手机的 IMEI，用来唯一的标识用户。(如果您的应用会运行在无法读取 IMEI 的平板上，我们会将 mac 地址作为用户的唯一标识，请添加权限：android.permission.ACCESS_WIFI_STATE)
INTERNET(必须)	允许应用程序联网，以便向我们的服务器端发送数据。
WRITE_EXTERNAL_STORAGE(必须)	程序的缓存 log 保存在 sdcard 中
android.permission.GET_TASKS（必须）	获取当前 activity 的类名、报错的类名

2.1.2 appkey 填写

将 <meta-data android:name="SHARESDK_APPKEY" android:value="YOUR_APP_KEY"/> 中的 YOUR_APP_KEY 替换为您在 ShareSDKStatistics 开源统计项目后台申请的应用 appkey。

不在 manifest 里配置 ShareSDKStatistics 开源统计项目的 appkey，而是在每个 Activity 中配置，可以通过此接口实现：MobclickAgent.onResume(this, MapStatic.YOU_MENG_APPK, MapStatic.ChannelId);

2.1.3 channel 填写

将<meta-data android:name="SHARESDK_CHANNEL" android:value="Channel ID"/> 中的 Channel ID 替换为您应用的推广渠道。
您可以使用20位以内的英文和数字为渠道定名，在您查看数据时，渠道会作为一个数据细分的维度。

2.2 session 的统计

正确集成如下代码，才能够保证获取正确的新增用户、活跃用户、启动次数、使用时长等基本数据。

在每个 Activity 的 onResume 方法中调用 MobclickAgent.onResume(Context), onPause 方法中调用 MobclickAgent.onPause(Context)

```
public void onResume() {

        super.onResume();

        MobclickAgent.onResume(this);

}

public void onPause() {

        super.onPause();

        MobclickAgent.onPause(this);

}
```

- 确保在所有的 Activity 中都调用 MobclickAgent.onResume() 和 MobclickAgent.onPause()方法，这两个调用将不会阻塞应用程序的主线程，也不会影响应用程序的性能。
- 注意如果您的 Activity 之间有继承或者控制关系请不要同时在父和子 Activity 中重复添加 onPause 和 onResume 方法，否则会造成重复统计(如使用 TabHost、TabActivity、ActivityGroup 时)。
- 当用户两次使用之间间隔超过30秒时，将被认为是两个的独立的 session(启动)，例如用户回到 home，或进入其他程序，经过一段时间后再返回之前的应用。可通过接口：MobclickAgent.setSessionContinueMillis(long interval) 来设置这个间隔。

2.3 页面的统计

页面统计集成正确，才能够获取正确的页面访问路径、访问深度（PV）的数据。

2.3.1 只由 Activity 构成的应用

如果您已完成上述步骤，那么 SDK 已默认统计了每个 Activity 的跳转路径。页面统计不需要再添加其他代码。

2.3.2 包含 Activity、Fragment 或 View 的应用

统计程序中包含 Fragment 的情况比较复杂，首先要明确一些概念。

1 MobclickAgent.onResume()和 MobclickAgent.onPause() 方法是用来统计应用时长的(也就是 Session 时长,当然还包括一些其他功能)
2 MobclickAgent.onPageStart()和 MobclickAgent.onPageEnd() 方法是用来统计页面跳转的

在仅有 Activity 的程序中，SDK 自动帮助开发者调用了 2. 中的方法，并把 Activity 类名作为页面名称统计。但是在包含 fragment 的程序 中我们希望统计更详细的页面,所以需要自己调用方法做更详细的统计。然后需要做两步集成：

3 使用 onResume 和 onPause 方法统计时长, 这和基本统计中的情况一样(针对 Activity)
4 使用 onPageStart 和 onPageEnd 方法统计页面(针对页面,页面可能是 Activity 也可能是 Fragment 或 View)

SDK在统计Fragment时，需要关闭Activity自带的页面统计，然后在每个页面中重新集成页面统计的代码(包括调用了 onResume 和 onPause 的Activity)。

```
MobclickAgent.openActivityDurationTrack(false);
```

