百度

百度移动统计
SDK-For-SDK

三方 SDK 统计开发者手册 Android-1.6 版本

2016/2/26

百度移动统计SDK产品使用说明

修改记录

版本号	变更内容
1.2	（1）增加 mac 等 head 部分信息的加密功能
1.1	（1）增加版本号设置接口
1.0	（1）支持自定义事件时长统计功能 （2）支持 Fragment 页面统计接口（只支持顺序调用，不支持交叉调用） （3）支持自定义页面统计接口（只支持顺序调用，不支持交叉调用） （4）不支持 SDK 错误收集 （5）支持 SDK 统计的 log 调试开关（发布时请去除调用） （6）支持代码渠道设置接口（有效防止设置的渠道号丢失）

目录

前言 关于调试（注意第五条和第六条）

（1） 保证自定义事件的 ID 已经注册。如果您需要使用自定义事件，那么需要在 mtj.baidu.com 网站上自定义事件中定义您需要的自定义事件的 ID，然后按照自定义事件的规范写入代码。

（2） 保证您已经替换了 AppKey。Appkey 需要在 mtj.baidu.com 网站上注册应用，那么就会生成一个 Appkey，该 key 是该应用的唯一标识。如果没有该 key，那么，统计数据将不会展现。

（3） 如果没有数据怎么办？请确认以上五点都已经确认，网站统计数据会在 15 分钟左右展示出来，如果没有数据有可能是您的时区设置或者时间出现暂时异常，可以选择换台设备来测试，或者联系我们，具体参见第七章。

（4） 调试时可以设置 Log 调试开关，具体参照第五章第九节打开 Log 开关。

（5） 三方 SDK 在使用统计 SDK 时，需要引用 android_for_sdk_1*.jar。

（6） 设置渠道和 debug 的函数需要在其他接口调用之前初始化一次。详见 demo：SDKTestA 和 SDKTestB

（7） 设置版本号需要和设置渠道以及 debug 一样在调用其他统计接口前调用。

第一章 简介

百度移动统计 SDK(ANDROID)是百度官方推出的移动统计 SDK 在 ANDROID 平台上的版本（以下简称 SDK）。SDK 的发行版本中包括 JAR 包、文档、示例以及您正在阅读的用户手册。 以下使用<SDK_PATH>代表 SDK 包解压后的根目录。

- JAR 包：<SDK_PATH>/ANDROID_API.JAR，开发过程中导入 ANDROID 项目；
- 文档：<SDK_PATH>/ANDROID_DOC，提供开发帮助信息；
- 示例：<SDK_PATH>/ANDROID_DEMO，统计示例程序代码，帮助您迅速熟悉 SDK 的使用方式；
- 用户手册：本文档。

如果阅读完还没有领会，请导入我们的 DEMO 实例到您的工程中，参考使用。

如有其他问题可以参考网站的 FAQ，或者及时与我们联系（APPTONGJI@BAIDU.COM）

第二章 阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员。

第三章 版本支持

可运行于 Android 1.5（API Level 3）及以上版本。

第四章 集成使用

第一节 申请 APPKEY

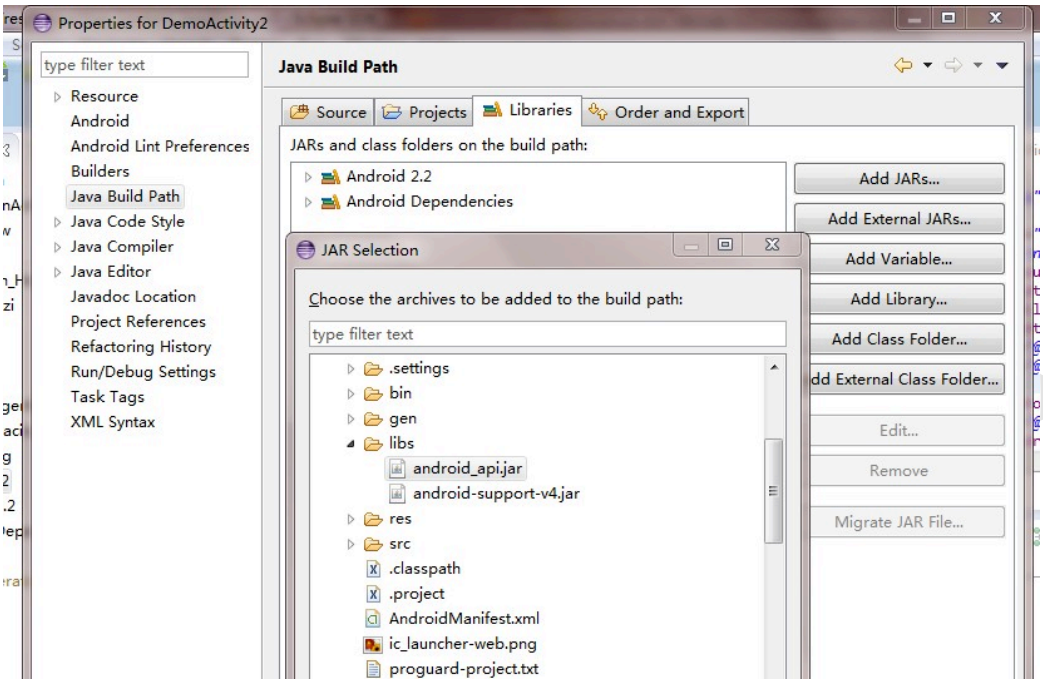
AppKey（BaiduMobAd_STAT_ID）在百度移动统计平台申请，用于标识您的应用程序。这个 key 是您的应用在我们的统计平台的唯一标识，请登录百度移动统计平台（登陆地址：http://mtj.baidu.com/web/welcome/login），然后点击“新增应用”按钮，完成菜单的填写后，点击“下一步”就可以看到生成的 APPKEY。同时请下载 SDK 工具包（包含 android_api.jar、android_doc、android_demo 等），请解压到本地使用。

