# 集成文档

## 1.集成准备

### 1.1获得Appkey

集成友盟SDK之前，您首先需要到[友盟官网](http://dev.umeng.com/analytics/android-doc/www.umeng.com)注册并且添加新应用，获得Appkey

* **特别提醒**：我们建议开发者在注册账号时使用企业邮箱，避免使用个人邮箱注册，防止由于个人离职带来的问题，建议使用的账号形式 ：umeng@企业域名、apps@企业域名、dev@企业域名。



#### 1.1.1 FAQ

**Q**：应用的安卓版和IOS版能否共用一个appkey

**A**：不同平台的应用禁止使用相同的Appkey，需要分开注册。

**Q**：注册应用时，提示“应用名称已存在”

**A**：友盟后台的应用名与实际应用名和包名无关，建议命名为“应用名+平台（IOS/Android)”。

### 1.2下载SDK

请使用libs里的SDK

### 1.3导入SDK

导入SDK有以下两种方法，选择其中一种即可 。

新版SDK支持Android 6.0，编译时请指定编译版本为6.0及以上。

#### 1.3.1 下载SDK并集成

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）

#### 1.3.2使用Android Studio导入SDK

Android Studio是谷歌推出了新的Android开发环境，友盟支持AndroidStudio的Gradle配置，如果您使用Android Studio开发，请按照如下的方法操作：

在Gradle依赖中添加：

dependencies {

compile 'com.umeng.analytics:analytics:latest.integration'

}

如果无法正常集成请添加如下代码 ：

allprojects {  
 repositories {  
 mavenCentral()   
 }  
}

## 2 基本功能集成

### 2.0新增场景类型设置接口

场景类型设置接口。

MobclickAgent.setScenarioType(Context context, EScenarioType etype)

etype是官方场景，有如下场景:

EScenarioType. E\_UM\_NORMAL 普通统计场景类型

EScenarioType. E\_UM\_GAME 游戏场景类型

EScenarioType. E\_UM\_ANALYTICS\_OEM 统计盒子场景类型

EScenarioType. E\_UM\_GAME\_OEM 游戏盒子场景类型

### 2.0.1UTDID相关

为防止冲突UTDID包单独提供,如果需要请导入UTDID包

### 2.1配置APPKEY和SecretKey

#### 2.1.1 XML配置appkey

manifest的配置主要包括添加权限，代码示例如下：

<manifest……>

<uses-sdk android:minSdkVersion="8"></uses-sdk>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"></uses-permission>

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

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

<uses-permission android:name="android.permission.READ\_PHONE\_STATE"></uses-permission>

<application ……>

……

<meta-data android:value="YOUR\_APP\_KEY" android:name="UMENG\_APPKEY"/>

<meta-data android:value="Channel ID" android:name="UMENG\_CHANNEL"/>

</application>

</manifest>

##### 权限说明

|  |  |
| --- | --- |
| 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) | 获取用户设备的IMEI，通过IMEI和mac来唯一的标识用户。 |
| ACCESS\_WIFI\_STATE(必须) | 获取用户设备的mac地址，在平板设备上，无法通过imei标示设备，我们会将mac地址作为用户的唯一标识 |
| 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。 |

#### 2.1.2.代码设置APPKEY

在程序的入口处,设置Appkey/Token/Channel.调用如下方法

代码中集成,需要在程序入口调用此接口:

MobclickAgent. startWithConfigure(UMAnalyticsConfig config)

UMAnalyticsConfig初始化参数类，提供多参数构造方式：

UMAnalyticsConfig(Context context, String appkey, String channelId)

UMAnalyticsConfig(Context context, String appkey, String channelId, EScenarioType eType)

UMAnalyticsConfig(Context context, String appkey, String channelId, EScenarioType eType,

boolean isCrashEnable)

构造意义：

String appkey:官方申请的Appkey

String channel: 渠道号

EScenarioType eType: 场景模式，包含统计、游戏、统计盒子、游戏盒子

Boolean isCrashEnable: 可选初始化. 是否开启crash模式

2.1.3 设置Secret Key

新增secret Key接口,防止appkey被盗用,secretkey网站申请

MobclickAgent.setSecret(Context context, String secretkey)

