Android SDK v3.3 Quickstart

Android 2.3 and Higher

v1.5 - November 2016

Ampiri Android SDK Integration

This documentation guides you through a **basic** integration of the Ampiri SDK for Google Android. If you need additional information (customization, examples, logging, errors), please consult the How-To or SDK Integration Support.

Prerequisites

The following components should be added to your project.

Component	Version	Additional Information
Ampiri SDK	3.3.0	jCenter AAR download
Android	2.3	(API Version 9) or Higher/ Gingerbread
support-annotations	25.0.0	Android Support Annotations
support-v4	25.0.0	Android Support Library
joor	0.9.6	Java Object Oriented Reflection
*Google Play Services	9.6.1	Android API package
Ampiri Data	-	adUnitId - for each individual ad unit. Must be generated from Ampiri.com and placed in your app.

***NOTE**: We strongly recommend that you compile your app using Google Play services in order to use the Android Advertising ID and not the Device ID. Incorrect use of the Android Advertising ID can result in your submission to the Play Store being rejected.*

Automatic Setup: Change the *App module* build.gradle file

Add the following rows in the XML file:

```
repositories {
  maven { url "http://ampiri.bintray.com/maven" }
  flatDir {
       dirs 'libs'
  }
}

depedencies {
  compile 'com.ampiri.sdk:ampiri-sdk:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-adcolony:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-admob:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-unityads:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-applovin:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-chartboost:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-facebook:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-mopub:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-nativex:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-vungle:3.3.0'
  compile 'com.ampiri.sdk:ampiri-sdk-mediation-baidu:3.3.0'

  compile 'com.google.android.gms:play-services-ads:9.6.1'
}
```

Manual Setup: Include the .aar libraries

Step 1 - Download the SDK

Download and save the Ampiri SDK and 3rd-party ad network aar files under the *app module* libs folder.

Step 2 - Add Ad Network Libraries

Add the following rows to the *app module* build.gradle xml file

```
repositories {
  flatDir {
       dirs 'libs'
  }
}

dependencies {
  compile(name: 'ampiri-sdk', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-adcolony', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-admob', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-applovin', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-unityads', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-chartboost', version:'3.3.0', ext:
'aar')
  compile(name: 'ampiri-sdk-mediation-facebook', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-mopub', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-nativex', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-vungle', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mediation-baidu', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-mraid', version:'3.3.0', ext: 'aar')
  compile(name: 'ampiri-sdk-vast', version:'3.3.0', ext: 'aar')

  compile 'com.google.android.gms:play-services-ads:9.6.1'
```

```
    compile 'com.facebook.android:audience-network-sdk:4.16.1'
    compile 'com.mopub:mopub-sdk:4.9.0@aar', {
        transitive = true
    }
    compile 'com.google.code.gson:gson:2.7'
    compile 'org.jooq:joor:0.9.6'
}
```

NOTE: *This folder is most often found under project -> apps -> libs. In some cases, you may need to create it in Android Studio. For more information, please refer here.*

Step 3 - Update your Android Manifest XML file

1. Under the `<manifest>` element, add these permissions:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

2. In `project.properties` set `manifestmerger.enabled` to `true`.

Step 4 - Ad Types

4.1 Native

The following ad networks serve native advertisements.

Ad Network	Location	Version	Android Version
ampiri-sdk-mediation-admob	Google Mobile Ads	9.6.1, API 9	2.3 Gingerbread
ampiri -sdk-mediation-mopub	Mopub	4.9.0, API 9	2.3 Gingerbread
ampiri-sdk-mediation-facebook	Facebook Audience	4.16.1, API 11	3.0 Honeycomb
ampiri.sdk-mediation-applovin	AppLovin	6.3.2, API 9	2.3 Gingerbread
ampiri-sdk-mediation-baidu	Baidu	5.6, API 11	3.0 Honeycomb

Advertising space ID for Native testing:

```
"e5cc8e6d-d674-402a-aeca-eda7856bd7af"
```

1. Initialize the native ad by adding the following code to your *activity*.

NOTE: Native Ads are loaded via the `NativeAd` class, which has its own `Builder` class to customize it during creation.

```
NativeAd nativeAd = new NativeAd.Builder()
  .setAdUnitId("e5cc8e6d-d674-402a-aeca-eda7856bd7af")
  .setCallback(adListener)
  .build(this);
```