对于一些典型，比如页面是直接放在 Activity 里面的，统计代码大约是这样 ：

```
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
MobclickAgent.openActivityDurationTrack(false);
.....XXXXXXX.....
}
public void onResume() {
    super.onResume();
    MobclickAgent.onPageStart("SplashScreen"); //统计页面
    MobclickAgent.onResume(this);          //统计时长
}
public void onPause() {
    super.onPause();
    // 保证 onPageEnd 在 onPause 之前调用,因为 onPause 中会保存信息
    MobclickAgent.onPageEnd("SplashScreen");
    MobclickAgent.onPause(this);
}
```

如果页面是使用 FragmentActivity + Fragment 实现的，需要在 FragmentActivity 中统计时长：

```
public void onResume() {
    super.onResume();
    MobclickAgent.onResume(this);       //统计时长
}
```

```
public void onPause() {
    super.onPause();
    MobclickAgent.onPause(this);
}
```

并在其包含的 Fragment 中统计页面：

```
public void onResume() {
    super.onResume();
    MobclickAgent.onPageStart("MainScreen"); //统计页面
}
public void onPause() {
    super.onPause();
    MobclickAgent.onPageEnd("MainScreen");
}
```

需要注意的是这些方法的调用，需要保证线性不交叉，每个 start 都有一个 end 配对，如下：

onPageStart ->onPageEnd-> onPageStart -> onPageEnd -> onPageStart ->onPageEnd

这样才能保证每个页面统计的正确，关于页面统计 API 的最佳实践说明见 Fragment 页面统计 。

2.4 事件统计

关于事件统计 API 的最佳实践说明见。自定义事件类型说明。

2.5 发送策略

发送策略定义了用户由统计分析 SDK 产生的数据发送回 ShareSDKStatistics 服务器的频率。

您需要在程序的入口 Activity 中添加

```
MobclickAgent.updateOnlineConfig( mContext );
```

Android 平台的数据发送策略有两种方式：

- 启动时发送：APP 启动时发送当次启动数据和上次的使用时长等缓存数据，当次使用过程中产生的自定义事件数据缓存在客户端，下次启动时发送
- 按间隔发送：按特定间隔发送数据，间隔时长介于10秒与1天之间。您可以在后台自定义发送间隔。

在没有取到在线配置的发送策略的情况下，会使用默认的发送策略：启动时发送。

你可以通过后台“设置-发送策略”来自定义数据发送的频率。

3.测试与调试

3.1 确认如下内容

- 确认所需的权限都已经添加：INTERNET, READ_PHONE_STATE
- 确认 APPKEY 已经正确的写入 Androidmanifest.xml
- 确认所有的 Activity 中都调用了 onResume 和 onPause 方法
- 确认测试手机(或者模拟器)已成功连入网络