### 2.2.Session统计

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，导致启动次数异常增高。(eg.使用TabHost、TabActivity、ActivityGroup时)。
* 当应用在后台运行超过30秒（默认）再回到前端，将被认为是两个独立的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) 来自定义这个间隔（参数单位为毫秒）。
* 如果开发者调用Process.kill或者System.exit之类的方法杀死进程，请务必在此之前调用MobclickAgent.onKillProcess(Context context)方法，用来保存统计数据。

☺ **session统计的相关问题请参见友盟开发者社区：** [启动次数的统计原理](http://bbs.umeng.com/thread-6402-1-1.html)， [单次使用时长的统计原理](http://bbs.umeng.com/thread-6302-1-1.html)

至此，基本功能集成已经结束。

**非常重要：必须调用 MobclickAgent.onResume() 和MobclickAgent.onPause()方法，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。**

### 2.3.账号的统计

友盟在统计用户时以设备为标准，如果需要统计应用自身的账号（需使用5.6.1及以上版本SDK），请使用以下接口：

**public** **static** **void** **onProfileSignIn**(String ID) ;

**public** **static** **void** **onProfileSignIn**(String Provider, String ID) ;

* ID：用户账号ID，长度小于64字节
* Provider：账号来源。如果用户通过第三方账号登陆，可以调用此接口进行统计。支持自定义，不能以下划线"\_"开头，使用大写字母和数字标识，长度小于32 字节; 如果是上市公司，建议使用股票代码。

**public** **static** **void** **onProfileSignOff**();

* 账号登出时需调用此接口，调用之后不再发送账号相关内容。

示例：

当用户使用自有账号登录时，可以这样统计：

MobclickAgent.onProfileSignIn("userID");

当用户使用第三方账号（如新浪微博）登录时，可以这样统计：

MobclickAgent.onProfileSignIn("WB"，"userID");

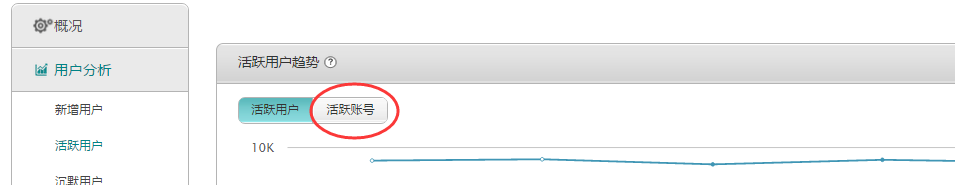
### 2.3.1.启动账号统计

集成账号统计功能后，请到我的产品-设置-应用信息，选择启动使用账号统计报表，如下图



如果您集成了新版SDK的账号接口，可以通过启动账号统计开关，来开启账号报表的展现。账号统计报表开关仅作用于报表的展现，不影响数据的计算。您可以根据实际的需要来启动或者暂停该功能。集成此功能的第二天才能在后台报表看见账号的数据。可在下图所示位置查看





### 2.4.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。

页面访问路径的数据为全量统计。若您希望查看测试数据，请使用 [集成测试](http://www.umeng.com/test_devices) ，了解更多集成测试请 [点击这里](http://dev.umeng.com/analytics/functions/testmode) 。

☺ **页面统计的常见问题参见友盟开发者社区：** [页面访问路径中你需要了解的知识点](http://bbs.umeng.com/thread-6281-1-1.html) ，[页面访问路径常见问题详解](http://bbs.umeng.com/thread-12608-1-1.html)

#### 2.4.1.只由Activity构成的应用

如果您已经完成2.3步骤之前的代码添加，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。

#### 2.4.2.包含Activity、Fragment或View的应用

统计应用中包含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的应用中，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。

首先，需要在程序入口处，调用 MobclickAgent.openActivityDurationTrack(false) 禁止默认的页面统计方式，这样将不会再自动统计Activity。

然后需要手动添加以下代码：

1. 使用 MobclickAgent.onResume 和 MobclickAgent.onPause方法统计时长, 这和基本统计中的情况一样(针对Activity)

