Android SDK 开发者使用文档

Cobub Razor

版本 1.0

作者:			主管:		
部门:	项目部	签名:	部门:	项目部	签名:
姓名:			姓名:		
电话:			电话:		

文件:	Cobub Razor 开发者使用文档(Android).doc	状态:	
日期:	2012-03-29	文件编号	

文档管理

更改记录

版本	状态	日期	负责人	更改原因
1.0	发布	2012-03-29	南京西桥科技有限公司	发布版本

目录

1 概要

1.1 文档目的

本文档的目的是讲解 Android SDK 的使用规则，方便开发者的使用。

1.2 适用范围

本文档是适用于 Android SDK 的使用者。

1.3 术语和缩写

Cobub Razor 移动应用统计分析

1.4 相关文档

无

2 基本设置指南

2.1 注册应用，获取 AppKey，下载 SDK

登陆账号后，到管理后台注册应用，填写应用的相关信息。App 建立成功后，可获得该 App 的 AppKey 以及最新的开发文档和 SDK 文件。

2.2 SDK 使用步骤

2.2.1 导入 com.wbkit1.0.jar（简称 SDK）

下载最新版本的 sdk 的压缩包，解压将其中的 com.wbkit1.0.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 com.wbkit1.0.jar 的路径，点击 OK，即导入成功。

2.2.2 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的 AppKey，将 AppKey 添加到 AndroidManifest.xml 的 meta-data 里。
  （注意：字符串必须为'*UMS_APPKEY*'）
- 添加权限 android.permission.INTERNET（必须）
  向服务器发送用户分析数据。
- 添加权限 android.permission.READ_PHONE_STSTE（必须）
  获取手机的相关状态信息
- 添加权限 android.permission.ACCESS_FINE_LOCATION（必须）
  获取当前用户的位置信息
- 添加权限 android.permission.ACCESS_WIFI_STATE（必须）
  访问 Wi-Fi 网络状态信息
- 添加权限 android.permission.GET_TASKS（必须）
  获取最近运行任务信息

- 添加权限 android.permission.WRITE_EXTERNAL_STORAGE（必须）
  向 sdcard 读写文件
- 添加权限 android.permission.READ_LOGS(必须)
  读取程序产生的错误日志
- 添加权限 android.permission.ACCESS_NETWORK_STATE(必须)
  访问 GSM 网络信息

2.2.3 添加代码

- 添加引用
  import com.wbtech.uns.UmsAgent;
- 注册 Activity、
  在每个 Activity 的 onResume 方法中调用 UmsAgent.onResume(Context),传入的参数为当前的 context 引用，这个方法将会自动地从 AndroidManifest.xml 文件里读取 AppKey。不要传递全局的 application context。

```
@Override
protected void onResume() {
    // TODO Auto-generated method stub
    super.onResume();
    UmsAgent.onResume(this);
}
```

在每个 Activity 的 onPause 方法中调用 UmsAgent.onPause(Context),参数为当前 activity 的 context.

```
@Override
protected void onPause() {
    // TODO Auto-generated method stub
    super.onPause();
    UmsAgent.onPause(this);
}
```

2.2.4 集成说明

- 建议在所有的 activity 中都调用 UmsAgent.onResume()和 UmsAgent.onPause()方法。如果在某些 activity 中不添加也可以，但会造成相应 Activity 的使用时间等相关信息统计不到。

2.2.5 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
- 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

3 高级基本设置指南

3.1 错误报告

SDK 可以帮您捕捉用户在使用应用过程中出现的异常退出（FC），并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改应用程序的 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
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 UmsAgent.onError(Context)

```
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.first);
    UmsAgent.onError(this);
```

后者需要开发者调用 UmsAgent.onError(Context,String)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查看错误报告。

3.2 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，例如您可以统计广告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：

3.2.1 简单事件

UmsAgent.onEvent(Context context,String event_id);
在您需要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 对应事件发送次数，变化趋势，例如广告点击，短信发送量等等。参数 context 为当前 context 的引用，event_id 为当前统计事件的 ID。
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 UmsAgent.onEvent(this,“ad_click“)通知服务器一个广告点击事件发生。

3.2.2 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：在应用程序中省份对应一个 event_id,每个城市对应一个 label，这样我们可以再生成的统计表中看到不同城市的比例。
UmsAgent.onEvent(Context context,String event_id,String label);

3.2.3 事件累计

对于程序中的某些可能被频繁触发的事件，开发者可以在程序中维护一个计数器，这样事件被多次触发只需要生成一个消息，这个消息包括该事件被触发的次数，这里我们重载了之前的两个接口：
UmsAgent.onEvent(Context context,String event_id,int acc);
UmsAgent.onEvent(Context context,String event_id,String label,int acc);
参数 acc 是对应事件被触发的次数。

3.2.4 分发渠道分析

不同的发布渠道，对应着不同的 AppKey，统计结果可以使您很好的了解有多少用户从联想乐园或者 Google Android market 下载到您的应用程序。

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，您只需修改 AndroidManifest.xml 中的 VersionCode，并把应用程序的 apk 文件上传到服务器。

4.2 添加权限

android.permission.WRITE_EXTERNAL_STORAGE

4.3 基本功能

在应用程序的入口 Activity 的 onCreate()方法中调用 UmsAgent.update(this)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.first);
    UmsAgent.update(this);
```

考虑到用户流量的限制，目前我们默认在 Wi-Fi 介入情况下才进行自动提醒。

4.4 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 文件上传到服务器。
UmsAgent.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，安装包会下载安装更新。（按照 version code 来检测是否需要更新）

5 数据发送

5.1 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。
- 实时发送
  应用程序每产生条消息时，就立即发送到服务器

5.2 设置发送模式

在入口 Activity 中调用 UmsAgent.setDefaultReportPolicy(Context,int)，参数 int 可取值 0 或者 1，其中 1 表示实时发送，0 表示启动时发送

6 使用在线配置功能

在程序的入口 Activity 的 onCreate()方法中调用
UmsAgent.updateOnlineConfig(Context)在程序启动时，我们将联网检测您的在线配置，将这下信息保存在本地。你也可以通过下面的函数读取您的自定义参数。
UmsAgent.updateOnlineConfig(Context context,String key) 其中 key 为在网站上编辑好的 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。