3.2 使用普通测试流程（不推荐）

如果您不使用集成测试服务来测试数据，那您可以通过普通测试流程查看测试数据。
使用普通流程，您的测试数据会与用户的真实使用数据同时处理，从而导致数据污染。

使用普通流程，请先打开调试模式：

```
MobclickAgent.setDebugMode( true );
```

打开调试模式后，您可以在 logcat 中查看您的数据是否成功发送到服务器，以及集成过程中的出错原因等。

4.技术支持

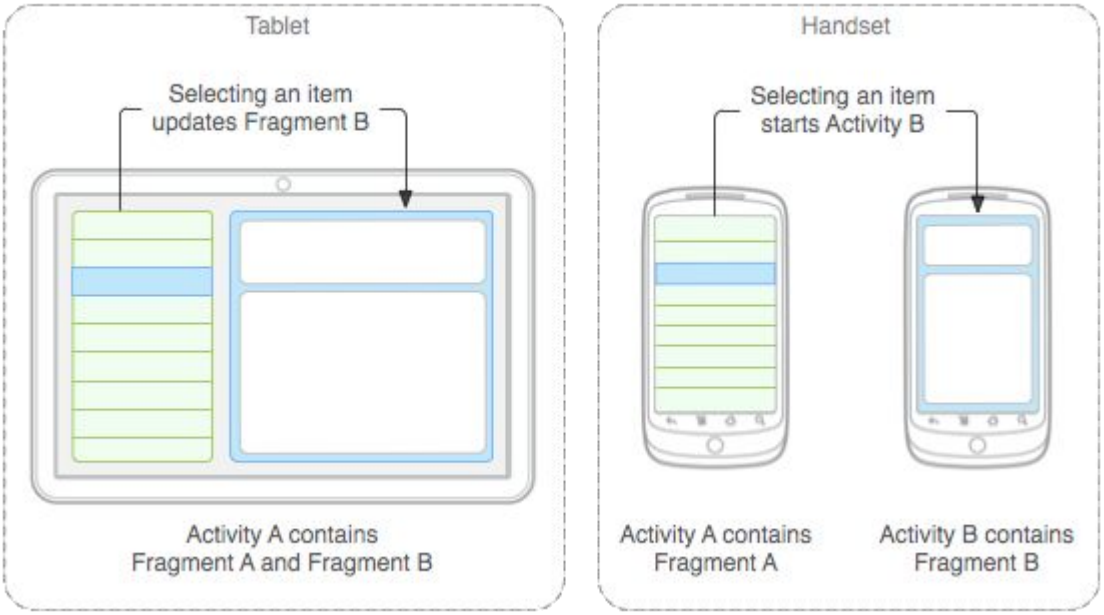
QQ:1449282202
Email：1449282202@qq.com

为了能够尽快响应您的反馈，请提供您的 appkey 及 logcat 中的详细出错日志，您所提供的内容越详细越有助于我们帮您解决问题。

5.Fragment 页面统计

5.1、什么是 Fragment

Android3.0 发布之后，很多开发者已经在使用 Fragment 控件。1.0 版本中对 Fragment 页面的统计。使用 Fragment 布局的 App 可以如下。



5.2、统计分析及页面统计方案

页面统计都是通过 Activity 自动实现的。开发者注册 Activity，在每个 Activity 的 onResume 方法中调用 MobclickAgent.onResume(Context), onPause 方法中调用 MobclickAgent.onPause(Context)。这两个方法会回传页面序列以及页面时长至服务器，从而计算得到页面路径、页面时长、App 时长信息。

5.3、支持 Fragment 统计的方案

Android 3.0之后，Fragment 的出现，细化了页面的粒度。现在的应用可能由 Activity、Fragment、自定义的 View 组成。针对 Android 页面的灵活多样化，我们对页面访问的统计也增加了灵活性。

使用 ShareSDKStatistics 统计分析对 Fragment 统计，开发者需要：

- 首先在程序入口处调用 MobclickAgent.openActivityDurationTrack(false) 来禁止默认的 Activity 页面统计方式。
- 然后在每个期望被统计的页面的开始和结束方法中分别调用 MobclickAgent.onPageStart(String pageName)，MobclickAgent.onPageEnd(String pageName)。这两个方法会回传页面序列以及页面时长至服务器，从而计算得到页面路径、页面时长、App 时长信息。被统计的页面要保证线性且不交叉，从而保证页面统计的正确性。

在 Activity 页面集成代码：

```
public void onResume() {
    super.onResume();
    MobclickAgent.onPageStart("SplashScreen");
    MobclickAgent.onResume(this);
}
public void onPause() {
    super.onPause();
    // 保证 onPageEnd 在 onPause 之前调用,因为 onPause 中会保存信息
    MobclickAgent.onPageEnd("SplashScreen");
    MobclickAgent.onPause(this);
}
```

在 Fragment 页面集成代码：

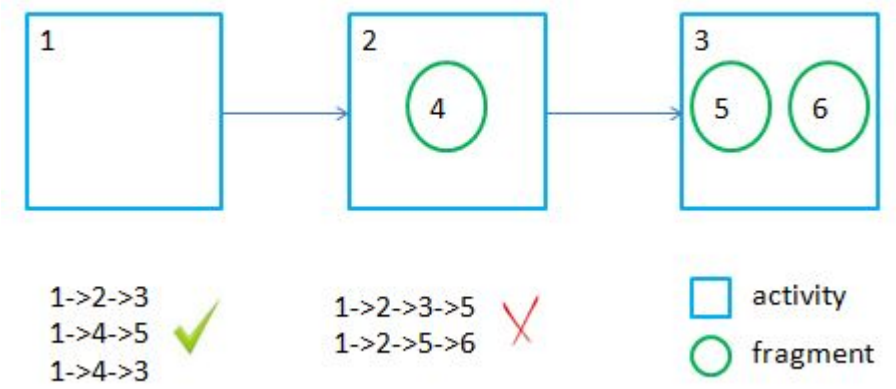
```
public void onResume() {
    super.onResume();
```

```
    MobclickAgent.onPageStart("MainScreen");
}
public void onPause() {
    super.onPause();
    // 保证 onPageEnd 在 onPause 之前调用,因为 onPause 中会保存信息
    MobclickAgent.onPageEnd("MainScreen");
}
```