2. 使用 MobclickAgent.onPageStart 和 MobclickAgent.onPageEnd 方法统计页面(针对页面,页面可能是Activity 也可能是Fragment或View)

如果页面是直接由Activity实现的，统计代码大约是这样：

**public** **void** **onResume**() {

**super**.onResume();

MobclickAgent.onPageStart("SplashScreen"); *//统计页面(仅有Activity的应用中SDK自动调用，不需要单独写。"SplashScreen"为页面名称，可自定义)*

MobclickAgent.onResume(**this**); *//统计时长*

}

**public** **void** **onPause**() {

**super**.onPause();

MobclickAgent.onPageEnd("SplashScreen"); *// （仅有Activity的应用中SDK自动调用，不需要单独写）保证 onPageEnd 在onPause 之前调用,因为 onPause 中会保存信息。"SplashScreen"为页面名称，可自定义*

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，"MainScreen"为页面名称，可自定义*

}

**public** **void** **onPause**() {

**super**.onPause();

MobclickAgent.onPageEnd("MainScreen");

}

**注意：**这些方法的调用，需要保证线性不交叉，每个 onResume 都对应一个 onPause ，每个 Start 都有一个 End 配对。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[这里](http://blog.umeng.com/?p=2928) 。

### 2.5.发送策略

发送策略设定了用户产生的数据发送回友盟服务器的频率。

Android平台的数据发送策略有两种方式：

1. 启动时发送：新增、活跃、启动次数等数据在APP本次打开时即刻发送，使用时长、自定义事件、错误统计等在使用过程中产生的所有消息数据都在下次启动应用时发送。如果应用程序启动时处在不联网状态，那么消息将会缓存在本地，下次再尝试发送。

2. 按间隔发送：按特定间隔发送数据，间隔时长介于90秒与1天之间。新增、活跃、启动次数等数据在APP本次打开时即刻发送，使用时长、自定义事件、错误统计等在使用过程中产生的所有数据都按间隔发送，如果应用程序启动时处在不联网状态，那么消息将会缓存在本地，下次再尝试发送。

#### 2.5.1.发送策略设置方法

您可以在后台统计分析->设置->发送策略 页面自定义发送间隔。具体如下图：



点击发送策略之后，可以看到设置页面



**注意**：

（1）在没有获取到在线配置时，默认使用启动时发送的策略。

（2）在打开debug调试模式或者使用集成测试时，不受发送策略控制。

### 2.6.日志加密设置

您可以通过在程序入口处的 Activity 中调用如下代码来设置加密模式

*/\*\* 设置是否对日志信息进行加密, 默认false(不加密). \*/*

MobclickAgent.enableEncrypt(boolean enable);

* 如果enable为true，SDK会对日志进行加密。加密模式可以有效防止网络攻击，提高数据安全性。
* 如果enable为false，SDK将按照非加密的方式来传输日志。
* 如果您没有设置加密模式，SDK的加密模式为false（不加密）。

### 2.7代码混淆

如果您的应用使用了混淆， 请添加

-keepclassmembers **class** \* {

**public** <init> (org.json.JSONObject);

}

这是由于SDK中的部分代码使用反射来调用构造函数， 如果被混淆掉， 在运行时会提示"NoSuchMethod"错误。 另外，由于SDK需要引用导入工程的资源文件，通过了反射机制得到资源引用文件R.java，但是在开发者通过proguard等混淆/优化工具处理apk时，proguard可能会将R.java删除，如果遇到这个问题，请在proguard配置文件中添加keep命令如：

-keep public class [您的应用包名].R$\*{

public **static** final **int** \*;

}

把[您的应用包名] 替换成您自己的包名，如com.yourcompany.example。如果您使用5.0.0及以上版本的SDK，请添加如下命令：

-keepclassmembers **enum** \* {

**public** **static** \*\*[] values();

**public** **static** \*\* valueOf(java.lang.String);

}

## 3.自定义事件的统计

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。自定义事件目前包括“计数事件”和“计算事件”，二者的区别以及详细说明请[点击这里](http://dev.umeng.com/analytics/functions/numekv)。

