HiAnalytics Android SDK使用说明

| 日期 | 版本 | 修改描述 | 作者 |
| --- | --- | --- | --- |
| 2012-11-28 | 1.1 | 添加1.7章节内容 | 杨崴 00203606 |
| 2012-12-17 | 1.2 | 添加1.8章节内容 | 王哲wKF66773 |
| 2013-2-16 | 1.4 | 添加setMaxSessionIDTimeOut接口 | 王哲wKF66773 |
| 2013-3-5 | 1.5 | 添加1.3.2章节内容 | 王哲wKF66773 |
| 2013-3-24 | 1.6 | 将1.6章节“设置应用退出超时时长”变为“设置发送数据间隔时长” | 杨崴 00203606 |
|  |  |  |  |

本文档为应用开发者介绍HiAnalytics Android SDK 的植入流程

# SDK植入流程

## 导入hianalytics-sdk-android\_v2.0.jar（简称SDK）

* 1. 在项目的根目录新建lib文件夹
  2. 将hianalytics-sdk-android\_v2.0.jar文件拷贝至项目的lib文件夹下
  3. 选中lib文件夹中hianalytics-sdk-android\_v2.0.jar文件，单击右键—〉Build Path—〉Add to Build Path,
  4. 至此，您已成功导入HiAnalytics Android SDK，接下来请“修改配置文件 AndroidManifest.xml”。

## 修改配置文件AndroidManifest.xml

1. 添加用户权限，在AndroidManifest.xml中配置如下权限：
   * + android.permission.INTERNET 向HiAnalytics服务器发送您需要的用户分析数据
     + android.permission.ACCESS\_NETWORK\_STATE 检测网络状态
     + android.permission.ACCESS\_WIFI\_STATE 检测wifi状态
     + android.permission.READ\_PHONE\_STATE 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)
     + android.permission.INTERNET 这个权限主要是允许用户手机访问互联网
     + android.permission.READ\_PHONE\_STATE 这个权限是用户手机允许读取来电信息。
2. 添加应用程序的appkey和channel
   * + 开发者获得自己应用的appkey后（appkey由各应用与采集服务器侧约定，作为应用的唯一标识符），将此Appkey写到AndroidManifest.xml的meta-data里。（注意：不要改变'APPKEY'）
     + 应用程序的channel为应用程序将添加到的应用商店的名称，用户可以自定义名称。，将此channel写到AndroidManifest.xml的meta-data里。（注意：不要改变'CHANNEL'），以下是CHENNEL的取值范围（均为小写）

|  |  |
| --- | --- |
| 配置的名称 | 解释 |
| app\_market\_name | 表示智汇云推广渠道 |
| pre\_install | 表示预装推广渠道 |
| desktop | 表示动态桌面推广渠道 |

**注意：**

appkey**可以不填写，如果不填，默认会上报APK的包名。**

Channel**需要填写，如果不填，将上报Unknown**

此处标红为在AndroidManifest.xml文档中添加部分，以下为例子：

|  |
| --- |
| <manifest ……="">  <application ……="">  ……  <activity ……=""></activity>  <meta-data android:value="123" android:name="APPKEY">  </meta-data>  <meta-data android:value=" app\_market\_name " android:name="CHANNEL">  </meta-data>  </application>  <uses-sdk android:minsdkversion="8"></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.INTERNET"/>  <uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>  </manifest> |

## 添加代码

### Activity信息收集（可选）

Activity信息包括应用使用者进入了哪些您监控的Activity、停留时间和时长信息。

在您需要监控的每个Activity中，添加如下代码：

1. 添加 import com.hianalytics.android.HiAnalytics;
2. 在Activity的onResume方法中调用 HiAnalytics.onResume(context), 传入的参数为当前context的引用这里请不要将全局的application context传入。

|  |
| --- |
| public void onResume() {  super.onResume();  HiAnalytics.onResume(this);  } |