第二节 添加 SDK 到项目

（注：以下提及的各个文件均可在<SDK_PATH>/android_demo中找到。）

➢ 将 SDK 中的 JAR 包导入您创建的 Android 项目，以下假设您已经创建了项目 A。

- 右键点击你的工程，然后选择“PROPERTIES”；
- 在工程属性对话框左边选择“JAVA BUILD PATH”；
- 在工程属性对话框主窗口选择“LIBRARIES”；
- 点击“ADD JARS”，添加<SDK_PATH>/ANDROID_API_1.0_FORSDK.JAR 和 NEED_LIB.JAR。



第三节 文档使用

- 若是在 ECLIPSE 或其他 IDE 下开发，可设置 JAR 包的 JAVADOC 路径为<SDK_PATH>/DOC，便于在 IDE 中获得即时的开发帮助信息；
- 也可以直接在浏览器中打开<SDK_PATH>/DOC/INDEX.HTML 查看该文档。

第四节 配置 AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.baidu.mobstat.demo" android:versionCode="1" android:
versionName="1.0">
    <uses-sdk android:minSdkVersion="4" />
    <!-- 必须申明的权限 -->
    <uses-permission
android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
    <uses-permission
android:name="android.permission.READ_PHONE_STATE"></uses-permission>
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

图1 AndroidManifest.xml

具体的信息参见图1。加亮文字标示了必须要配置的信息：

- 添加必要的权限
  - 必须权限：您必须申明的权限（若不申明将统计不到数据）
  - 可选权限：您可根据实际需求设置

第五章 代码集成

第一节 初始化

初始化的操作需要在接口调用之前调用，。渠道设置可以在代码中设置 3.3 版本新增了设置渠道的接口，可以让开发者通过参数 true 来保存渠道，从而杜绝网站渠道统计缺失的情况。
每个接口的最后一个参数为您在 mtj 上申请的 APPKEY 字符串。需要在调用基本接口前保证设置渠道的接口以及调试接口优先调用，初始化一次即可，具体参见 demo。

示例如下：
在调用其他接口前比如页面或者自定义事件，必须调用 initMTJForOnce 函数。最后面的字符串"898587128f"为 appkey，需要在 mtj 网站上申请，参见第四章第

一节。

```
public static boolean isInitMtj = false;
    public static void initMTJForOnce(Context context){
        if(!isInitMtj){
            isInitMtj = true;
            StatSDKService.setAppChannel(context, "channelB", true, "898587128f");
            StatSDKService.setDebugOn(true,"898587128f");
            StatSDKService.setAppVersionName("1.0", "898587128f");
        }

    }
```

(1) 设置渠道

```
/*
（1）如果第三个参数设置为true（防止渠道代码设置会丢失的情况），将会保存该渠道，每次设置都会更新保存的渠道
（2）appChannel是SDK的发布渠道，不需要在mtj网站上注册，直接填写就可以
（3）使用时直接使用下面的函数调用就可以，替换成您自己的真实渠道
*/
StatSDKService.setAppChannel(this, "RepleceWithYourChannel", true, ”appkey”);
```


(2)设置日志发送延迟时间

```
/*
* 设置启动时日志发送延时的秒数<br/> 单位为秒，大小为0s到30s之间<br/>
* 注：请在StatService.setSendLogStrategy之前调用，否则设置不起作用
* 如果设置的是发送策略是启动时发送，那么这个参数就会在发送前检查您设置的这个参数，表示延迟多少S发送。<br/>
* 这个参数的设置暂时只支持代码加入， 在您的首个启动的Activity中的onCreate函数中使用就可以。<br/>
*/
StatSDKService.setLogSenderDelayed(seconds, appkey)
```

