友盟统计

Android SDK V3.1.1 开发指南

更新日志

V3.1.1
更新时间：2012-01-04
更新项目：

- 修复 Android4.0 自动更新 bug
- 增加 session 长度控制的接口
- 增加按天提示自动更新的接口

注：在 3.0 以上（包括 3.0）版本中我们对资源文件做了稍微的改动，需要注意

1. 使用自动更新功能时，引入 values 和 values-cn 文件夹下的 umeng_analyse_strings.xml 文件，同时替换 umeng_download_notification.xml 为 umeng_analyse_download_notification.xml

2.使用双向反馈功能时，需要把我们提供的（resource 文件夹下）所有资源复制到对应的文件夹下。

3.如果您要混淆 apk，要添加如下的混淆参数：

```
-keep public class [package name].R$*{
    public static final int *;
}
```

将[package name]替换为您的包名，避免 R 文件被中的资源被混淆。

```
-keep public class com.feedback.ui.ThreadView {
}
```

在使用双向反馈时，避免我们自定义的 UI 类被混淆。

友盟主站链接：
http://www.umeng.com/
友盟在线文档链接：
http://www.umeng.com/doc/sdk_android.html
友盟 FAQ 链接：
http://www.umeng.com/doc/faq.html

基本设置指南

1 注册应用，下载 SDK

登录你的帐号后，看到友盟的管理后台，点击"+添加新应用"，进入新应用信息填写的页面。在新应用信息填写中，请尽量填写真实的信息。App 建立成功后，可以获得该 App 的 AppKey，以及最新的开发指南和 SDK 文件。 *Tips*
您可以通过友盟统计分析平台的特性节省重复建立 App 的时间。如果您要对 App 不同的发布渠道进行统计，不需要创建新 App （请查看【使用分发渠道分析】），通过分发渠道分析，您可以更方便的对比数据。

2 基本步骤

2.1 导入 Analytics_Android_SDK_3.1.1.jar（简称 SDK）

下载最新版 sdk 的 zip 包，解压将其中的 **Analytics_Android_SDK_3.1.1.jar** 释放到本地目录，Eclipse 用户右键您的工程根目录，选择 Properties -> Java Build Path -> Libraries ，然后点击 Add External JARs... 选择指向 Analytics_Android_SDK_3.1.1.jar的路径，点击 OK，即导入成功。

2.2 配置 AndroidManifest.xml

- 添加应用程序的 Appkey （必须）
  需要先添加应用程序获得 Appkey，获得后写到 AndroidManifest.xml 的 meta-data 里。（注意：不要改变字符串'UMENG_APPKEY'）
- 添加权限 android.permission.INTERNET（必须）
  向我们的服务器发送用户分析数据。
- 添加权限 android.permission.READ_PHONE_STATE（必须）
  这个权限仅为了获取用户手机的 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)
- 添加权限 android.permission.WRITE_EXTERNAL_STORAGE（可选）
  如果您使用了友盟自动更新提醒功能，需添加这个权限，为了将更新的 APK 临时存在 SD 卡里。
- 添加权限 android.permission.ACCESS_NETWORK_STATE（可选）
  检测网络状态，用于自动更新前判断网络环境用，友盟 SDK 1.6版本新增权限。
- 添加权限 android.permission.READ_LOGS （可选）

如果您想获得客户端 crash 的报告，需要添加这个权限。具体见高级功能错误报告。示例程序如下：

```
<manifest ……>
    <application ……>
        ……
        <activity ……/>
        <meta-data android:value="4d5cf21c112cf713c5004cba"
android:name="UMENG_APPKEY" />
    </application>
    <uses-sdk android:minSdkVersion="4" />
    <uses-permission    android:name="android.permission.INTERNET"/>
    <uses-permission    android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission    android:name="android.permission.READ_LOGS"/>
</manifest>
```

2.3 添加代码

- **添加引用**

import com.mobclick.android.MobclickAgent

- **注册 Activity**

在每个 Activity 的 onResume 方法中调用 MobclickAgent.onResume(Context)，传入的参数为当前 context 的引用，这个方法将会自动地从 AndroidManifest.xml 文件里读取 Appkey。 这里请不要将全局的 application context 传入。

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

在每个 Activity 的 onPause 方法中调用 MobclickAgent.onPause(Context)，只需传入一个参数当前 activity 的 context.