5.4、策略开关

开发者可以选择基于 Activity 页面统计的方式，也可以选择基于 Fragment 页面统计的方式。在一个 App 中，能且仅能选择其中一种统计方式。通过 MobclickAgent.openActivityDurationTrack(boolean traceActivity) 来控制。

5.5、使用案例



图为三种典型的页面统计场景

5 纯粹基于 Activity 的页面

6 Activity 中包含 Fragment 的页面（也有可能是 Fragment 包含 Fragment）

7 Activity 中包含多个并列 Fragment 的页面

对 于场景1 只需要按照文档说明在 Activity 调用相关函数即可。但是对于场景2，开发则可以统计外层测 Activity 或者内部的 Fragment 。对于场景3一个

Activity 中包含多个 Fragment 开发者可以统计外层 Activity 或者 里面包含的任一个 fragment。

6.行为的统计：自定义事件

自定义事件可以实现在应用程序中埋点来统计用户的点击行为。

6.1 使用自定义事件的依赖条件

- 使用自定义事件功能请先在网站应用管理后台(设置->事件)中添加相应的事件后，服务器才会对相应的事件请求进行处理。
- 自定义事件的代码需要放在 Activity 里面的 onResume 方法后面。

6.2 计数事件

使用计数事件需要在后台添加事件时选择“计数事件”。

6.2.1 统计发生次数

在您希望跟踪的代码部分，调用如下方法：

```
MobclickAgent.onEvent(Context context, String eventId);
```

eventId 为当前统计的事件 ID。

示例：统计微博应用中"转发"事件发生的次数，那么在转发的函数里调用

```
MobclickAgent.onEvent(mContext,"Forward");
```

6.2.2 统计点击行为各属性被触发的次数

考虑事件在不同属性上的取值，可以调用如下方法：

```
MobclickAgent.onEvent(Context context, String eventId, HashMap<String,String> map);
```

map 为当前事件的属性和取值（键值对）。

示例：统计电商应用中“购买”事件发生的次数，以及购买的商品类型及数量，那么在购买的函数里调用：

```
HashMap<String,String> map = new HashMap<String,String>();

map.put("type","book");

map.put("quantity","3");

MobclickAgent.onEvent(mContext, "purchase", map);
```

6.3 计算事件

使用计算事件需要在后台添加事件时选择“计算事件”。

6.3.1 统计数值型变量的值的分布

统计一个数值类型的连续变量，用户每次触发的数值的分布情况，如事件持续时间、每次付款金额等，可以调用如下方法：

```
public static void onEvent(Context context, String id, HashMap<String,String> m, long value){

    m.put("__ct__", String.valueOf(value));

MobclickAgent.onEvent(context, id, m);

}
```

6.3.2 统计点击次数及各属性触发次数

计算事件除能够统计数值型的参数外还具有计数事件的所有功能。自定义事件类型说明。

6.4 注意事项

- event_id 和 tag 不能使用特殊字符，且长度不能超过128个字节；map 中的 key 和 value 都不能使用特殊字符，key 不能超过128个字节，value 不能超过256个字节
- id， ts， du 是保留字段，不能作为 eventId 及 key 的名称。

- 每个应用至多添加500个自定义事件，每个 event 的 key 不能超过10个，每个 key 的取值不能超过1000个（不允许通过 key-value 结构来统计类似搜索关键词，网页链接等随机生成的字符串信息）。 如有任何问题，请联系客服 qq: 1449282202。

6.5.错误统计

SDK 通过 Thread.UncaughtExceptionHandler 捕获程序崩溃日志，并在程序下次启动时发送到服务器。如果开发者自己捕获了错误，需要上传到服务器可以调用下面方法：

```
public static void reportError(Context context, String error)

//或

public static void reportError(Context context, Throwable e)
```

