

InMobi Android SDK 集成指南

简介

此文档将向您介绍如何集成轻量级的Android版InMobi SDK到您的应用当中，以便高效便捷地将您的流量变现。

文档结构如下：

1. 配置SDK
2. 集成原生广告
3. 集成插屏（全屏）、插屏视频广告
4. 集成激励视频广告
5. 集成横幅广告

1. 配置SDK

在正式集成SDK之前，您还需要进行一系列的配置从而让SDK能够正常工作。

注意：Android SDK 6系列仅支持Android系统4.0.3（API Level 15）及以上版本。我们强烈建议开发者使用Android Studio进行广告集成。

中国版SDK下载地址：https://dl.inmobi.com/cn/SDK/InMobi_Android_SDK.zip

国际版SDK下载地址：https://dl.inmobi.com/SDK/InMobi_Android_SDK.zip

Demo下载地址：<https://github.com/Bella085/InMobiNativeRebootAndroid>

添加SDK到您的工程中

方法一：通过JCenter获取最新的InMobi Android SDK

步骤一：在工程的build.gradle文件中添加：

```
allprojects {  
    repositories {  
        jcenter()  
    }  
}
```

步骤二：添加下列内容到工程application module的build.gradle中的dependencies下
compile 'com.inmobi.monetization:inmobi-ads:7.0.0'

步骤三：同步您的Gradle工程，确保SDK已完整下载

方法二：手动添加SDK到工程

步骤一：点击[此处](#)获取InMobi最新的Android SDK，下载解压后将SDK的jar文件放到工程中application module的libs文件夹下。

步骤二：添加下列内容到工程application module的build.gradle中的dependencies下
compile fileTree(dir: 'libs', include: ['*.jar'])

创建广告位

2

Create a Placement

Scroll to see more placements

FULL SCREEN AD

INTERSTITIAL

Interstitial ads are full screen ads that can be image-only, rich-media, or video. For maximum effect, use them at the natural breakpoints of your game.

RECOMMENDED FOR
ALL TYPES OF GAMING APPS

FULL SCREEN VIDEO AD WITH REWARD

REWARDED VIDEO

Rewarded Videos are landscape-locked, full-screen videos that reward the user for watching a video. These ads engage users and have a clear, opted-in user experience.

RECOMMENDED FOR
GAMING APPS WITH IN-APP PURCHASES AND VIRTUAL GOODS

IN STREAM AD

NATIVE CONTENT

Native content ads seamlessly blend in with your app's content, maximizing user engagement without compromising on user experience. You can choose and customize the layout that best matches the design of your app.

RECOMMENDED FOR
NEWS, COMMUNICATION OR UTILITY APPS

BANNER AD

BANNER

Banner ads consist of an image, either static or animated, along with a hyperlink that takes the user to a website or the relevant app store.

RECOMMENDED FOR
ALL TYPES OF APPS

Create Placement

Native Content 1

Native Content | ID: 1479541046414

Ad layout

Select a layout for your native content ad

0 Tags selected

Select tags that best describe your placement

Select Native Ad Layout

TILE

Sponsored

STREAM

Ad Title

★★★★★

Sponsored

Brief Description of the Ad goes here.

Button

FEED

Ad Title

★★★★★

Sponsored

Call to Action

SPLASH

Sponsored

CONFIGURATION

PLACEMENTS

Integration Instructions

Placements

Metadata

User Acquisition

Settings

Placement Name: Native Content 1

Ad Type: Native Content

Placement ID: 1470252471762

Tags Selected

None

在工程内添加必要的依赖包

1. Google Play Services(国内应用可省略)

在工程application module的build.gradle文件中添加下面的内容:

```
compile 'com.google.android.gms:play-services:8.4.0'
```

在工程的manifest文件中添加下面的内容:

```
<meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version" />
```

2. Picasso

在工程application module的build.gradle文件中添加下面的内容:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

3. Support Library, RecyclerView

```
compile 'com.android.support:appcompat-v7:24.0.0'
```

```
compile 'com.android.support:recyclerview-v7:24.0.0'
```

