Android SDK Developer Documentation
Cobub Razor
Version 1.0

Author:		**Manager:**	
Dept.: Project Dept. Name: Tel:	Sign:	Dept.: Project Dept. Name: Tel:	Sign:
File: Cobub Razor Developer Documentation(Android).doc Date: 2012-03-29		Status: File NO.	

Document Management

Change Records

Version	Status	Date	Person in Charge	Change Reason
1.0	Publish	2012-03-29	Western Bridge Tech	Publish Version

Contents

1 Overview

1.1 Documentation Goal

The goal of this document is to explain usage rules of Android SDK for facilitating developers.

1.2 Application Scope

This document is for Android SDK developers.

1.3 Terms and Abbreviations

Cobub Razor	Mobile Applications Analytics

1.4 Relevant Documentation

None

2 Basic Setup Guide

2.1 Register App，Obtain AppKey，Download SDK

Register App and fill App info after login. Once App is created successfully, you could obtain AppKey, the latest development documentation and SDK.

2.2 SDK Usage Step

2.2.1 Import SDK

Download the latest release of SDK and unzip the zip file to a folder on your hard drive. Eclipse users right click your project root directory and select Properties —>Java Build Path —>libraries. Then click Add External JARs and select SDK path. Finally, click OK and complete import.

2.2.2 Configure AndroidManifest.xml

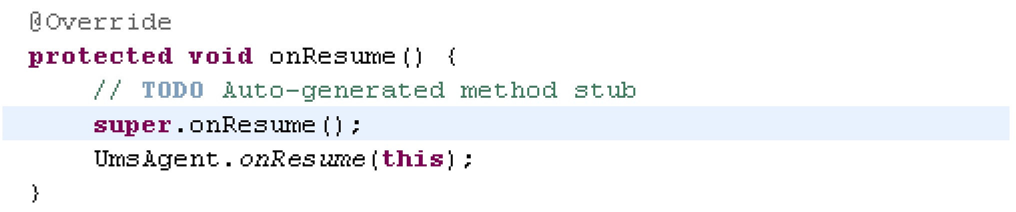
- Add App AppKey (necessary)
  Add AppKey obtained by App to meta-data of AndroidManifest.xml.
  (Note: String must be '*UMS_APPKEY*')
- Add permission android.permission.INTERNET (necessary)
  Send user analytic data to server.
- Add permission android.permission.READ_PHONE_STSTE (necessary)
  Obtain relevant status info of phone.
- Add permission android.permission.ACCESS_FINE_LOCATION (necessary)
  Obtain current user's location info.
- Add permission android.permission.ACCESS_WIFI_STATE (necessary)
  Visit Wi-Fi network status info.
- Add permission android.permission.GET_TASKS (necessary)
  Obtain recent running task info.
- Add permission android.permission.WRITE_EXTERNAL_STORAGE (necessary)
  Read and write file to sdcard.
- Add permission android.permission.READ_LOGS(necessary)

Read program error log.

- Add permission android.permission.ACCESS_NETWORK_STATE(necessary)
  Visit GSM network info.

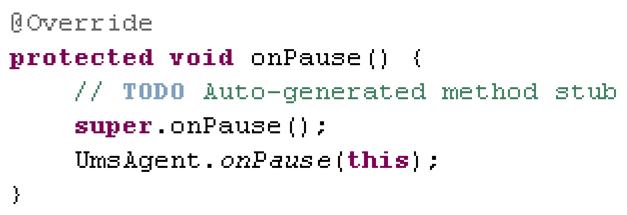
2.2.3 Add Code

- Add Reference
  import com.wbtech.uns.UmsAgent;
- Register Activity
  Call UmsAgent.onResume(Context) in onResume method of every Activity. The passed parameter is the reference of current context. This method will read AppKey from AndroidManifest.xml automatically. Do not pass global application context.



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

Call UmsAgent.onPause(Context) in onPause method of every Activity. Parameter is context of current context.



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

2.2.4 Integration Instructions

- We recommend you Calling UmsAgent.onResume() and UmsAgent.onPause() in all activities. If methods are not added in some activities, use time and other info of corresponding Activity will not be included in statistics.

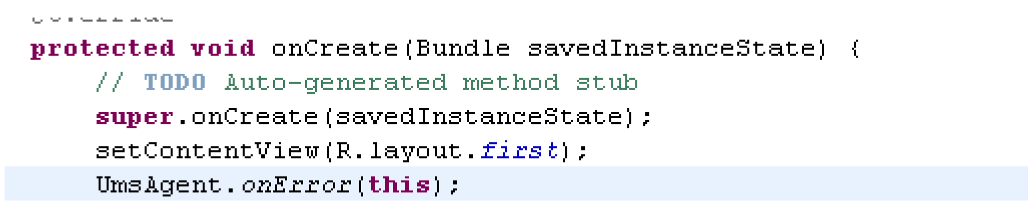
2.2.5 Note

- AppKey
  Check to make sure that AppKey has been corrently written to Androidmanifest.xml.
- Permission
  Check to make sure that all required permissions have been added.
- Use API
  Check to make sure that all Activities have called onResume and onPause.
- Check Network
  Check to make sure that test phone or simulator has successfully connected to network.