附录

自定义事件类型说明

ShareSDKStatistics 为用户提供了自定义事件的功能，用于追踪用户行为，记录行为发生的具体细节。 我们提供了两种自定义事件的形式：计数事件、计算事件。

1. 计数事件

计数事件统计事件的发生次数、独立用户数、事件时长及事件各参数的发生次数、时长。

事件 ID	事件名称	发生次数	独立用户数
click	按钮点击	5881	931
play	播放	6322	1003
shop	进入商店	2318	335

针对 play(播放)事件，还可以追踪到更细节的参数，如：

参数=style(歌曲类型)

参数值	发生次数	次数占比
轻音乐	5500	20.6%
摇滚	2778	10.4%
民歌	1335	5%

参数=singer(歌手)

参数值	发生次数	次数占比
刘若英	7985	23%
梁静茹	6323	18.2%
周杰伦	5572	16%

计数事件主要是以事件 ID+参数+参数值为统计项，统计相应的 PV、UV，其本质是针对字符串信息的计数。随着移动开发者运营的不断深入，这个方法的局限性逐渐显露。

如，一款拍照美化类应用，想了解相册的使用次数及每次打开相册的照片浏览张数。那么，以打开相册为事件，浏览张数为普通参数，可以得到如下报表：

参数值	发生次数	次数占比
1	217623	15.9%
2	171438	12.5%
3	122709	8.9%
4	93647	6.8%
5	74712	5.4%
6	60622	4.4%
7	49559	3.6%

虽然能得到照片浏览的张数，但这张报表的可读性较差。如果想了解平均每人浏览多少张照片，大多数人浏览多少张照片等信息还需运营人员进一步加工和整理。

类似的情景还有：

- 一款游戏，想了解每天多少人购买道具，以及购买道具的金币消耗值；
- 一款电商类应用，想了解每天多少人购买商品，以及消费额；
- 一款媒体类应用，想了解每天多少人浏览内容，以及浏览内容的数量；

等等。

计数事件调用 api 案例：音乐案例

2. 计算事件

程序中的某些事件，如登录、分享、下载等，是定性变量（categorical variable），对应的统计项是字符串类型。开发者只需了解它们发生的次数及独立用户数，使用计数事件即可满足。

还有一些事件，如上面提到的支付金额、内容浏览数量等是连续变量，对应的统计项是数值类型。开发者需要查看这些事件的数值分布特征，这就需要使用计算事件。

计算事件可以解决的一些问题：
（1）对某个参数值求和
（2）计算某个参数值的单次均值及单用户均值
（3）近似得到某个参数值的概率分布函数[1]
（4）近似得到某个参数值的累计分布函数[2]

其中，
概率分布函数由**p%分位数取值**近似估计。按次数（人数）的分位数取值，即单次（单设备）的参数值从小到大排列后第 p%的取值。
累计分布函数由**p%累计值贡献**近似估计。按次数（人数）的累计值贡献，即单次（单设备）的取值小于某数的所有次数（设备）的累计值贡献了总体累计值的 p%

以游戏中的“道具购买”事件为例，如果想了解道具购买的情况，可以这样上报自定义事件

```
itemsBuy(itemId=001, payment=50)
itemsBuy(itemId=003, payment=25)
itemsBuy(itemId=012, payment=20)
...
```