1. 在Activity的onPause方法中调用HiAnalytics.onPause(context), 只需传入一个参数当前activity的context.

|  |
| --- |
| public void onPause() {  super.onPause();  HiAnalytics.onPause(this);  } |

* 确保在所有需要监控的activity中都调用 HiAnalytics.onResume() 和HiAnalytics.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。
* 当调用onResume方法时，会判断距上次发送数时间间隔是否达到30秒钟（默认值，可配置），如达到，则上报数据
* 当调用onPause方法时，首先保存该界面退出时的activity信息，接着会判断距上次发送时间间隔是否达到30秒钟（默认值，可配置），如达到，则上报。如将该方法遗漏，那么将不会采集任何activity的信息。

**注意：**

onResume**和**onPause**接口都有上报数据的功能。**

**对于没有界面的应用程序，以上**onResume**和**onPause**接口可以不作为必选条件。**

### 应用退出时调用接口（必选）

为了确保应用启动后采集的数据都能够发送到服务器，需要在应用退出的时候，在退出事件处理中，添加如下代码：

* HiAnalytics.onReport(Context context)。

**注意：**

* + - 1. **强烈推荐在应用退出的时候（如点击“退出”按钮），调用一次，做到最后一次清空缓存发送数据的目的，最大概率地将之前的数据发送出去。**
      2. **该接口可以实时发送采集的数据（**onResume**和**onPause**为30秒间隔发送），不建议频繁调用**

### 自定义事件收集（可选）

在您需要监控的特定的事件发生的代码段中，可以调用如下方法，来收集事件：

HiAnalytics.onEvent(Context context, String key, String value);

* 参数context为当前context的引用
* Key、value为开发者自定义要收集的数据，如

HiAnalytics.onEvent(this, "click events", "click ad1");

表示一个点击事件，点击的是广告1（ad1）。

该接口最大的特点是提供了key/value对的输入风格，各应用可以自由灵活地添加自定义的key/value，尤其对于同一个应用中多个模块为在服务器上区别各自的上报信息，则可以将各自模块的名称最为key，上报内容作为value上送。

例如，某应用APK，集成了A模块与B模块，由于身处同样的context，在调用onResume和onPause方法上报到服务器上是无法区分A和B的数据，此时使用onEvent方法，A模块的key定义为”A”，B模块的key定义为”B”，上报内容在value中进行填写，这样上报的数据可以通过key进行筛选。

## 收集信息上报设置（可选）

在应用中收集到的信息包括：Activity信息、自定义事件信息

收集到的信息上报给服务器包括以下四种方式：

* **默认方式：**SDK将在下次应用启动时，主动将信息上报给服务器，这个不需要您额外添加代码
* **周期性上报：**将收集到的信息按固定的时间周期上报给服务器，调用如下方法：

public static void setTimer(Context context,int peroidSeconds)

参数context为当前context的引用

参数peroidHours为周期性上报的时间间隔，单位为秒，peroidSeconds设置为0，则取消周期性上报

**注意：请慎重使用此方法，当开启周期性上报后，即使应用进入后台，也会周期性发送消息，此上报方式适合常驻后台的应用。**

* **立即上报：**若您想将信息马上上报给服务器，则需调用如下方法 HiAnalytics.onReport(Context context);

参数context为当前context的引用

**注：该方法在应用退出时作为必选方法调用，意在应用退出前将所有数据再次清空发送。如果在非应用退出时的场景下，该方法还可以作为实时上报的接口。**

* **按收集到的信息大小上报：**若您想设置，当收集到的信息达到某个值时，马上将信息上报给服务器，则需调用如下方法

HiAnalytics. setMaxMsg(Long maxMsgSize);

参数maxMsgSize为能缓存的最大消息，单位为 byte，默认配置为500 byte.