第二节 Activity 页面统计

开发者需要在每个 Activity 的 onResume()和 onPause()中调用 SDK 提供的
StatSDKService.onResume(Context context，“appkey”)
StatSDKService.onPause (Context context，“appkey”)；
context 参数是当前 Activity 的引用,这里请不要将全局的 application context 传入。

注意：

如果开发者在父类 Activity 中添加了这两个函数调用，那么在子类 Activity 中就务必不要添加这两个函数调用。

注：为了使使用时长等数据更加准确，建议开发者在每个页面都添加统计接口，至少要在主界面添加。

说明：

- 确保在所有的 activity 中都调用 StatService.onResume(this) 和 StatService.onPause(this)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，否则会造成统计数据混乱(例如.使用 TabHost、TabActivity、ActivityGroup 时)。
- 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)。
- 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。

特别注意：

Activity 的页面的 onResume 以及 onPause 函数不能多个页面交叉调用如出现调用顺序：onResume------onResume 或者 onPause-------onPause，统计将会出问题。必须是顺序的 onResume---onPause---- onResume ---- onPause。

如果需要进行交叉调用，那么可以使用 onPageStart 和 onPageEnd 自定义页面统计函数来配合使用。onPageStart 和 onResume 是可以交叉调用的。

具体参见 demo 使用。

第三节 Fragment 页面统计

开发者需要在需要统计的 Fragment 的 onResume()和 onPause()中调用 SDK 提供的

```
StatSDKService.onResume(Context context，“appkey”)

StatSDKService.onPause (Context context，“appkey”);
```

context 参数是当前 Fragment 的引用,这里请不要将全局的 application context 或者 Activity 的 Context 传入。

注意：

如果开发者在父类 Fragment 中添加了这两个函数调用，那么在子类 Fragment 中就务必不要添加这两个函数调用。

注：建议开发者在需要统计的页面都添加统计接口，至少要在能够填充整个界面的 Fragment 中添加（具体参见 demo 实例）。

特别注意：

Fragment 的页面的 onResume 以及 onPause 函数不能多个页面交叉调用如出现调用顺序：onResume------onResume 或者 onPause-------onPause，统计将会出问题。必须是顺序的 onResume---onPause---- onResume ---- onPause。

如果需要进行交叉调用，那么可以使用 onPageStart 和 onPageEnd 自定义页

面统计函数来配合使用。onPageStart 和 onResume 是可以交叉调用的。

具体参见 demo 里面的 Fragment 可以和 Activity 的页面以及自定义的页面交叉调用。

如果 Fragment 以及包含 Fragment 的 Activity 都使用了统计，那么页面顺序是按照页面的结束时间来排序。例如 MainActivity 中有三个 Fragment 1、2、3. 那么依次访问的顺序为 MainActivity→Fragment1→2→3。由于我们是按照页面结束时间来记录，那么我们收集的日志顺序为 Fragment1→2→3→MainActivity。

特殊用法注意：

若有一个 Fragment 如 TestFragment 同时被初始化多次，那么如果调用 StatService.onResume()和 StatService.onPause()函数，我们只会记录 TestFragment 这个页面名称，而且在运行时由于被同时多次初始化，从而造成多次调用 onResume 和 onPause 函数，且顺序不正常，从而造成有页面丢失的 exception 异常抛出到 Log 日志，可以通过 eclipse 的 log 查看，请特别注意。

详细使用请打开 Log 开关，来测试您的统计行为。参见打开 Log 开关一节。

第四节 自定义页面统计

开发者需要在需要统计的View页面的onPageStart和onPageEnd()中调用SDK提供的


```
StatSDKService.onPageStart(Context context, Sring pageName，“appkey”);

StatSDKService.onPageEnd(Context context, Sring pageName，“appkey”);
```

context 参数是 Activity 的 Context 或者 application 的 Context 的引用，这里请注意同一个页面 pageName 必须一样，否则无法统计到。

注意：

如果开发者在父类自定义 View 中添加了这两个函数调用，那么在子类 View 中就务必不要添加这两个函数调用。

注：为了使使用时长等数据更加准确，建议开发者在每个页面都添加统计接口，至少要在能够填充整个界面的自定义 View 中添加。

特别注意：

自定义页面统计的 onPageStart 以及 onPageEnd 函数不能多个页面交叉调用如出现调用顺序：onPageStart------onPageStart 或者 onPageEnd-------onPageEnd，统计将会出问题。必须是顺序的 onPageStart---onPageEnd---- onPageStart----onPageEnd。具体参见 demo 里面的 Fragment 可以和 Fragment 的页面以及 Activity 的页面交叉调用。