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

2.4 集成说明

- 建议在所有的 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。如果在某些 activity 中不添加也可，但会造成相应 Activity 的使用时间无法被统计到。
- 一个应用程序在多个 activity 之间连续切换时，将会被视为同一个 session(启动)
- 如果您的 Activity 之间有继承关系或者控制关系，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(比如使用 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，30秒内返回之前的应用，统计数据仍计入前一次启动。在 V3.1.1以上版本中我们提供了新的接口来自定义这个时间间隔，您只要调用：MobclickAgent.setSessionContinueMillis(long)传入适当的参数，就可以控制 session 重新启动时间，注意参数是以毫秒为单位的。例如，如果您认为在60秒之内返回应用可视为同一次启动，超过60秒返回当前应用可视为一次新的启动，那么请写成 MobclickAgent.setSessionContinueMillis(60000).

2.5 注意事项

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

如果几分钟后报表中仍然没有收到数据，请与我们的技术支持联系 QQ ：800083942或者邮件到 support@umeng.com 我们会尽快回复您的报表.

高级基本设置指南

1 错误报告（Crash Report）

友盟统计分析工具，还可以帮助您捕捉用户在使用应用程序过程中出现的异常退出(FC)，并在应用程序下次启动时将错误报告发送给服务器。错误报告包含应用程序版本，操作

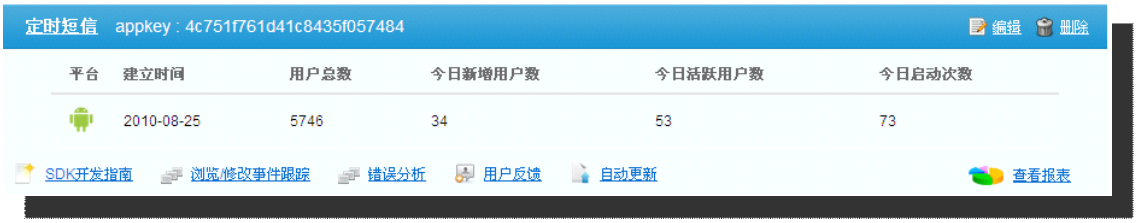
系统版本和设备型号以及程序出现异常时的 Stacktrace，这些数据将帮助您修正应用程序的 Bug。我们提供两种方式报告错误信息，一种是我们自动捕获的错误信息，一种是开发者自己传递的错误信息。

前者，您需要在 AndroidManifest.xml 里面添加权限 android.permission.READ_LOGS，并且在程序的 Main Activity（应用程序入口）的 onCreate 方法里调用 MobclickAgent.onError(Context):

```
public void onCreate(Bundle savedinstanceState) {
super.onCreate(savedInstanceState);
MobclickAgent.onError(this);
...
}
```

后者需要开发者调用 MobclickAgent.reportError(Context context,String error)方法，在第二个参数中传入自己捕获的错误信息。

