Flurry Analytics

Android SDK Instructions

SDK version 3.2.1
Updated: 06/21/2013

Welcome to Flurry Analytics!

This file contains:

---

1. Introduction

The Flurry Android Analytics Agent allows you to track the usage and behavior of your Android application on users' phones for viewing in the Flurry Analytics system. It is designed to be as easy as possible with a basic setup complete in few minutes.

---

2. Integration

To integrate Flurry Analytics into your Android application:

1. Add the FlurryAgent.jar file to your classpath.
    - If you're using Eclipse, modify your Java Build Path, and choose Add External JAR...
    - If you're using the SDK tools directly, drop it into your libs folder and the ant task will pick it up.

2. Configure AndroidManifest.xml:
  Required Permission:
      `android.permission.INTERNET`
      Required to send analytics data back to the flurry servers

  Optional Permission:
      `android.permission.ACCESS_COARSE_LOCATION` or
      `android.permission.ACCESS_FINE_LOCATION`
      If your application has location permissions, analytics will track where your application is being used. Without this, only country level location information will be available.
      To disable detailed location reporting even when your app has permission, call `FlurryAgent.setReportLocation(false)` before calling `FlurryAgent.onStartSession()`
      and no detailed location information will be sent.

Specify a versionName attribute in the manifest to have data reported under that version name.

3. Add calls to onStartSession and onEndSession
   - Insert a call to `FlurryAgent.onStartSession(Context, String)`, passing it a reference to a Context object (such as an Activity or Service), and your project's API key. We recommend using the onStart method of each Activity in your application, and passing the Activity (or Service) itself as the Context object - passing the global Application context is not recommended.

   - Insert a call to `FlurryAgent.onEndSession(Context)` when a session is complete. We recommend using the onStop method of each Activity in your application. Make sure to match up a call to onEndSession for each call of onStartSession, passing in the same Context object that was used to call onStartSession.

So long as there is any Context that has called onStartSession but not onEndSession, the session will be continued. Also, if a new Context calls onStartSession within 10 seconds of the last Context calling onEndSession, then the session will be resumed, instead of a new session being created. Session length, usage frequency, events and errors will continue to be tracked as part of the same session. This ensures that as a user transitions from one Activity to another in your application that they will not have a separate session tracked for each Activity, but will have a single session that spans many activities. If you want to track Activity usage, we recommend using onEvent, described below.

If you wish to change the window during which a session can be resumed, call `FlurryAgent.setContinueSessionMillis(long milliseconds)` before the first call to FlurryAgent.onStartSession.

You're done! That's all you need to do to begin receiving basic metric data.

---

3. Optional Features

You can use the following methods (during a session only) to report additional data:

```
FlurryAgent.logEvent(String eventId)
FlurryAgent.logEvent(String eventId, boolean timed)
FlurryAgent.logEvent(String eventId, Map<String, String> parameters)
FlurryAgent.logEvent(String eventId, Map<String, String> parameters, boolean
timed)
```

Use FlurryAgent.logEvent to track user events that happen during a session. You can track how many times each event occurs, what order events happen in, how long events are, as well as what the most common parameters are for each event. This can be useful for measuring how often users take various actions, or what sequences of actions they usually perform. Each project supports a maximum of 300 event names, and each event id, parameter key, and parameter value must be no more than 255 characters in length. Each event can have no more than 10 parameters. The parameter argument is optional, and may be null. Each session can log up to 200 events and up to 100 unique event names.

```
FlurryAgent.onError(String errorId, String message, Throwable exception)
```

Use onError to report errors that your application catches. Flurry will report the first 10 errors to occur in each session.

```
FlurryAgent.setCaptureUncaughtExceptions(false)
```

Used to allow/disallow Flurry SDK to report uncaught exceptions. The feature is enabled by default and if you would like to disable this behavior, this must be called before calling `onStartSession`.

`FlurryAgent.onPageView()`
Use onPageView to report page view count. You should call this method whenever a new page is shown to the user to increment the total count. Page view is tracked separately from events.

To disable FlurryAgent logging call `FlurryAgent.setLogEnabled(false)`.

`FlurryAgent.setUseHttps(boolean useHttps)`
Use setUseHttps to change the default session reporting request from HTTP to HTTPS.

Tracking Demographics
`FlurryAgent.setUserID(String);`
Use this to log the user's assigned ID or username in your system.

`FlurryAgent.setAge(int);`
Use this to log the user's age. Valid inputs are between 1 and 109.

`FlurryAgent.setGender(byte);`
Use this to log the user's gender. Valid inputs are `Constants.MALE` or `Constants.FEMALE`.

ProGuard
If you plan to run ProGuard on your APK before releasing your app, you will need to add the following to your "proguard.cfg" file:

```
-keep class com.flurry.** { *; }
-dontwarn com.flurry.**
-keepattributes *Annotation*,EnclosingMethod
```

Please let us know if you have any questions. If you need any help, just email androidsupport@flurry.com!

Cheers,
The Flurry Team
http://www.flurry.com
androidsupport@flurry.com