



Flurry Advertising

Flurry Android Adapter for MoPub

Adapter version 5.1.0.r1

Updated: 02/02/2015

Mediate Flurry Ads through MoPub

To integrate Flurry as the Custom Native Network in the MoPub ad serving flow, you need the Custom Event Class code incorporated into your application in addition to the Flurry SDK. Three quick steps are necessary:

1. Integrate the Flurry SDK and Flurry adapter for MoPub code into your app
2. Configure Flurry's Ad space(s)
3. Configure MoPub to mediate Flurry

1. Integrate Flurry SDK and Flurry adapter for MoPub into your app

1.1 If your application is not yet using Flurry analytics, create a new application on Flurry's dev portal. After logging into <https://dev.flurry.com>, select the Applications tab and from the top right-hand corner select Add New Application. In case your application is already tracked by Flurry, you can download the latest SDK from the adjacent top right-hand link.



1.2 Download the [Flurry Android SDK](#). Record the API Key found on the download page. This will identify your app in the Flurry system.

1.3 Add the Google Play Services SDK to your project. This is required for Android Advertising ID support. See <http://developer.android.com/google/play-services/setup.html>.

1.4 Add the jar files from the Flurry SDK (FlurryAndroidAnalytics-5.x.x.jar and FlurryAndroidAds-5.x.x.jar) to your project. Configure the build path of the project to include the jar files.

1.5 Add the Flurry MoPub adapter classes (found in the com.mopub.mobileads & nativeads package) to your project.

STEP 1

CustomEvent class integration steps are as follows:

I. FlurryCustomEventBanner and FlurryCustomEventInterstitial

Create com.mopub.mobileads package inside your project where you intend to implement the MoPub mediation for Banner and Interstitial.

II. FlurryCustomEventNative

FlurryCustomEventNative is new class introduced in 5.1.0.r1 and provides support for native ads. Create com.mopub.nativeads package inside your project where you intend to implement the MoPub mediation for Flurry Native ads.

STEP 2

Ensure that com.mopub.nativeads and com.mopub.mobileads packages are available in your App where you intend to integrate MoPub mediation, if not please create them.

Place following classes in com.mopub.mobileads package:

- I. FlurryCustomEventBanner
- II. FlurryCustomEventInterstitial
- III. FlurryAgentWrapper

Place following classes in com.mopub.nativeads package:

- I. FlurryCustomEventNative
- II. FlurryForwardingNativeAd

STEP 3

A. Flurry Native Ads integration via MoPub is similar to the steps described [here](#):

- I. Create an XML layout for your native ads
- II. Define where ads should be placed within your feed
- III. Create a MoPubAdAdapter to wrap your existing Adapter subclass and begin loading ads.

The broad range of assets that can be used in the ViewBinder are again similar to MoPub:

```
ViewBinder viewBinder = new ViewBinder.Builder(R.layout.native_ad_layout)
    .mainImageId(R.id.native_ad_main_image)
```

```
.iconImageId(R.id.native_ad_icon_image)
.titleId(R.id.native_ad_title)
.textId(R.id.native_ad_text)
```

Sample code from [MoPub](#)

```
private ListView mListView;
private MoPubAdAdapter mAdapter;
private RequestParameters myRequestParameters;
private static final String MY_AD_UNIT_ID = "myAdUnitId";

@Override
public void onCreate(Bundle savedInstanceState) {
    // Set up your adapter as usual.
    Adapter myAdapter;

    // Set up a ViewBinder and MoPubNativeAdRenderer as above.
    ViewBinder viewBinder = new
ViewBinder.Builder(R.layout.native_ad_layout)
        .mainImageId(R.id.native_ad_main_image)
        .iconImageId(R.id.native_ad_icon_image)
        .titleId(R.id.native_ad_title)
        .textId(R.id.native_ad_text)
        .build();

    // Set up the positioning behavior your ads should have.
    MoPubNativeAdPositioning.MoPubServerPositioning adPositioning =
        MoPubNativeAdPositioning.serverPositioning();
    MoPubNativeAdRenderer adRenderer = new
MoPubNativeAdRenderer(viewBinder);

    // Set up the MoPubAdAdapter
    mAdapter = new MoPubAdAdapter(this, myAdapter, myAdPositioning);
    mAdapter.registerAdRenderer(adRenderer);

    myListView.setAdapter(mAdapter);
```

```

    }

    @Override
    public void onResume() {
        // Set up your request parameters
        myRequestParameters = RequestParameters.Builder()
            .keywords("my targeting keywords")
            .build();

        // Request ads when the user returns to this activity.
        mAdAdapter.loadAds(MY_AD_UNIT_ID, myRequestParameters);
        super.onResume();
    }