您可以在我的产品页面的 “错误分析”标签或“查看报表”中查看错误报告。



2 自定义事件

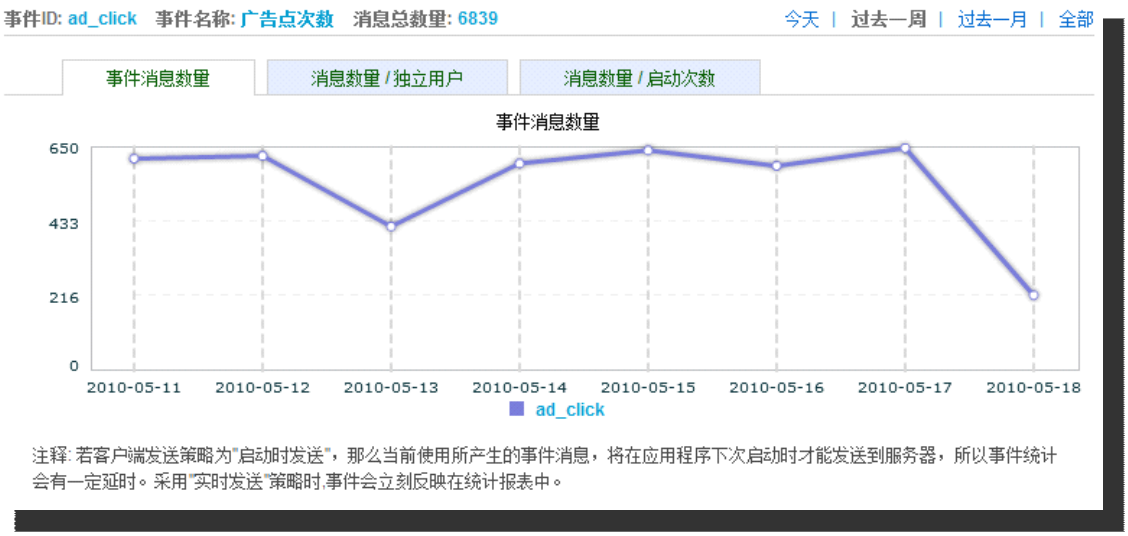
除了基本统计分析功能外，我们还支持您自定义的事件分析，例如您可以统计游戏中通过不同关卡的人数，广告的点击次数或者视频被播放的次数等等。这里我们将提供几个简单而通用的接口：

2.1 简单事件

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

在您希望发送事件报告的代码段，调用如下方法就可以向服务器发送事件记录，将统计 event_id 对应事件的发生次数，变化趋势，例如广告点击，短信数量等等。参数 context 为当前 context 的引用，event_id 为当前统计的事件 ID,注意 event_id 中不要加空格或其他的转义字符。

比如，监测应用程序里广告的点击次数，事件 ID 为“ad_click”。那么需要在程序里每次广告点击后调用 MobclickAgent.onEvent(this, "ad_click") 通知服务器一个广告点击事件。