使用计算事件，可以得到如下报表：

购买次数	购买人数	购买金额	单次购买金额	人均购买金额
1100	1000	54980	50	55

人均购买金额为55元，这是否意味着大部分玩家的购买额都在50-60元之间呢？回答这个问题，需要进一步了解人均购买金额的分布情况：

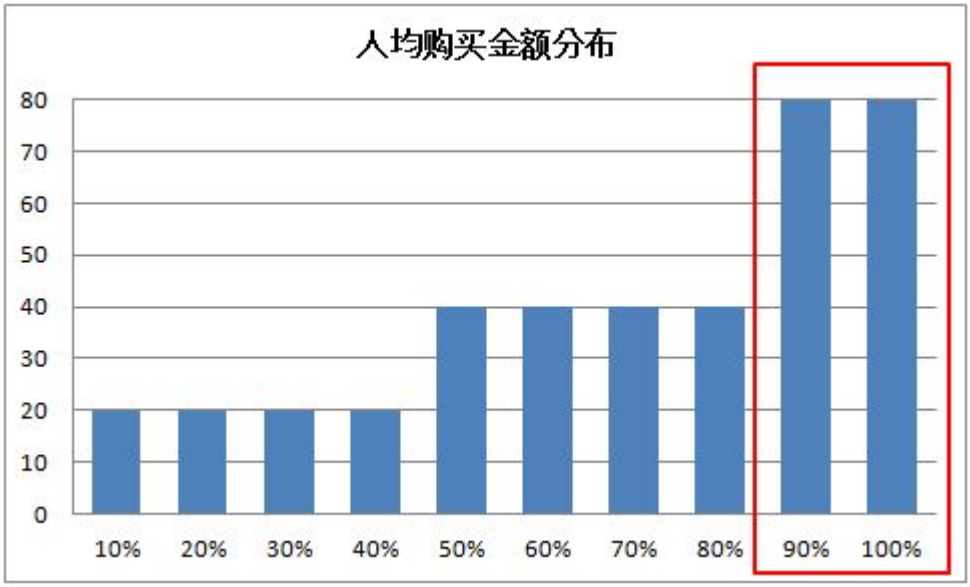


图1

从图1不难发现，80%的玩家购买金额在40元以下，大部分玩家的消费额是较低的。存在少量玩家（20%）不惜消耗重金购买道具，这部分玩家的人均购买金额为80元，而他们的消费总额占到了全部玩家累计购买金额的40%（图2）。

图2

在产品运营的过程中，应该更重视这部分高消费玩家的体验，给予及时的响应和回馈。而对剩余玩家，我们了解到他们的消费力在20-40元区间内，可以根据这个数据合理定价，开展适当的促销活动。

由上可知，针对计算事件，除了统计常规的发生次数、人数，还需做出定量的分析，才能更准确的了解用户的行为，为产品运营和决策提供帮助。

计算事件调用 api 案例：游戏案例

3. 配置

使用计算事件需要
1. 集成 ShareSDKStatistics 统计分析 SDK，并配置自定义事件 — 计算事件
2. 在 ShareSDKStatistics 后台创建事件 id，选择类型为计算事件

3.1 SDK 集成

我们通过将 ekv 事件的某参数标记成数值型来实现对计算事件的统计，您可以通过如下方式配置数值型参数：

Android 平台

```
   public static void onEvent(Context context, String id, HashMap<String,String> m, long value){

         m.put("__ct__", String.valueOf(value));

     MobclickAgent.onEvent(context, id, m);

}
```

注意：
1.每个事件只能有一个__ct__ 字段，即每个事件只能统计一个数值型参数。
2.__ct__ 字段的取值是不超过32位的 int 型。

3.2 后台设置

如果要使用数值型的参数，请在后台添加自定义事件时，选择“计算事件”的事件类型。

4. 查看

入口1：在事件列表页面点击“查看”进入某事件的详情报表中。若为计算事件，会默认显示该事件的数值型参数报表。

入口2：在事件详情页快速切换同一事件的数值型及字符串型参数的报表

5. FAQ

Q：为什么有的事件有字符串型和数值型两页报表？
A：对于计数事件，仍是原来的一页报表，即事件消息数和参数取值分布。
对于计算事件，字符串型报表页仍展示原来的事件消息数和参数取值分布；数值型报表页展示数值型的事件消息数、数值型参数的取值分布及累计贡献值分布。

Q：计算事件在数值型报表页和字符串型报表页中的事件消息数为何会不相等？
A：有的事件在发生时不一定会触发所有的参数。只要触发了数值型参数，那么这次消息在数值型报表中的事件消息数中会被计算；只要触发了字符串型参数，那么这次消息在字符串型报表中的事件消息数中会被计算。一次事件可能在数值型参数和字符串型参数的事件消息数中同时被计算。

Q：如果同一个事件传了2个数值型参数，会怎样？
A：每个事件只能有一个__ct__ 字段，如果同时统计了多个数值型参数，则后面的调用会将前面的结果覆盖，最终只回传一个数值结果。

Q：自定义事件在使用时有哪些限制？
A：自定义事件至多传递10个参数，且每个事件只能传递一个数值型参数。 自定义事件和参数名的长度不能超过64个字符；字符型参数值的长度不能超过256个字符，数值型参数的取值不能超过32位的 int 型；字符型参数至多传递1000个不同取值。