4. Multidex(非必需)

根据您的实际情况, 如果遇到64K reference limit / 65536问题, 可以通过enable multidex 和添加 multidex library依赖解决:

```
defaultConfig {
    applicationId "com.inmobi.samples"
    minSdkVersion 15
    targetSdkVersion 24
    versionCode 1
    versionName "1.0.0"
    multiDexEnabled true //启用multi-dex
}
```

```
compile 'com.android.support:multidex:1.0.1' //如果您的minSdkVersion 为20或更低
```

更改Manifest文件

1. 权限配置

你必须在你的应用manifest中添加下面必须的权限。

注意: 如果不添加如下的必需项你这边就拉取不到广告

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

除了必需的权限以外, 这里还强烈建议你这边添加 [ACCESS_COARSE_LOCATION](#) 或者 [ACCESS_FINE_LOCATION](#) 权限, 为了更好精准投放广告。

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

为了更好的体验广告, 这里也需要添加 [ACCESS_WIFI_STATE](#) 和 [CHANGE_WIFI_STATE](#) 权限。

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

Full Screen Activity

你必须要把这个全屏的activity加到你的应用中，为了某些全屏的视频广告。

```
<activity
    android:name="com.inmobi.rendering.InMobiAdActivity"
    android:configChanges="keyboardHidden|orientation|keyboard|
smallestScreenSize|screenSize|screenLayout"
    android:hardwareAccelerated="true"
    android:resizeableActivity="false"
    android:theme="@android:style/Theme.NoTitleBar"
    tools:ignore="UnusedAttribute" />
```

APP下载广告的属性

某些下载类的广告，为了检测安装，需要在manifest文件中添加如下广播接受器：

```
<receiver
android:name="com.inmobi.commons.core.utilities.uid.ImIdShareBroadCastRece
iver"
    android:enabled="true"
    android:exported="true"
    tools:ignore="ExportedReceiver">
    <intent-filter>
        <action android:name="com.inmobi.share.id"/>
    </intent-filter>
</receiver>
```

添加下载服务

```
<service
    android:name="com.inmobi.ads,ApkDownloader$ApkDownloadService"
    android:enabled="true">
</service>
```

需要写入外部存储权限

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

2. 硬件加速

添加此项将可以获取到运行在HTML5中的视频广告，您需要在application标签中添加 `hardwareAccelerated:true`

3. 适配Android N

我们这边的SDK 是适配Android N (API level 24)的，为了更好的适配，你需要声明 [InMobiAdActivity](#) 的参数如下：

```
<activity
    android:name="com.inmobi.rendering.InMobiAdActivity"
    android:configChanges="keyboardHidden|orientation|keyboard|
smallestScreenSize|screenSize|screenLayout"
    android:hardwareAccelerated="true"
```

```
android:resizeableActivity="false"
android:theme="@android:style/Theme.NoTitleBar"
tools:ignore="UnusedAttribute" />
```

混淆配置

我们建议将下列Proguard配置添加到工程的proguard-rules.pro文件当中：

```
-keepattributes SourceFile,LineNumberTable
-keep class com.inmobi.** { *; }
-keep public class com.google.android.gms.**
-dontwarn com.google.android.gms.**
-dontwarn com.squareup.picasso.**
-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient{
    public *;
}
-keep class
com.google.android.gms.ads.identifier.AdvertisingIdClient$Info{
    public *;
}
# skip the Picasso library classes
-keep class com.squareup.picasso.** {*; }
-dontwarn com.squareup.picasso.**
-dontwarn com.squareup.okhttp.**
# skip Moat classes
-keep class com.moat.** {*; }
-dontwarn com.moat.**
# skip AVID classes
-keep class com.integralads.avid.library.* {*; }
```

初始化SDK

请在您的Launcher Activity中初始化InMobi SDK，并将Activity和InMobi Account ID传入SDK(点击InMobi开发者后台右上角的邮箱地址处获取AccountID，或联系商务同事获取)

```
InMobiSdk.init(MainActivity.this, "InMobi Account ID");
InMobiSdk.init(MainActivity.this, "InMobi Account ID here", "Insert consentObject JSONObject
here"); （欧盟国家流量app必须使用此初始化方法）
```