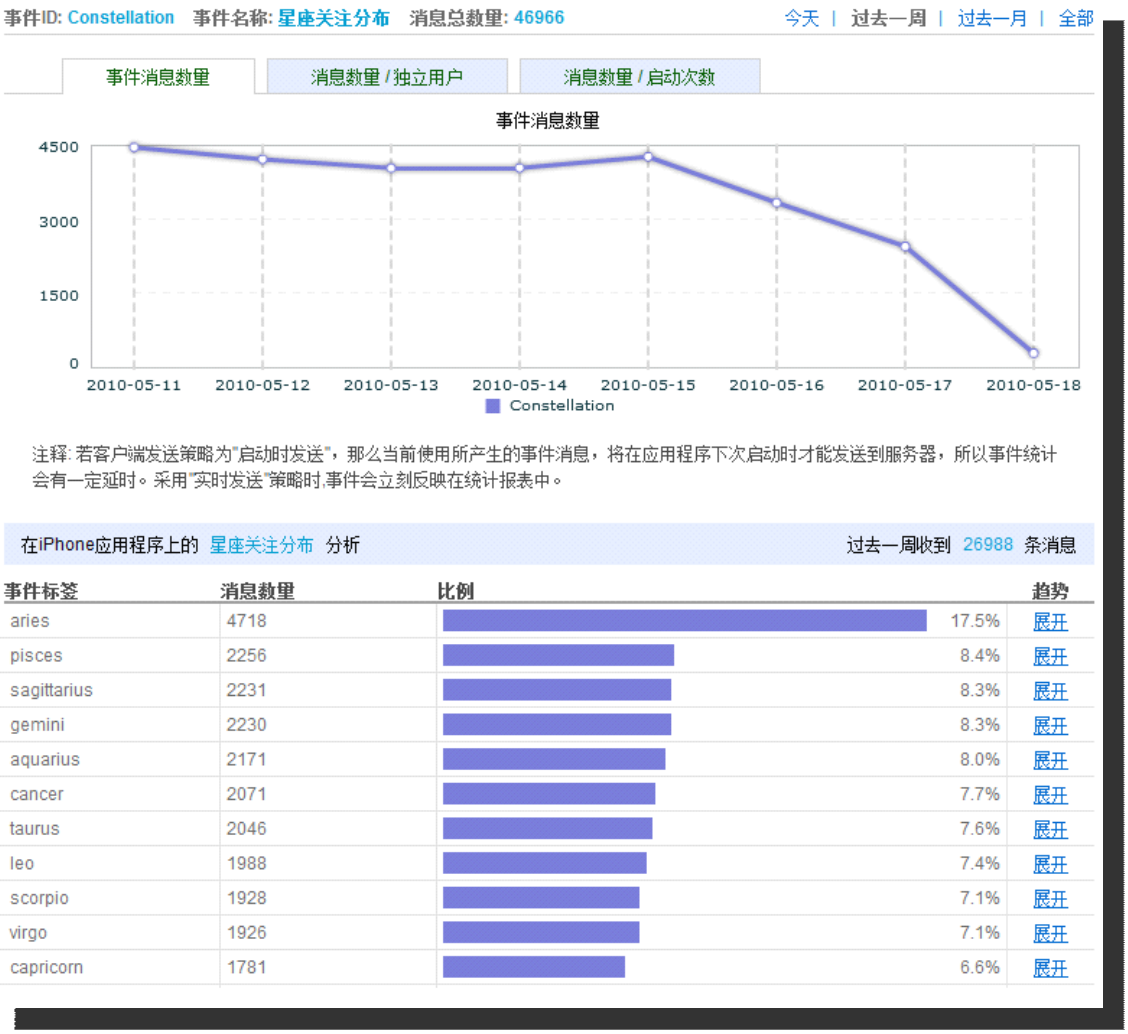
2.2 多标签事件

除了能够统计 event_id 所对应事件的发生次数，变化趋势外，还能统计事件中具体标签所占的比例。label 为当前标签，同样这里的 event_id 字符串中也不能有空格。

例如：在应用程序“星座罗盘”中，定义了一个事件“星座关注分布”，每个星座对应这个事件中的一个标签。我们可以在生产的统计图表中看到用户关注不同星座的比例。

```
MobclickAgent.onEvent(Context context, String event_id, String label);
```

多标签事件分析图表示例：



2.3 事件累计

在应用程序中某些自定义事件可能会被频繁触发，例如用户点击某个按钮。开发者可以在程序中维护一个计数器，这样某个事件被多次触发但只需要生成一个消息，这个消息中包括该事件被触发的次数。为了支持这个功能，这里我们简单重载了之前的两个接口：

MobclickAgent.onEvent(Context context, String event_id, int acc);
MobclickAgent.onEvent(Context context, String event_id, String label, int acc)

参数 acc 是对应事件 （和对应标签）被触发的次数。

3 分发渠道分析

有时需要统计应用程序的分发渠道，例如有多少用户来从联想乐园下载了您的应用,又有多少用户通过 Google android market 下载到您的应用程序。您只需要在

AndroidManifest.xml 里添加 meta-data，并将 value 属性修改为对应的发布渠道名。

```
<application ......    >
        <activity ……/>
        <meta-data                    android:value="Android                    Market"
android:name="UMENG_CHANNEL"/>
   </application>
```

当然，这需要您在不同渠道发布应用程序时，重新编译打包。（注意：不要改变 'UMENG_CHANNEL'）

4 应用程序更新

4.1 上传 APK

这个功能将帮助您把新版应用程序推送给用户，您只需要按照按照以下集成 sdk 每次更新应用程序，您只需要修改 AndroidManifest.xml 中的 VersionCode，并把应用程序的 apk 文件上传到友盟。

4.2 集成资源

将 zip 包中 layout 文件夹下的 umeng_analyse_download_notification.xml 和 values、values-en 文件夹下的 umeng_analyse_strings.xml 复制到您应用程序的 res/下对应的文件夹，此 XML 文件的作用是绘制状态栏里的下载进度条和多语言支持。