当 超过此值 ，无论是否到达发送时间,都会马上将消息上报服务器,应设置为大于1000byte的值

## 设置收集信息的超时时间（可选）

* 对SDK收集的信息，没有及时上报给服务器的（如上报时网络问题等），在手机终端只保留一定时间，默认为24h，超过此时长的信息将被丢弃
* 应用开发者用如下方法，设置保留在用户收集中的信息的超时时长：

|  |
| --- |
| **protected** **void** onResume() {  **super**.onResume();  //例如15天则在参数中这样写  HiAnalytics.*setRecordExpireTimeOut*(15\*24);    HiAnalytics.*onResume*(**this**);  } |

public static void setRecordExpireTimeOut(Long recordExpireTimeOut)

* 设置的recordExpireTimeOut单位为h，必须大于24h，默认为24h。
* **注意**：此方法必须在应用mainActivity中被调用，且在方法HiAnalytics.onResume(context)之前。

## 设置发送数据间隔时长（可选）

* 数据建议不进行实时发送,以便减少网络交互。
* 用户每次调用onResume和onPause时，都会留下操作记录，第一次调用时，默认实时发送，然后客户端会记录发送时间，从第二次开始，一旦调用接口的当前时间与之前的发送时间之差达到默认时间（30秒），就会发送

|  |
| --- |
| **protected** **void** onResume() {  **super**.onResume();  HiAnalytics.*setSessionExpireTimeOut*(60L);    HiAnalytics.*onResume*(**this**);  } |

public static void setSessionExpireTimeOut(Long sessionExpireTimeOut)

* 设置的sessionExpireTimeOut单位为s，必须大于30s，默认为30s。
* **注意**：此方法必须在应用mainActivity中被调用，且在方法HiAnalytics.onResume(context)之前。

## sessionID和referID操作

类似网页的session，客户端应用我们也自定义了其会话周期，默认周期是30分钟，即当前后两次操作该应用的时间差大于30分钟，就认为是开启了另外一个会话周期。该功能不影响采集数据的上报，仅供服务器统计查询使用。

* 增加了Session\_id和refer\_id的接口。其中session\_id由SDK自动生成，refer\_id由客户端生成传给SDK处理，客户端自行决定需要何时处理session\_id和refer\_id，SDK提供以下调用接口：
* 初始化：

（客户端需要传递上下文context和referID）

SessionContext. init(this, referID);

* Set（不推荐使用，因为SDK程序会自行维护一套SessionID）：

（如果需要重新设置变量，调用如下方法，传递相应的变量值）

SessionContext. setReferID(referID);

SessionContext. setSessionID(sessionID);

* Get：

（获取当前状态下的变量的值）

SessionContext. getReferID();

SessionContext. getSessionID();

* 销毁：

（执行销毁方法时不仅销毁session\_id同时销毁refer\_id）

SessionContext. destroy(this);

(单独销毁refer\_ID)

SessionContext. destroyReferID(this);

* 设置sessionID超时时间：

HiAnalytics.setMaxSessionIDTimeOut(long maxSessionIDTimeOut)

* 此参数需要大于30，即超时时间不小于30分钟
* 初始化之后会将参数保存在：data/data/应用包名/shared\_prefs/hianalytics\_sessioncontext.com.huawei.hianalytics.xml这个文件里面，可以在此文件中查看sessionID和referID
* SDK默认采取客户端30分钟不操作即更新sessionID的处理机制，用户也可以根据需要灵活设置.

## 为SDK提供服务器的自适应

SDK默认链接的是商用的服务器地址，如果用户在自己的APK中进行SDK的调试，那么需要连接到测试服务器，待测试通过后，再作为商用发布。

因此，SDK提供了一个服务器的自适应机制，调试的时候连接测试环境，不影响商用环境。正式发布的时候再恢复到商用环境。方法如下：

* 用户在自己的APK代码中添加assets/config.properties文件
* 在文件中添加test\_version = true元素，表示当前版本使用服务器为测试服务器，供测试使用，
* 修改此元素值为false，或者assets/config.properties文件不存在，则为连接商用服务器，即APK正式发布的时候，请确保该属性为false或删除该文件