1 概率分布函数：描述参数值在某个确定的取值点附近的可能性的函数。
2 累计分布函数：参数值落在某个区域之内的概率则为概率密度函数在这个区域上的积分。当概率密度函数存在时，累计分布函数是概率密度函数的积分。

自定义事件案例

自定义事件用于追踪用户的行为，包括记录行为发生的次数、持续的时长以及该行为的具体细节。不同的需求可以采用不同的方式来跟踪。

1.音乐案例

如果想监控 music play（播放音乐）事件，那么

1. 跟踪播放音乐事件发生的总次数和平均每次启动该事件发生的次数，只需要下面一行代码：

```
MobclickAgent.onEvent(this, "music_play");
```

2. 跟踪播放音乐事件发生的总时间：

在音乐播放开始的时调用：

```
MobclickAgent.onEventBegin(this, "music_play");
```

在音乐播放结束时调用：

```
MobclickAgent.onEventEnd(this, "music_play");
```

3. 播放音乐事件还有很多其它特征：比如所播放音乐的风格、表演者，用户播放音乐时的状态（登录或匿名）。了解这些细节需要下面的代码：

```
Map<String, String> music = new HashMap<String, String>();

music.put("type", "popular");

music.put("artist", "JJLin");

music.put("User_status", "registered");

MobclickAgent.onEvent(this, "music", music);
```

4. 更进一步，您可以捕捉到每种状态的持续时间，比如每种风格的音乐播放了多久。您需要采用如下方法：

```
Map<String, String> music = new HashMap<String, String>();

music.put("type", "popular");

music.put("artist", "JJLin");

music.put("User_status", "registered");
```

在音乐播放开始时调用：

```
MobclickAgent.onEventBegin(this, "music", music);
```

在音乐播放结束时调用：

```
MobclickAgent.onEventEnd(this, "music", music);
```

媒体类应用都可以依照上面的方法去追踪事件，比如阅读文章、观看视频、播放音乐等。这样可以清楚地了解哪些作品有更高点击率、更长展示时间，以及更受用户喜欢，从而在内容选择和推荐上更有针对性。

2.游戏案例

再来看一个游戏案例：这是一个即时策略的塔防游戏，采用的是内置付费模式。

1. 首先，我们想观察玩家在每次过关时的状态，那么可以监控”user_status”这个事件。

```
Map<String, String > player_status = new HashMap<String, String>();

status.put("gold", "30");

status.put("item", "gun");

status.put("paid", "false");

MobclickAgent.onEvent("level_one",status);
```

这样，我们就会知道每次过关时，等级怎样，剩余多少金币，是否购买玩了多少次，玩了多少时间。从而了解玩家在哪一关的流失率较高，接下来应该重点去改善。

2. 进一步，我们可以验证关卡难度，即监控”player_dead”这个事件。

```
MobclickAgent.onEvent(this, "level_one","player_dead");
```

可以观察玩家在哪一关的死亡率最高，是否与上面流失率最高的关卡吻合，这样能判断关卡难度是否是造成流失的主要原因。如果是，我们可以通过降低关卡 难度来让更多的玩家留在游戏中；如果不是，那么需要从其他方面入手，比如丰富游戏内容增加对玩家吸引力、加强对新手的引导环节等。

3. 购买监控道具购买是该游戏主要的收入模式，我们非常关心用户购买行为的发生次数，那么我们可以监控”purchase”这个事件：

```
MobclickAgent.onEvent(this, purchase);
```

购买这个事件还有很多细节特征可以记录。比如，玩家是在哪些关购买的道具、购买道具的类型是什么、是否购买成功、购买道具时人物处于什么样的角色等等。您可以这么做:

```
Map<String, String> purchase = new HashMap<String, String>();

purchase.put("level", "3");

purchase.put("item", "sword");

urchase.put("succeed", "true");

purchase.put("hero_level", "5");

MobclickAgent.onEvent("purchase",purchase);
```

知道了玩家更喜欢在哪几关购买道具，那么我们可以设法尽量让更多玩家到达那一关；知道了玩家喜欢什么样的道具，大概是什么样的功能和定价，那么我们可以考虑优化道具设计、适当促销，从而优化收益。