FLURRY

Flurry Advertising

Android SDK Instructions

SDK version 3.3.4
Updated: 02/11/2014

Welcome to Flurry Advertising!

This README contains:

---

1. Introduction

Flurry Advertising is a flexible ad serving solution that allows you to easily manage the complete monetization of your mobile applications. Through integration with Flurry Analytics and various mobile ad networks, publishers can easily generate the maximum value from ad campaigns. Both AppCircle and AppSpot (direct and custom network) ad campaigns are created and managed in the same interface.

With AppSpot you will be able to:

1. Define your inventory
2. Traffic your ad campaigns
3. Track & optimize your performance

The Flurry SDK contains all the existing Flurry Analytics functionality as well as the new Advertising functionality. It is designed to be as easy as possible with a basic setup completed in under 5 minutes.

These instructions assume that you have already integrated Flurry Analytics into your application. If you have not done so, please refer to **Analytics-README** to get started.

For ad space setup and more information on Flurry advertising, please visit
http://support.flurry.com/index.php?title=Guides/s/Publishers
Please note, it is **required** that you create ad spaces before retrieving ads. Adspaces can be created on the Flurry Developer Portal or in the application code. If Adspaces are created in the code, they will appear

in the dev portal designated as “Determined by SDK” in the Adspace setup page.

---

2. Basic Integration

Flurry Ads requires minimum Android API level 10. To integrate Flurry Advertising into your Android application, just complete two steps:

Step 1. Include appropriate jar

Flurry Ads work on top of Flurry analytics. Add the FlurryAnalytics_3.3.1.jar and FlurryAds_3.3.1.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.

Step 2. Declare our Activity in your manifest

Declare the Activity “FlurryFullscreenTakeoverActivity” in your AndroidManifest.xml file:

```
<activity
android:name="com.flurry.android.FlurryFullscreenTakeoverActivity"
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode
|screenSize|smallestScreenSize"
</activity>
```

This is required to show interstitial ads.

Step 3. Requesting an ad

In order to serve ads please use methods `fetchAd`, `displayAd` and the interface `FlurryAdListener`.

Following is sample integration code to serve banner ad. To see integration of interstitial ad and for additional information, please visit flurry’s support site at http://support.flurry.com/index.php?title=Guides/f/Publishers/Code/Interstitials/Android.

```
public class Example extends Activity implements FlurryAdListener {
    RelativeLayout mBanner;
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.example);
        mBanner = (RelativeLayout)findViewById(R.id.banner);
        // allow us to get callbacks for ad events
        FlurryAds.setAdListener(this);
    }
    public void onStart() {
        super.onStart();
        FlurryAgent.onStartSession(this, mApiKey);
        // fetch and prepare ad for this ad space. won’t render one yet
        FlurryAds.fetchAd(this, mAdSpaceName, mBanner,
                               FlurryAdSize.BANNER_BOTTOM);
    }
```

```
        public void onStop() {
            super.onStop();
            FlurryAgent.onEndSession(this);
        }
        public void spaceDidReceiveAd(String adSpace) {
            // called when the ad has been prepared, ad can be displayed:
            FlurryAds.displayAd(this, mAdSpaceName, mBanner);
        }
    }
```

Note: please see **Analytics-README** for details on `FlurryAgent.onStartSession()`

Here are the details on the parameters to `fetchAd` and `displayAd`:

- **`Context context`** - We advise using the current Activity.
- **`String adSpace`** - Unique name of your ad placement, may defined on the Flurry website, but does not have to be
- **`ViewGroup viewSpace`** - ViewGroup inside the Context that you want the ad to reside in. We suggest using a `RelativeLayout`
- **`FlurryAdSize size`** - The size of ad to fetch, if this has not been set on the server for this ad space

"*`displayAd`*" must be called from the main thread.

Note that the same ad call is used for all ad formats and sizes including Banner, Takeover and Custom ad formats.
In order to specify the format and size, please visit Flurry's developer portal at
https://dev.flurry.com/appSpotSetupAdSpaces.do

---

3. Additional Ad Controls (Optional)

Request ads ahead of time

Add the following line to your app after the call to `FlurryAgent.onStartSession()` in your main Activity's `onStart` method. This method will make an asynchronous ad request to the Flurry ad server, which will respond with ads for ad spaces that have pre-caching enabled. This only needs to be called once. Please see the **Analytics-README** for details on `FlurryAgent.onStartSession()`.

```
FlurryAds.initializeAds(Context context);
```

Check if the ad is ready

The following method call returns a boolean value to let you know if an ad has been fetched and prepared for display. This method works best after calling fetch(), and then a true value indicates that you can make a display() call to display the ad.

```
FlurryAds.isAdReady(String adSpace);
```

Removing an ad

Flurry manages the lifecycle of the ads it displays, however, you can exercise finer control over display by choosing when to add and remove the ads from your app. To remove an ad just call

```
FlurryAds.removeAd(Context context, String adSpace, ViewGroup viewGroup);
```

The parameters are as follows:

- **Context context** - Context you want the ad to reside within. We advise using the current Activity.
- **String adSpace** - Unique name of your ad placement, may defined on the Flurry website, but does not have to be
- **ViewGroup viewGroup** - ViewGroup inside the Context that you want the ad to reside in

---

4. Enabling Ad Network Mediation (Optional)

Once you have your Ad Spaces set up, you will have the option of selecting 3rd party ad networks to serve ads into your Ad Spaces using the Flurry website (in addition to AppCircle, AppSpot and your own ads). You can change which ad networks serve ads at any time on the Flurry website, but in order to enable them you need to add the ad network SDKs into your application and configure them. The list of currently supported Ad Networks contains:

- Admob - SDK Version 6.2.1
- Millennial - SDK Version 5.0.1
- InMobi - SDK Version 3.7.0
- Mobclix - SDK Version 3.2.0

To implement an Ad Network you must perform the following steps:

1. Include the Ad Network Android SDK with your app and add it to the build path. Follow the instructions from the Ad network on how to complete this step.
2. Create the proper “activity” and “meta-data” tags in AndroidManifest.xml
    a. the first meta-data tag instructs the SDK about how to find the API_KEY
    b. the second meta-data tag instructs the SDK whether to request test ads
3. Add your API_KEY in strings.xml

Here is the example of implementing the Admob SDK into AppSpot:

AndroidManifest.xml

```
<activity android:name="com.google.ads.AdActivity"
       android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|
       uiMode|screenSize|smallestScreenSize"/>

<meta-data android:name="com.flurry.admob.MY_AD_UNIT_ID"
       android:value="@string/appSpot_admob_apikey"/>
<meta-data android:name="com.flurry.admob.test"
       android:value="true"/>
<meta-data android:name="com.flurry.admob.MYTEST_AD_DEVICE_ID"
```

```
      android:value="@string/appSpot_admob_testdevicekey"/>
<meta-data android:name="com.flurry.admob.test"
      android:value="true"/>
```

strings.xml

```
<string name="appSpot_admob_apikey">YOUR_API_KEY</string>
<string name="appSpot_admob_testdevicekey">(add device id that shows in
logcat</string>

Note: admob.test value needs to be set to true above. This will enable admob
test ads to show up in the emulator and on a device.
```

Here is the example of implementing the Inmobi SDK into AppSpot:

AndroidManifest.xml

```
<activity  android:name="com.inmobi.androidsdk.IMBrowserActivity”
      android:configChanges=
          "keyboardHidden|orientation|keyboard|smallestScreenSize|screenSize"
      android:hardwareAccelerated="true" />

<meta-data android:name="com.flurry.imobi.APP_ID"
      android:value="@string/appSpot_inmobi_apikey"/>
```

strings.xml

```
<string name="appSpot_inmobi_apikey">YOUR_INMOBI_API_KEY</string>
```

---

5. Request Configuration (Optional)

There are a number of configuration parameters that you can use to modify the behavior of your ad spaces.

Option 1. Set Location

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

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

Option 2. User Cookies

Add a call to identify any user specific information you want associated with the ad request:

```
FlurryAds.setUserCookies(Map<String, String> cookies)
```

To remove any user cookies call:

```
FlurryAds.clearUserCookies()
```

Option 3. Keyword Targeting

Add a call to specify keywords to be used when targeting ads:

```
FlurryAds.setTargetingKeywords(Map<String,String> keywords)
```

---

6. Implementing Ad Listener

To be notified of certain events during the full lifecycle of the Ad, you shall implement the FlurryAdListener interface and then call the setAdListener method to attach your implementation of FlurryAdListener to the Flurry SDK. You will need to implement the following callback methods:

- `onAdClicked(String adSpace)`
  - This method will be called when the user has clicked on the ad.
- `onAdClosed(String adSpace)`
  - This method will be called when the user dismisses the current Ad for the provided Ad Space name.This is applicable for fullscreen ads only.
- `onAdOpened(String adSpace)`
  - This method will be called when the user has opened the ad.
- `onApplicationExit(String adSpace)`
  - This method will be called when the user is leaving the application after following events associated with the current Ad in the provided Ad Space name.
- `onRendered(String adSpace)`
  - This method will be called when ad is successfully rendered.
- `onRenderFailed(String adSpace)`
  - This method will be called when ad was fetched from the ad network but was not successfully rendered.
- `shouldDisplayAd(String adSpace, FlurryAdType type)`
  - This method will be called if there is an ad available to display for that adSpace. Return value is true if the ad should be displayed; false if not.
- `spaceDidReceiveAd(String myAdSpaceName)`
  - This method will be called when the ad has been received from the server
- `spaceDidFailToReceiveAd(String myAdSpaceName)`
  - This method will be called when the ad request was made but no ads were retrieved.
- `onVideoCompleted(String adSpace)`
  - In case the ad served is a video clip, this method indicates the user completed watching the video.

Example usage:

```
public class MyAdListener implements FlurryAdListener {

      @Override
      public boolean shouldDisplayAd(String myAdSpaceName, FlurryAdType type)
      {
            return true;
      }
```

```
@Override
public void onAdClosed(String adSpaceName)
{
      // Handle the user closing the ad

}

@Override
public void onApplicationExit(String adSpaceName)
{
      // Handle the user exiting the application
}

@Override
public void onRendered(String adSpaceName)
{
      // Handle rendered
}

@Override
public void onRenderFailed(String adSpaceName)
{
      // Handle render failed
}

@Override
public void spaceDidReceiveAd(String adSpaceName)
{
      // Handle the adspace receiving the ad
}

@Override
public void spaceDidFailToReceiveAd(String adSpaceName)
{
      // Handle the adspace not receiving the ad
}

@Override
public void onAdClicked(String adSpaceName)
{
      // Handle the user clicking the ad
}

@Override
public void onAdOpened(String adSpaceName)
{
      // Handle the user opening the ad
```

```
        }
        @Override
        public void onVideoCompleted(String adSpace)
        {
              // called when a user finishes watching a video
        }
}

FlurryAdListener myAdListener = new MyAdListener(); // Create an instance
FlurryAds.setAdListener(myAdListener);        // Register the listener
```

---

7. Using ProGuard (Optional)

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);
}
```