```

You can selectively fetch desired assets in which scenario the `RequestParameters` can be set as follows:

```

final EnumSet<NativeAdAsset> desiredAssets = EnumSet.of(
    NativeAdAsset.TITLE,
    NativeAdAsset.TEXT,
    // You can choose not to pull the ICON_IMAGE
    // NativeAdAsset.ICON_IMAGE,
    NativeAdAsset.MAIN_IMAGE);
mRequestParameters = new RequestParameters.Builder()
    .keywords("my targeting keywords")
    .desiredAssets(desiredAssets)
    .build();

```

B. Flurry Interstitial Ads integration via MoPub follows the steps described [here](#)

```

private MoPubInterstitial mInterstitial;
mInterstitial = new MoPubInterstitial(this,
YOUR_INTERSTITIAL_AD_UNIT_ID_HERE);
// Remember that "this" refers to your current activity.

```

```
mInterstitial.setInterstitialAdListener(this);
```

C. Flurry Banner Ads integration via MoPub follows the steps described [here](#)

```
private MoPubView moPubView;  
moPubView = (MoPubView) findViewById(R.id.adview);  
moPubView.setAdUnitId("xxxxxxxxxx"); // Enter your Ad Unit ID from www.mopub.com  
moPubView.loadAd();
```

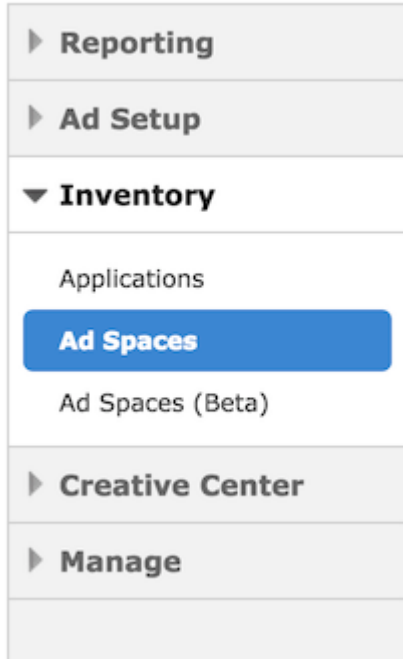
1.6 If you plan to run [ProGuard](#) on your APK before releasing your app, you will need to add the following to your “proguard.cfg” file:

```
# Preserve Flurry  
-keep class com.flurry.** { *; }  
-dontwarn com.flurry.**  
-keepattributes *Annotation*,EnclosingMethod  
-keepclasseswithmembers class * {  
public <init>(android.content.Context, android.util.AttributeSet, int);  
}  
  
# Preserve Flurry mediation classes for MoPub  
-keep public class com.mopub.mobileads.FlurryCustomEventBanner  
-keep public class com.mopub.mobileads.FlurryCustomEventInterstitial  
-keep public class com.mopub.mobileads.FlurryAgentWrapper  
-keep public class com.mopub.nativeads.FlurryCustomEventNative  
-keep public class com.mopub.nativeads.FlurryForwardingNativeAd  
  
# 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;  
}
```

2. Configure Flurry Ad space(s)

For each MoPub ad unit that you would like to mediate Flurry through, please create a matching ad space on Flurry's dev portal (<http://dev.flurry.com>).

Log into the developer portal and navigate to the **Publishers** tab. On the left-hand navigation bar select **Inventory** and then **Ad Spaces**.



With Ad Spaces selected you'll see an index of previously created ad spaces. To set up a new one, Click on the New Ad Space button on the top right. The Ad Space setup screen has four modules.

The Basic Setup section includes fields required to define the name, application, dimensions, placement and orientation of the ad space.