2. To show native ads, you can use two methods:

    - Create an ad view programmatically from the template and add it to the screen
    - Add `NativeAdView` view in the layout and bind loaded data to this view

3. The Ampiri SDK provides 3 types of templates for native ads:

    - `FeedCardNativeAdView` - Icon, title, description, star rating, and CTA button
    - `StoryCardNativeAdView` - Icon, image, title, description, star rating, and CTA button
    - `VideoCardNativeAdView` - Icon, image/video/carousel, title, description, star rating, and CTA button

***NOTE**: Every template has a label that clearly indicates it is an ad. For example: "Ad" or "Sponsored".*

To use one of the templates, add the selected template in the creation of the `NativeAd`:

```
.setAdViewBuilder(FeedCardNativeAdView.BUILDER);
```

4. And a banner layout. e.g.:

```
<FrameLayout
    android:id="@+id/ad_container"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="gone"/>
```

5. Display the banner once it has been downloaded by calling the `renderAdView` method:

```
adContainerView = (FrameLayout) view.findViewById(R.id.ad_container);

@Override
public void onAdLoaded() {
  adContainerView.setVisibility(View.VISIBLE);
  adContainerView.removeAllViews();
  adContainerView.addView(nativeAd.renderAdView());
}
```

4.1.1 In-feed

1. Initialize the native in-feed ad by adding the following code to your activity:

NOTE: *Native in-feed currently only supports ListView.*

```
MainAdapter adapter = new MainAdapter(this);
adAdapter = new StreamAdAdapter.Builder()
   .setAdapter(adapter)
   .setAdUnitId("e5cc8e6d-d674-402a-aeca-eda7856bd7af")
   .setViewBuilder(FeedCardNativeAdView.BUILDER)
   .build(this);
listview.setAdapter(adAdapter)
adAdapter.loadAd();
```

2. Native in-feed templates are explained in Step 3, section 4.1 Native

To use one of the templates, add the selected template in the creation of the NativeAd:

```
.setAdViewBuilder(FeedCardNativeAdView.BUILDER);
```

4.2 Videos

Ampiri direct demand does not serve video ads. In order to serve (or test) video ads on your app, you MUST register your app with an ad network that serves video ads.

The following ad networks serve video advertisements.

Ad Network	Location	Version	Android Version
ampiri-sdk-mediation-adcolony	AdColony	2.3.6, API 4	4.0 Ice Cream Sandwich
ampiri -sdk-mediation-unityads	UnityAds	2.0.4, API 9	2.3 Gingerbread
ampiri-sdk-mediation-chartboost	Chartboost	6.5.1, API 9	2.3 Gingerbread
ampiri.sdk-mediation-nativex	NativeX	5.5.8, API 11	3.0 Honeycomb
ampiri-sdk-mediation-vungle	Vungle	4.0.2, API 11	3.0 Honeycomb

Advertising space ID for Videos testing:

```
"032c0809-0335-4e98-8f8c-8e522f291d0f"
```

1. Add the following method signature to your *activity* (without a close button):

```
public VideoAd(@NonNull Activity activity,
               @NonNull String adUnitId,
               @Nullable AdEventCallback adEventCallback)
```

- Example:

```
VideoAd videoAd = new VideoAd(this, "032c0809-0335-4e98-8f8c-8e522f291d0f",
adListener);
videoAd.loadAd();
```

Alternative to step 1 - Not all ad networks support a close button for video. To enable this button add a boolean parameter in the method signature:

```
public VideoAd(@NonNull Activity activity,
               @NonNull String adUnitId,
               boolean closeButtonEnabled,
               @Nullable AdEventCallback adEventCallback)
```

Example:

```
VideoAd videoAd = new VideoAd(this, "032c0809-0335-4e98-8f8c-8e522f291d0f",
closeButtonEnabled, adListener);
```

2. Display the video after it has been downloaded by calling the `showAd()` method.

```
videoAd.showAd();
```

3. A notification for the completion of the download is available by calling the `isReady()` method

```
videoAd.isReady();
```

4. The following event methods should be called depending on the activity lifecycle:

```
@Override
protected void onPause() {
       super.onPause();
       videoAd.onActivityPaused();
}

@Override
protected void onResume() {
       super.onResume();
       videoAd.onActivityResumed();
}


@Override
protected void onDestroy() {
       super.onDestroy();
       videoAd.onActivityDestroyed();
}
```