AppCircle

Android SDK Instructions

SDK version 2.2
Updated: 8/13/2011

Welcome to Flurry AppCircle!

This README contains:

1. Introduction

Flurry AppCircle is an intelligent, cross-selling network for mobile applications. Using Flurry Analytics in combination with an advanced recommendation engine, AppCircle creates a market for developers looking to acquire users (Promoters) and application developers who want to earn incremental revenue from their applications (Publishers). AppCircle introduces a relevant app to the right user at the right time, across a network of participating applications. The service was designed to solve discovery, drive new user acquisition and increase revenue for application developers.

To become a publisher in the AppCircle network, allowing you to display recommended applications and begin earning more revenue from your users, you need to complete additional integration steps. The Flurry SDK contains all the existing Flurry Analytics functionality as well as the new AppCircle 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. AppCircle is contained in the same JAR file that contains the Analytics libraries so no additional libraries are required.

2. AppCircle Integration

Since you've already integrated Flurry Analytics into your application, adding AppCircle recommendations is simple. To integrate Flurry AppCircle into your Android application, complete the following steps:

Step 1. Turning on AppCircle

Add a call to enable AppCircle before onStartSession:

```
FlurryAgent.enableAppCircle();
FlurryAgent.onStartSession(Context, String)
```

Step 2. Getting the AppCircle module

AppCircle uses a different interface than Analytics. No matter how you decide to display recommendations you will need to retrieve this interface as follows:

```
AppCircle appCircle = FlurryAgent.getAppCircle();
```

Subsequent examples that use the `appCircle` variable assumes that it has been declared as above.

When you start the session, AppCircle related data will be downloaded to the device. You may need to wait a couple seconds on initial application launch before all the AppCircle data is downloaded. Once AppCircle data is downloaded, it is cached for use on subsequent application launches.

Step 3. Display Recommendations

There are 3 ways to display AppCircle recommendations:

- Banners
- Full screen catalog page
- Custom ad formats

Note: A **hook** is the representation for the placement of a banner. It allows you to track the performance of, and to set different ad policies for, the different locations or triggers in your app. Thus, it may be advisable to have a unique hook name for each ad location in your application. This will also allow control of the ads the user sees, e.g. restricting the ads in your main screen to only cross-promote your own apps, while your store front may contain recommendations for any apps.

Although it is not necessary, you can create this hook in the developer portal in the AppCircle > Revenue page. It will be auto-generated if our system has not seen this hook before (as identified by its unique hookName) when the report is sent to Flurry servers.

Display Option: Banner View

The most basic integration of AppCircle involves placing a banner on an View.

The following example shows the the creation and placement of a new ad banner.

```
// Example
ViewGroup viewGroup = (ViewGroup) findViewById(R.id.mainView);
View promoView = appCircle.getHook(context, "hookName", int mode);
if (promoView != null) {
      viewGroup.addView(promoView);
}
```

The `context` object refers to the instance of an android.context.Context, e.g. an Android Activity.

The `mode` is for defining whether the Banner is for portrait or landscape mode (`com.flurry.android.Constants.MODE_PORTRAIT` and `com.flurry.android.Constants.MODE_LANDSCAPE` respectively).

There may be instances when there are no ads available to be displayed, e.g when the application is first used without network access. A empty text message is displayed in the place of the banner. To change this message, use the method

```
appCircle.setDefaultNoAdsMessage(msg);
```

The Android Market screen is launched by default when a user clicks on a banner (assuming that the network is available). To change the behavior to open the Catalog first, use the following line of code then follow the instructions in the next section.

```
appCircle.launchCatalogOnBannerClicked(true);
```

Display Option: Catalog Integration

The catalog integration opens up a view containing a number of recommendations that users can browse. It is implemented as an Activity that will take over the full screen of the device.

To enable the Catalog Activity, the following XML snippet must be included within the <application> node in your AndroidManifest.xml configuration file.

```
<activity android:name="com.flurry.android.CatalogActivity"
      android:theme="@android:style/Theme.Translucent">
   <intent-filter>
            <action android:name="my.unique.intent.name"/>
            <category android:name="android.intent.category.DEFAULT"/>
   </intent-filter>
</activity>
```

and set the action name in intent filter *before* enabling AppCircle in your code.

```
FlurryAgent.setCatalogIntentName("my.unique.intent.name");
FlurryAgent.enableAppCircle();
```

Note that you should strive to create a unique action name in intent filter to avoid a name collision with other applications that may launch the Catalog as well.