☺ **自定义事件使用中的问题请参见友盟开发者社区：** [自定义事件常见问题](http://bbs.umeng.com/thread-5417-1-1.html)，[关于自定义事件的那些事儿](http://bbs.umeng.com/thread-11284-1-1.html)

### 3.1.自定义事件字段说明

event id：自定义事件id

key：自定义事件下的参数

value：自定义事件参数下的参数值

### 3.2.使用自定义事件的依赖条件

1. 使用自定义事件功能请先登陆[友盟官网](http://www.umeng.com/) ， 在 “统计分析->设置->事件” （子账户由于权限限制可能无法看到“设置”选项，请联系主帐号开通权限。）页面中添加相应的事件id（事件id可用英文或数字，不要使用中文和特殊字符且不能使用英文句号“.”您可以使用下划线“\_”），然后服务器才会对相应的事件请求进行处理。
2. 自定义事件的代码需要放在Activity里的onResume--onPause之间，请在友盟初始化之后调用事件,不支持在service中统计。

### 3.3.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”。

#### 3.3.1.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);

context指当前的Activity

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");

#### 3.3.2.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 map);

map为当前事件的属性和取值（Key-Value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);

### 3.4.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”。

#### 3.4.1.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（该变量必须为整数），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：

MobclickAgent.onEventValue(Context context, String id, Map<String,String> m, int du)

id 为事件ID  
map 为当前事件的属性和取值  
du 为当前事件的数值为当前事件的数值，取值范围是-2,147,483,648 到 +2,147,483,647 之间的有符号整数，即int 32类型，如果数据超出了该范围，会造成数据丢包，影响数据统计的准确性。

示例：统计一次音乐播放，包括音乐类型，作者和播放时长，可以在音乐播放结束后这么调用：

int duration = 12000; *//开发者需要自己计算音乐播放时长*

　　Map<String, String> map\_value = **new** HashMap<String, String>();

　　map\_value.put("type" , "popular" );

　　map\_value.put("artist" , "JJLin" );

MobclickAgent.onEventValue(**this**, "music" , map\_value, duration);

上面方法在Android统计分析V5.2.2之后(>=)才提供，之前的版本可以通过下面的方式封装计算事件：

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：onEventBegin(...),onEventEnd(...)以及onEventDuration(...)后续不再支持，不建议使用。  
- 如需统计使用时长，开发者需要自己上传时长数据（如示例中的duration所示），时长数据会在计算事件的数值型部分展示；  
- 如果使用了onEventBegin(...),onEventEnd(...)以及onEventDuration（...）方法，时长数据会在计算事件的字符串型中显示。**

如图：使用了onEventBegin(...),onEventEnd(...)以及onEventDuration（...）方法，时长数据会显示在【消息时长】选项中，为了兼容老用户，线上仍保留这个选项，此方法已不建议使用。 

如图：开发者使用计算事件自己上传时长数据（如示例中的duration所示），时长数据会在计算事件的数值型部分展示。 

#### 3.4.2.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，[详细说明](http://dev.umeng.com/analytics/functions/numekv)。

### 3.5.注意事项

1. event id不能使用特殊字符，不建议使用中文，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。
2. id， ts， du是保留字段，不能作为event id及key的名称。
3. 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。如需要统计支付金额、使用时长等数值型的连续变量，请使用计算事件（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）。若需要提高指标上限，请[申请APP企业认证](http://www.umeng.com/enterprise_certificate)，了解企业认证[点击这里](http://www.umeng.com/enterprise_certificate_intro)。

### 3.6.在webview中统计自定义事件

如果你的页面中使用了WebView 嵌入HTML,js 的代码，并且希望统计HTML中的点击事件，比如点击了某个链接的次数， 这时你需要阅读下面的文档，通过友盟的 js 接口来统计相关的事件。注意如果你的HTML是运行在浏览器的，那么还 是无法统计的，下文仅针对使用WebView 加载 网页的情况。