4.3 添加权限

```
android.permission.WRITE_EXTERNAL_STORAGE
```

4.4 基本功能

在应用程序的入口 Activity 里的 OnCreate() 方法中调用

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MobclickAgent.update(this);
}
```

考虑到用户流程的限制，目前我们默认在 Wi-Fi 接入情况下才进行自动提醒。如需要在其他网络环境下进行更新自动提醒，则请添加以下代码：

```
      MobclickAgent.setUpdateOnlyWifi(false);
```

同时，我们提供了一个判断下载状态的接口 MobclickAgent.*isDownloadingAPK*() 返回 boolean 类型的值，true 表示正在下载。
如果程序想自己处理更新可以按如下步骤，设置 updateAutoPopup= false，我们将禁止主动弹框，实现更新监听接口，处理更新事件：

```
MobclickAgent.update(this);
MobclickAgent.updateAutoPopup= false;
MobclickAgent.setUpdateListener(new UmengUpdateListener(){
@Override
public void onUpdateReturned(int arg) {
    switch(arg){
    case 0:                         //has update
        MobclickAgent.showUpdateDialog(DemoActivity.this);
        Log.i(TAG, "show dialog");
        break;
    case 1:                         //has no update
        Toast.makeText(getParent(), "has no update", Toast.LENGTH_SHORT).show();
        break;
    case 2:                         //none wifi
        Toast.makeText(getParent(), "has no update", Toast.LENGTH_SHORT).show();
        break;
    case 3:                         //time out
        Toast.makeText(getParent(), "time out", Toast.LENGTH_SHORT).show();
        break;
    }
}
});
```

4.5 按渠道更新

渠道更新，需要开发者在网站上按照渠道添加对应的更新包，渠道更新的规则是： SDK 检测本地安装的软件对应的渠道，然后连接服务器检测更新，如果服务器上有对应渠道的更新包 则选择此更新包并返回提示，如果服务器上没有对应渠道的更新包但有默认的更新包，则选择默认的 更新包并返回提示，如果服务器上没有默认的更新包，但是有其他渠道的更新包，将不会有更新提醒。

4.6 按时间间隔更新

通过对 update（Context）函数的简单重载，我们提供了一个按时间间隔更新的接口 update（Content, long），实现按一定的时间间隔进行更新，比如实现按天更新，只要传作如下调用：update(context, 1000*60*60*24),注意传入的参数，是以毫秒为单位的。这样在下次更新的时候，我们会先检查上次的更新时间，如果间隔超出了设定的值，我们才会再次提示更新。

4.7 机制说明

每次更新应用程序，您只需要修改 VersionCode，把应用程序的 apk 文件上传到友盟。MobclickAgent.update 方法会判断是否有新版应用程序，如果发现可更新的应用程序安装包，会提示用户是否更新。用户选择更新后，安装包会在后台下载自动安装更新。（按照 version code 来检测是否需要更新）

5 双向用户反馈（new）

为了让开发者和用户之间能顺畅沟通，我们隆重推出用户反馈的升级版本——双向用户反馈！用户可在使用应用的过程中提交反馈信息，开发者收到反馈后，在第一时间将回复推送给用户。您只需要：

5.1 添加资源包，并更新 Manifest 文件

将我们提供的 resource 文件夹下的所有资源文件（layout、values、values-en、drawable）复制到您的应用程序的对应文件夹下面。在 AndroidManifest.xml 中添加如下代码：

```
<activity android:name="com.feedback.ui.SendFeedback"
android:windowSoftInputMode="adjustResize" />
<activity android:name="com.feedback.ui.FeedbackConversations" />
<activity android:name="com.feedback.ui.FeedbackConversation"
        android:configChanges="keyboard" />
```

对于我们提供的默认 Layout xml 文件，您可以修改布局，但不要修改元素的 ID。

5.2 启动反馈

代码中启用 Feedback 模块，调用下面函数进入反馈界面。

```
UMFeedbackService.openUmengFeedbackSDK(this);
```

5.3 回复提醒

开发者回复用户反馈后，如需提醒用户，请在 Activity 的 OnCreate()方法中下添加以下代码：