This trigger is called by the Banner after it has been clicked by the user, if the launchCatalog flag is set. Alternatively, it can be programatically triggered by the following line of code

```
appCircle.openCatalog(Context context, String hookname);
```

where `hookname` is the String ID of the hook.

This will launch the Catalog, highlighting a random recommended application along with a recommendation list. The `context` object refers to the instance of an android.context.Context, e.g. an Android Activity.

When the Catalog Activity is launched, the Activity the Catalog was launched from will go into Paused mode. After the Catalog exits, that prior Activity will be resumed.

Display Option: Custom Ad Formats Integration

You can create custom ad formats using the Offer interface of AppCircle. The offer data returns the Offer ID, the app name, the app icon, the app price, and the app description. Flurry will not render any ads when the offer is called, *but the SDK will report an impression when obtaining an Offer for the ad*. It is up to you to use the offer data to render their own ads.

<u>1a) Retrieving a random Offer</u>
To do this call the following:

```
Offer offer = appCircle.getOffer(hookname); // String hookname
```

1b) Retrieving all available Offers

```
List<Offer> = appCircle.getAllOffers(hookname); // String hookname
```

The relevant methods for the returned classes are

```
Offer
    long getId()
    String getName()
    int getPrice()
    String getDescription()
    AdImage getImage()

AdImage
    int getWidth()
    int getHeight()
    String getMimeType()
    byte[] getImageData()
```

2) Accepting an Offer

To accept an offer, simply pass the Offer ID back the to SDK, e.g.

```
long offerId = acceptedOffer.getId();
appCircle.acceptOffer(context, offerId);
```

where `context` is the Activity Context. When an Offer is accepted, the Android Market will be launched.

3) Cleaning up an Offer

If an Offer is no longer used, e.g. when the user navigates away from the screen, the following method should be called to release the memory each Offer holds.

```
appCircle.removeOffers(offerIds); // List<Long> offerIds
```

---

3. AppCircle Rewards

Normally, when a user downloads and launches an AppCircle promoted app, the user is not rewarded for doing so. For apps that reward users with virtual currency or other incentives, the app can reward users when they download and launch an AppCircle promoted app. Once Flurry receives the data from the promoted app, Flurry will make a callback to the developer's server with the user's Android ID, reward currency, reward amount, and any other data that should be returned in the callback. The developer can use the Flurry callback to reward the user accordingly.

To integrate AppCircle Reward, the developer will need to do the following:

- Setup a server to receive the reward callback from Flurry
- Contact Flurry to setup the reward callback URL in the Flurry system
- Use offers, banners, or takeovers to render ads with custom reward messages
- Set user cookies with reward information that should be returned in the callback
- Reward user when Flurry callback is received

If you need to additional tracking beyond the user's Android ID, you can create custom parameters that will be included in the callback requests to your servers. These custom parameters can be any

information you need, including the kind of reward, amount of reward or application username of the user who earned the reward.

To set a user cookie, use the following call.

```
appCircle.addUserCookie(String key, String value)
```

To clear all cookies, use the following method.

```
appCircle.clearUserCookies()
```

Note: since the key and value pairs are sent as part of a URL (using GET) there is a limit to the size and number of user cookie parameters. Also, these need only be set once per session (after `FlurryAgent.onStartSession()`).

---

4. FAQ

How much data does the Agent send each session?

All data uploaded by the Flurry Agent is sent in a compact binary format. The total amount of data can vary but in most cases it is around 2Kb per session.

All data downloaded by the Flurry Agent is also sent in a compact binary format. Since AppCircle involves the displaying of images the amount of data downloaded is significantly larger than the amount uploaded with the typical session involving about 10Kb of download.

How many recommendations can I display?

By default, 7 recommendations are cached locally on the device for fast loading. These recommendations will rotate through at random whenever a recommendation is displayed.

Recommendations are automatically refreshed at session start and when your app is resumed from the background.

Why does a downloaded app continue to display as an ad?

There are a few possibilities on why an ad continues to display even after you downloaded the app. Android pauses the app when the app closes so the same view is kept in memory on next launch. The banners on the same view are also preserved for the next launch.

Another possibility is that the app is using cached ads. On the app launch, new ads are fetched and cached. If ads are requested before new ads are not completely fetched, the previous set of cached ads are used to fulfill the ad requests.

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