* Java代码集成 首先要找到你的WebView对象，并做下面的初始化工作。 WebView webview = (WebView) findViewById(R.id.webview); new MobclickAgentJSInterface(Context, webview);

public MobclickAgentJSInterface(Context context, WebView webview, WebChromeClient client)

context 当前Activity引用  
webview 当前的WebView对象  
client WebChromeClient 实例，如果开发者实现了自己的WebChromeClient，那么请传入这个实例

public MobclickAgentJSInterface(Context context, WebView webview)

context 当前Activity引用  
webview 当前的WebView对象

集成了上面的代码后，我们将会监听友盟的 js 代码调用。

* js 代码集成在网页端，请添加如下的 js 代码：
* <script type="text/javascript">
* **function** **onEvent**(tag, label, duration) {
* prompt("event", JSON.stringify({tag:tag,label:label, duration:duration}));
* }
* **function** **onKVEvent**(tag, map, duration) {
* map.id = tag;
* map.duration = duration;
* prompt("ekv", JSON.stringify( map ));
* }
* </script>

这里面提供了两个js接口，分别用来统计自定义事件和K-V对形式的KV事件，并且和Java接口有如下的映射：

onEvent(tag) -> MobclickAgent.onEvent(Context context, String tag);

*//tag 传入字符串*

onEvent(tag, label)-> MobclickAgent.onEvent(Context context, String tag, String label);

*// tag ,label 传入字符串*

onEvent(tag, '', du) -> MobclickAgent.onEvent(Context context, long duration);

*// tag 传入字符串， 第二个变量 传入空字符串, du 传入 数值类型*

onEvent(tag, label, du) -> MobclickAgent.onEvent(Context context, String tag, String label, long du);

*// tag, label 传入字符串, du 传入数值类型*

onKVEvent(id, {} ) -> MobclickAgent.onEvent(Context context, HashMap map);

*// id 传入字符串， 第二个参数传入 js 对象eg：{'item':'gun','type':'weapon'}*

onKVEvent(id, {} , du) -> MobclickAgent.onEvent(Context context, HashMap map, long du);

*// id 传入字符串， 第二个参数传入 js 对象eg：{'item':'gun','type':'weapon'}, du 传入 数值类型*

* 示例：
* <a onClick="onEvent('click')"> SimpleEvent</a>
* <a onClick="onEvent('click','button')"> EventWithLabel </a>
* <a onClick="onEvent('play','music',123)"> EventWithLabelAndDuration </a>
* <a onClick="onEvent('play','',123)"> EventWithDuration </a>
* <a onClick="onKVEvent('buy',{'item':'gun'})"> KVEvent </a>
* <a onClick="onKVEvent('play',{'music':'blue'}, 123)"> KVEventWithDuration </a>
* 实现原理：  
  如您所见，我们提供的 js 接口，最后都会调用一个 prompt(...) 方法， 然后我们通过拦截 prompt 接口的事件，得到 js 代码中的参数，然后再映射到 Java 代码上的指定方法。所以我们在第一步初始化的时候，实现了自己的WebChromeClient 然后覆盖其中的方法来完成 prompt 事件的拦截，如果您的代码中同样实现了自己的 WebChromeClient 请务必在初始化的时候，调用以下代码，并传入相应的 WebChromeClient 。
* **public** \*MobclickAgentJSInterface\*(Context context, WebView webview, WebChromeClient client)

**说明：**   
上面示例的完整代码可以在 example 程序的 assets/demo.html 文件夹下找到。对应的Java初始化代码可以在 com.umeng.example.analytics.AnalyticsHome.java 中找到。

## 4.错误统计

Android统计SDK从V4.6版本开始内建错误统计，不需要开发者再手动集成。

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。 如不需要错误统计功能，可通过此方法关闭

MobclickAgent.setCatchUncaughtExceptions(false);

如果开发者自己捕获了错误，需要上传到友盟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)

使用自定义错误，查看时请在错误列表页面选择【自定义错误】

☺ **错误统计的常见问题参见友盟开发者社区：** [友盟错误分析常见问题汇总](http://bbs.umeng.com/thread-6310-1-1.html)

