



Flurry Analytics

Android SDK Instructions

SDK version 4.2.0

Updated: 09/30/2014

Welcome to Flurry Analytics!

This file contains:

1. Introduction
2. Integration Instructions
3. Optional Features
4. FAQ

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.

Flurry Analytics uses the Android Advertising ID provided by Google Play Services and will check for and respect the user's ad tracking preference. For more information, please visit

<https://developer.android.com/google/play-services/id.html>

2. Integration

Flurry Analytics requires minimum Android API level 10. To integrate Flurry Analytics into your Android application:

Step 1. Add the FlurryAnalytics_4.2.0.jar file to your classpath.

Using Android Studio:

1. Add FlurryAnalytics4.2.0.jar to your project's libs folder.
2. Navigate to File > Project Structure > Module > Dependencies. Click the '+' button in the bottom of the 'Project Structure' popup to add dependencies. Select 'File dependency' and add libs/FlurryAnalytics4.2.0.jar.
3. Add Google Play Services library. Please follow instructions at <http://developer.android.com/google/play-services/setup.html#Setup>
4. Add v4 support library (or greater). Please follow instructions at

<http://developer.android.com/tools/support-library/features.html#v4>

Using Eclipse:

1. Add FlurryAnalytics4.2.0.jar to your project's libs folder. Right click on each JAR file and select Build Path > Add to Build Path.
2. Add Google Play Service library. Please follow instructions at <http://developer.android.com/google/play-services/setup.html#Setup>
3. Add v4/v7 Support Library. Please follow instructions at <https://developer.android.com/tools/support-library/setup.html#add-library>.

Step 2. Configure AndroidManifest.xml:

Required Permission:

```
android.permission.INTERNET
```

Required to send analytics data back to the flurry servers

Optional Permission (Highly Recommended):

```
android.permission.ACCESS_NETWORK_STATE
```

If your application has network state permissions, transmission of analytics data can be optimized.

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()`

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

Step 3. Add calls to onStartSession and onEndSession

1. 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.
2. 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 `logEvent`, 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

Report Additional Data

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.endTimedEvent(String eventId)
FlurryAgent.endTimedEvent(String eventId, Map<String, String> parameters)
```

Timed event can be logged using `FlurryAgent.logEvent`. Use `endTimedEvent` to end the timed event.

```
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.

```
FlurryAgent.setLogEvents(boolean logEvents)
```

Use `setLogEvents` to enable/disable the event logging.

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`.

```
FlurryAgent.setLocationCriteria(Criteria locationCriteria)
```

Use `setLocationCriteria` to set the `android.location.Criteria` for location information.

Additional Features

```
FlurryAgent.setUseHttps(boolean useHttps)
```

Use `setUseHttps` to change the session reporting request to use HTTP or HTTPS. HTTPS is the default.

```
FlurryAgent.getReleaseVersion()
```

Returns a string containing current Flurry SDK release version

```
FlurryAgent.setVersionName(String versionName)
```

Sets the version name of the app. This name will appear in the `http://dev.flurry.com` as a filtering option by version.

```
FlurryAgent.getAgentVersion()
```

Gets the version of the Flurry SDK.

```
FlurryAgent.setLocation(float latitude, float longitude)
```

Call this method to set the current location (used with geographical targeting).

```
FlurryAgent.setLogLevel(int logLevel)
```

Sets the log level of the internal Flurry SDK logging. Valid inputs are `Log.VERBOSE`, `Log.WARN` etc. Default log level is `Log.WARN`.

```
FlurryAgent.setLogEnabled(boolean isEnabled)
```

Use `setLogEnabled` to enable/disable internal Flurry SDK logging.

```
FlurryAgent.addOrigin(String originName, String originVersion)
```

```
FlurryAgent.addOrigin(String originName, String originVersion, Map<String, String> originParameters)
```

Use `addOrigin` to add the origin attribution. The event is identified by the `originName`, `originVersion` and `originParameters`. `OriginParameters` can be passed in as a `Map<String,String>` where the key is the parameter name, and the value is the value.

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
-keepclasseswithmembers class * {
public <init>(android.content.Context, android.util.AttributeSet, int);
}
```

```
# Google Play Services library
```

```
-keep class * extends java.util.ListResourceBundle {
```

```
    protected Object[][] getContents();
}

-keep public class
com.google.android.gms.common.internal.safeparcel.SafeParcelable {
    public static final *** NULL;
}

-keepnames @com.google.android.gms.common.annotation.KeepName class *
-keepclassmembernames class * {
    @com.google.android.gms.common.annotation.KeepName *;
}

-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}
```

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