如果自定义页面以及包含自定义页面的 Activity 或者 Fragment 都使用了统计，那么页面顺序是按照页面的结束时间来排序。例如 MainActivity 中有三个 Page 1、2、3.那么依次访问的顺序为 MainActivity→Page1→2→3。由于我们是按照页面结

束时间来记录，那么我们收集的日志顺序为 Page1→2→3→MainActivity。
详细参见 Demo 中自定义页面的使用。

第五节 自定义事件

1 / 自定义事件次数统计

void onEvent(Context context, String event_id, String label，“appkey”)
void onEvent(Context context, String event_id, String label, int acc，“appkey”)
参数： context 设备上下文
event_id 业务端注册的事件 id
label event id 下的各种事件添加的标签
aac 事件的发生次数，不指定时值为 1.

嵌入位置：任何地方。注意：时长统计函数中的 event_id 应该与 onEvent 函数中的 event_id 不同。

功能：统计开发者的自定义事件，自增计数。单纯的统计自定义事件的次数。这里的 event_id 需要开发者在 mtj 网站上添加自定义事件的 id 相同。例如：我们要统计更新的各个渠道的次数，那么我们可以定义 event_id 为 update，更新 1、更新 2 等各个渠道的更新表示 label。

那么在 mtj.baidu.com 网站上需要相同的 eventId: registered id

2 / 自定义事件时长统计

（1）使用事件开始和结束统计事件时长
void onEventStart(Context context, String event_id, String label, ”appkey”)
void onEventEnd(Context context, String event_id, String label, ”appkey”)

嵌入位置：任何地方，配对使用。注意：时长统计函数中的 event_id 应该与 onEvent 函数中的 event_id 不同。

功能：统计开发者的自定义事件时长统计

参数：
context	设备上下文
event_id	业务端注册的事件 id（需在 mtj 网站上注册）
label	事件添加的标签
appkey	mtj 上申请的 sdk-app 的唯一标识

也可以直接使用函数 onEventDuration，传入自己计算的事件时长：

（2）直接传入某事件的时长：

```
void onEventDuration(Context context, String event_id, String label, int milliseconds, ”appkey”)
```

注: 参数 milliseconds 的单位为 毫秒。该值是您自己计算出来的事件时长。

第六节 应用版本统计

请在 AndroidManifest.xml 文件中设置 versionName，SDK 会读取该版本号来作为你的 app 的版本。

第七节关于渠道设置

详见第五章第一节（设置渠道一段文字）以及第四章第四节例子。

在代码中设置渠道，请在调用其他接口前优先调用一次即可。

```
StatSDKService.setAppChannel(this, "RepleceWithYourChannel", true, ”appkey”);
```

第八节 获取 CUID 接口

String getCuid(Context context) （SDK3.1 新增）

1. 一般在 Activity.onCreate 函数中调用。
2. 功能：用于取得设备的 CUID 号。
3. 参数：context ---Activity 的上下文。
4. 调用函数：

```
StatSDKService.getCuid(this);
```

第九节 打开调试 Log 开关

调试时，可以在第一节初始化的地方增加调试 Log 的打开调用，这样你在接入 sdk 的时候可以通过 Log 的打印来看接入的情况。发布时去除该行代码即可。打开调试的代码：

```
StatSDKService.setDebugOn(true, ”appkey”);
```

第十节 设置发布的 SDK 的版本号

第一个参数是版本号，第二个参数是 appkey。具体注意事项参见第五章第一节初始化。

```
StatSDKService.setAppVersionName("1.0", "898587128f");
```

第六章 统计实例

Demo 工程已经嵌入了 SDK 提供的所有功能接口，开发者只需将 APPKey 更换成自己申请的 ID，然后运行，在网络畅通的情况下，统计数据就会被发送到业务端，开发者可以查看相应报表。Demo 工程中的接口都有详细注释，使用时请注意用法。

第七章 统计 SDK 包的目录

zip 包中解压会有以下目录结构：

- JAR 包：android_api_*_forsdk.jar；
- 文档：<SDK_PATH>/ANDROID_DOC，提供开发帮助信息；
- 示例：<SDK_PATH>/ANDROID_DEMO，统计示例程序代码，帮助您迅速熟悉 SDK 的使用方式；
- 用户手册：本文档。

第八章 混淆相关（1.6 新增）

SDK 已经做了相关必要混淆，开发者集成此 SDK 时无需再进行混淆，不当的混淆可能造成 SDK 功能不可用甚至崩溃。因此，请使用 keep 选项指定不对 SDK 进行混淆，具体在混淆配置文件 proguard-project.txt 中添加如下 keep 配置即可：

```
-keep class com.baidu.mtjstatsdk.** { *; }
```

第九章 联系我们

感谢您的阅读,如果有问题请 email 我们。 邮箱: apptongji@baidu.com