## 5.在线参数

**注意：**在Android统计分析SDK5.6.1及以上版本中，setOnlineConfigureListener()这个接口已经废弃，如果使用在线参数功能，需下载独立的在线参数SDK，下载请[点击这里](http://dev.umeng.com/online-parameters/android-doc/sdk-download) 。

**以下【在线参数】内容仅针对5.6.1以前版本的SDK**

在线参数使您可以在网站上动态配置预设好的参数的参数值。例如： 动态修改应用的欢迎语，修改应用中开关选项的"on"或"off"，以及类似游戏中虚拟物品的价格等。

您需要在程序的入口 Activity 中添加

MobclickAgent.updateOnlineConfig( mContext );

在线参数需要在网站编辑，在程序中使用这句代码从服务器获取参数，并缓存本地。 当在项目里需要获取某个具体参数时调用

String value = MobclickAgent.getConfigParams( mContext, "xxxx" );

"xxxx"为友盟后台事先设置好的参数id。  
同时，我们提供了在线参数回调接口，注意此接口只在在线参数有变化的时候才会回调，实现此接口：

MobclickAgent.updateOnlineConfig(**this**);

MobclickAgent.setOnlineConfigureListener(**new** UmengOnlineConfigureListener(){

@Override

**public** **void** **onDataReceived**(JSONObject data) {

}

});

data 为在线参数回调函数，JSONObject 对应网页上编辑的参数值。

## 6.社交统计

针对社交行为的垂直统计，可以非常详尽地统计应用中发生的各种社交行为。 只需要调用一行代码，便可享用到丰富的社交行为和社交用户分析报表。

在发生社交行为，比如分享了到新浪微博之后，调用这样一行代码：

UMPlatformData platform = **new** UMPlatformData(UMedia.SINA\_WEIBO, "user\_id");

platform.setGender(GENDER.MALE); *//optional*

platform.setWeiboId("weiboId"); *//optional*

MobclickAgent.onSocialEvent(**this**, platform);

会把分享信息发送到友盟服务器, 我们会通过这些信息创建社交行为报表。

相关参数说明： UMPlatformData：  
UMeida meida平台枚举类型（必填）  
String user\_id用户的id（必填）  
String weiboId 微博id   
String name用户姓名   
GENDER gender 用户性别

## 7.其他API

* MobclickAgent.setOpenGLContext(GL10 gl) 设置openGL 信息，辅助统计GPU 信息
* MobclickAgent.onKillProcess(Context context) 如果开发者调用 Process.kill 或者 System.exit 之类的方法杀死进程，请务必在此之前调用此方法，用来保存统计数据。
* android6.0中采集mac方式变更，新增接口 public static void setCheckDevice(boolean enable)该接口默认参数是true，即采集mac地址，但如果开发者需要在googleplay发布，考虑到审核风险，可以调用该接口，参数设置为 false 就不会采集mac地址。

## 8.测试与调试

### 8.1.确认如下内容

* 确认所需的权限都已经添加：INTERNET, READPHONESTATE
* 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

### 8.2.使用集成测试服务（推荐）

**什么是集成测试？** 集成测试是通过收集和展示已注册测试设备发送的日志，来检验SDK集成有效性和完整性的一个服务。 所有由注册设备发送的应用日志将实时地进行展示，您可以方便地查看包括应用版本、渠道名称、自定义事件、页面访问情况等数据，提升集成与调试的工作效率。

**注意：** 使用集成测试之后，所有测试数据不会进入应用正式的统计后台，只能在“管理--集成测试--实时日志”里查看，您不必再担心因为测试而导致的数据污染问题，让数据更加真实有效的反应用户使用情况。

使用集成测试模式请先在程序入口处调用如下代码，打开调试模式：

MobclickAgent.setDebugMode( true );

☺ 使用集成测试请点击[这里](http://www.umeng.com/test_devices?ticket=ST-1450072700rPWK4t4q46xw-qpBXzk)。

### 8.3.使用普通测试流程

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。 使用普通测试流程，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。

使用普通测试流程，请先在程序入口添加以下代码打开调试模式：

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友盟服务器，以及集成过程中的出错原因等，友盟相关log的tag是MobclickAgent。