3 Advanced Setup Guide

3.1 Error Report

SDK could help you catch exit exception during App usage and send error report to server. Error report includes App version, OS version, device type and stacktrace of program exception. These data will help you modify App bug. We provide two ways to report error info. One is catched automatically by system and another is passed by developers.
For the former, you need to add android.permission.READ_LOGS permission in AndroidManifest.xml and call UmsAgent.onError(Context) in onCreate of Main Activity(App entry).



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

For the latter, developers need to call UmsAgent.onError(Context,String) and pass error info catched by their own to the second parameter.
You can view error report in product page.

3.2 Custom Event

Except basic analysis, we also support analysis of custom event. For example, you could count ad clicks or times video has been played. Here we provide several simple and common interfaces:

3.2.1 Simple Event

UmsAgent.onEvent(Context context,String event_id);
Call following method could send event logs to server in the code you need to send event report. It will analyze sending times and changing trends of corresponding event of event_id, such as ad clicks, message numbers and so on. Parameter of context is the reference of current context and event_id is the ID of current statistics event.
For example, monitor ad clicks of App. Event ID is " ad_click". Once ad is clicked, you need call UmsAgent.onEvent(this,"ad_click") in App to notify server that an ad click event has occurred.

3.2.2 Multi-Tag Event

Besides statistics on event occurrence times and changing trends of corresponding event of event_id, It can also do statistics on the proportion of specific label in event. The lable is current lable and event_id could not have spaces.
For example, province has a corresponding event_id and city has a corresponding label in App. Thus, we can see the proportion of different cities in generated statistics reports.
UmsAgent.onEvent(Context context,String event_id,String label);

3.2.3 Accumulative Event

For the events that may be triggered frequently in the program, developers can maintain a counter in the program. Thus, event that triggered many times just need generate one message. This message includes triggered times of this event. Here we overloaded two interfaces used before.
UmsAgent.onEvent(Context context,String event_id,int acc);
UmsAgent.onEvent(Context context,String event_id,String label,int acc);
Parameter acc is triggered times of corresponding event.

3.2.4 Channel Analytics

Different channels have different AppKeys. Analytics results could help you make a good understanding of how many users have downloaded your App from Lenovo park or Google Android market.

4 App Update

4.1 Upload APK

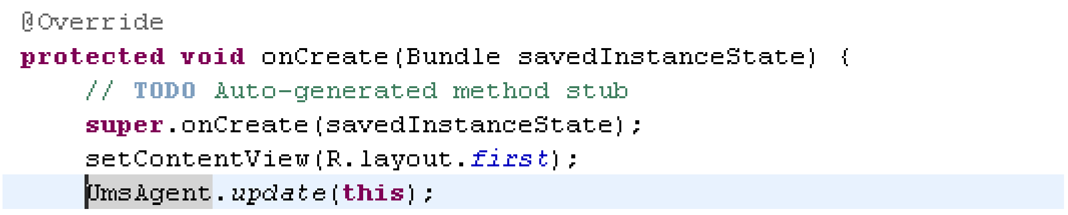
This function will help you push your new version App to users. You just need to modify VersionCode in AndroidManifest.xml and upload App APK to server.

4.2 Add Permission

android.permission.WRITE_EXTERNAL_STORAGE

4.3 Basic Function

Call UmsAgent.update(this) in onCreate() of App entry Activity.



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

In consideration of limitations of the user traffic, we now do automatic reminders under Wi-Fi intervention by default.

4.4 Mechanism Instructions

When updating App every time, you just need to modify VersionCode and upload App APK to server.
UmsAgent.update determines whether there is a new version App. It will notify user whether to ungrade SDK if there is a new App SDK. SDK wil upgrade after user choose to ungrade. (Detect whether to ungrade according to version code)

5 Data Sending

5.1 Model Explaination

- Start sending(recommended)
  App only sends a message to server when it starts. All messages produced during App will be sent to server on next time start. If the App starts without network, then the messages will be stored in local and App will try to send next time.
- Real-time Sending
  Once App produces a message, sending it to server immediately.

5.2 Set sending model

Call UmsAgent.setDefaultReportPolicy(Context,int) in entry Activity. Parameter int could be 0 or 1. 1 stands for real-time sending and 0 stands for start sending.

6 Online Configuration

Call UmsAgent.updateOnlineConfig(Context) in onCreate() of App entey Activity.
We will detect your online configuration on the network and store this info in local. You can also read your custom parameters by using following method.
UmsAgent.updateOnlineConfig(Context context,String key) The key is the key which has been edited on the web and return value is the corresponding value. A empty string will be returned if the

corresponding value is not read.