```
UMFeedbackService.enableNewReplyNotification(this, NotificationType.AlertDialog);
```

方法第一个参数类型为：Context，第二个参数为枚举类型，可选值为 NotificationType.*AlertDialog* 或 NotificationType.*NotificationBar*，分别对应两种不同的提

示方式： *AlertDialog* 以 AlertDialog 方式提醒用户，*NotificationBar* 则使用 NotificationBar 方式提醒用户。
若调用该接口，反馈模块将在你程序启动后于后台检查是否有新的来自开发者的回复，若有，我们将提醒用户，若无，则不会打扰用户。
您也可以选择不调用该接口，这样我们会在用户进入反馈界面后，再去检查是否存在新的回复。

5.4 使用场景示例

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    UMFeedbackService.enableNewReplyNotification(this, 0);
}

public boolean onCreateOptionsMenu(Menu menu) {
    //一般可以在 OptionsMenu 中添加一个 Item,用于作为反馈界面的入口
    menu.add(0, 0, 0, "feedback");
    ...
    return true;
}

 public boolean onOptionsItemSelected(MenuItem item) {
     switch (item.getItemId()) {
       case 0:
         // 调用反馈提供的接口，进入反馈界面
         UMFeedbackService.openUmengFeedbackSDK(this);
         return true;
       default:
          ...
      }
 }
```

6 设置数据发送模式

6.1 模式解释

- **启动时发送（推荐使用）**

应用程序每次只会在启动时会向服务器发送一次消息，在应用程序过程中产生的所有消息(包括自定义事件和本次使用时长)都会在下次启动时候发送。 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。 发送策略默认为启动时发送。

- **实时发送**

应用程序产生每条消息(包括启动信息，自定义消息，退出消息)时都会立即发送到服务器。

- **设置每日单次发送**

设置每日单次发送后，当日第二次及之后的启动将不再发送数据，未发送的数据将缓存在手机，等待次日（或他日）发送。

注：由于不同应用的用户使用行为不同，数据发送会有相应的延迟。

6.2 设置发送模式

新版本的 SDK 支持在线动态配置发送策略，您需要在 onCreate（）函数中调用 MobclickAgent.updateOnlineConfig(this);方法，这样我们就可以在程序启动的时候，联网检测动态配置的发送策略了。详细配置见下面的9.使用在线配置功能。

旧版本 SDK（3.0 以前版本）发送模式设置需要在 onCreate（）函数中调用 MobclickAgent.setReportPolicy(int policy)，一旦应用发布就不能再更改,我们推荐使用最新版本的 SDK。如果您需要更灵活的发送模式， 请发邮件至 support@umeng.com，我们会尽快回复您。

7 使用在线配置功能

这个功能目前可以帮你在网站上动态配置两种类型的参数：

- 数据发送策略
- 自定义 key-value 型的键值对

在程序的入口 Activity 的 OnCreate()方法中调用

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MobclickAgent.updateOnlineConfig(this);
    }
```

这样程序启动的时候，我们将联网检测您的在线配置，并将这些信息保存在本地，等待下此启动的时候，我们将按照保存的配置来设定发送策略，您可以通过下面的函数读取您的自定义参数：

```
String MobclickAgent.getConfigParams(Context c, String key);
```

参数为 Context 和 在网站上编辑好的 key， 返回值是对应的 value 如果没有读到相应的 value 将返回空字符串。

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，实现次接接口：

```
MobclickAgent.updateOnlineConfig(this);
MobclickAgent.setOnlineConfigureListener(new UmengOnlineConfigureListener(){
        @Override
        public void onDataReceived(JSONObject data) {
```

```
            }
        });
```

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，通过下面函数设置（如果不设置的话，我们默认为启动时发送）

```
MobclickAgent.setDefaultReportPolicy(this, ReportPolicy.BATCH_AT_LAUNCH);
```

在线参数设置请上友盟网站更新，位于开发工具->参数配置

数据发送策略设置界面在统计分析->设置->发送策略。