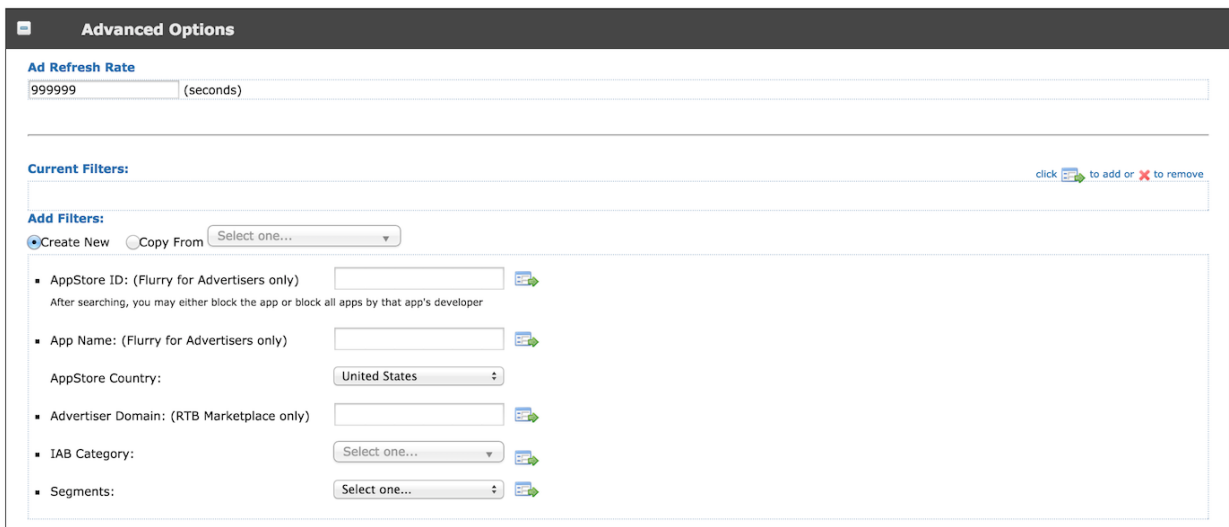
A screenshot of the 'Basic Setup' form in the Flurry developer portal. The form has a dark header bar with a small icon and the text 'Basic Setup'. Below the header, there are four rows of form fields. Each row has a label on the left, a red asterisk, a text input or dropdown, and a blue question mark icon on the right. The rows are: 'Name:' with the value 'MopubBanner'; 'Application:' with the value 'MoPubMediatesFlurry'; 'Placement:' with the value 'Banner Top'; and 'Dimensions:' with the value 'Standard Banner'.

The basic setup is all you need to start, you can click Save.

Please note that mediating Flurry through MoPub requires no additional Flurry related coding.

The Flurry Advertising code is already incorporated in the `com.mopub.mobileads` and `com.mopub.nativeads` package (added to your project in the previous step).

If you are integrating banner ads, and would like to streamline reconciliation of the impressions count, we recommend you turn off banner refresh on the Flurry side and let MoPub refresh the ads. ***This can be done by setting the refresh rate for the ad space to an unusually high number, like 9999 (instead of the default 30 seconds). This setting is found 4 section of the ad space setup - Advanced Options. Change the value for 'Ad Refresh Rate' setting.***





Advanced Options

Ad Refresh Rate









999999 (seconds)

Current Filters:

click  to add or  to remove

Add Filters:

☒ Create New ☐ Copy From Select one...

- AppStore ID: (Flurry for Advertisers only) 
After searching, you may either block the app or block all apps by that app's developer
- App Name: (Flurry for Advertisers only) 
- AppStore Country: United States 
- Advertiser Domain: (RTB Marketplace only) 
- IAB Category: Select one...  
- Segments: Select one...  

3. Configure MoPub to mediate Flurry

Flurry's custom events are implemented in accordance with instructions provided by MoPub (<https://github.com/mopub/mopub-android-sdk/wiki/Custom-Events>).

Complete integration steps are described in section 1.5 in this guide. After you incorporate the Flurry files into your project (as described in the Section 1 of this guide), you need to configure Flurry as the Custom Native Network into your mediation flow. Please follow instructions provided by MoPub with any of the Flurry custom events class noted below:
<https://dev.twitter.com/mopub/ad-networks/network-setup-custom-native>

Following Flurry Custom Event classes are available in the mediation package:

- `com.mopub.mobileads.FlurryCustomEventBanner` for Banner Ads
- `com.mopub.mobileads.FlurryCustomEventNative` for Flurry Native Ads
- `com.mopub.mobileads.FlurryCustomerEventInterstitial` for Interstitial Ads

An important setup to get this entire integration working is to configure an Ad Network and setup Flurry API key and Flurry Ad Space (as described in Section 1 and 2 above) and send them as server extras (Custom Event class Data) for the mediation to work. Here are some sample screenshots showing where these settings are configured on MoPub portal:

mopub

DashboardInventoryOrdersMarketplace**Networks**

Account Settings Reports Help

Overview / Custom Native Network Detail / Edit Network

Custom Native

Set up Custom Native Network

Edit Section

Special Instructions

- Custom Native Networks require the development of a Custom Event Class
- You must enter a Custom Class to enable ad serving for this type of network. Custom Event Class Data can be sent down in JSON format. Note: Custom Native Networks using a Custom Method implementation will not be supported in future MoPub SDKs.

Title

Flurry Network Android

Android

FlurryAndroidAdapter

RichMediaFS	Full	No Custom Content Entered	com.mopub.mobileads.FlurryCustomEventInterstitial	<pre>{"adSpaceName": "FLURRY_INTERSTITIAL_AD_SPACE", "apiKey": "FLURRY_INTERSTITIAL_API_KEY"}</pre>
Flurry Native Ad	Native	No Custom Content Entered	com.mopub.nativeads.FlurryCustomEventNative	<pre>{"apiKey": "FLURRY_NATIVE_API_KEY", "adSpaceName": "FLURRY_NATIVE_AD_SPACE"}</pre>