注意！ SDK初始化需要在UI thread完成，否则将会导致初始化失败，影响变现效果。

consentObject - [consentObject](#) 是一个JSONObject 表示开发者同意授权给SDK的所有权限。

InMobi依赖开发者获取用户授权以符合GDPR约束。

GDPR 详情可参考 [here](#).

consentObject内容

Key	Type	Value
gdpr_consent_available	boolean	是否同意SDK收集用户数据，其他值都记为无效。默认false "true" - 同意 "false" - 不同意
gdpr	String	0或1:表示是否遵循GDPR (0 = No, 1 = Yes)

gdpr_consent_available这个key可以通过 [IM_GDPR_CONSENT_AVAILABLE](#) 获取

重要：

- 每个session对话中都需要提供consentObject. SDK不会永久保存授权信息，只会把consentObject存在内存中. 如果应用重启或崩溃consentObject就会丢失
- 如果运行中**consentObject**值有改变，可以按照下列方式更新授权信息：
[IMSdk.updateGDPRConsent\(<Insert consentObject dictionary here>\)](#)

[重要！] 如果您的应用能够获取用户标签、设备地理位置，请传给SDK以优化广告收入：

```
InMobiSdk.setLocation(Location location);  
InMobiSdk.setGender(InMobiSdk.Gender.MALE);  
InMobiSdk.setAgeGroup(InMobiSdk.AgeGroup.BETWEEN_21_AND_24);
```

2. 集成原生广告

InMobi原生广告，是指InMobi推出的一款可以将广告内容以符合开发者App风格的方式展现在应用中的一种广告形式。原生广告包含了视频和静态图的广告，集成方法都是一致的，无论是视频还是静态图都是通过View返回给开发者的，开发者只需要把这个view放在相应的位置就OK，而且这个view会自己处理展示和点击事件上报，开发者只需在相应的回调函数中处理相应的操作即可。在集成原生广告之前，请先联系InMobi的商务同事来获取原生广告的Placement ID。

步骤一：创建InMobiNative对象

//[重要！] 请保证应用运行时广告对象不会被意外回收，致使后续的广告行为失败。
`InMobiNative nativeAd = new InMobiNative(NativeAdsActivity.this,
YOUR_PLACEMENT_ID,nativeAdListener);`

参数说明

`NativeAdsActivity.this`: 从SDK6.0.0开始，创建原生广告对象时需提供当前Activity。旧的构造函数不需要Activity，在SDK6.0.0中已经废弃但仍可以使用，我们将不再对使用旧构造函数产生的任何问题进行支持。

`PlacementID` (Long型)：原生广告的PlacementID

`nativeAdListener`：您需要实现NativeAdListener监听原生广告的后续行为

```
new InMobiNative.NativeAdListener() {
```

```

//广告拉取成功的回调
@Override
public void onAdLoadSucceeded(final InMobiNative inMobiNative) {

}
//广告拉取失败
@Override
public void onAdLoadFailed(InMobiNative inMobiNative, InMobiAdRequestStatus
inMobiAdRequestStatus) {
    Log.e(TAG, "Failed to load ad. " + inMobiAdRequestStatus.getMessage());
}
//视频广告二级全屏页消失
@Override
public void onAdFullScreenDismissed(InMobiNative inMobiNative) {
    Log.e(TAG, "=onAdFullScreenDismissed. " );
}

//视频广告二级全屏也展示
@Override
public void onAdFullScreenDisplayed(InMobiNative inMobiNative) {
    Log.e(TAG, "=onAdFullScreenDisplayed. " );
}
//APP准备切换到后台
@Override
public void onUserWillLeaveApplication(InMobiNative inMobiNative) {
    Log.e(TAG, "=onUserWillLeaveApplication. " );
}
//广告曝光上报成功的回调
@Override
public void onAdImpressed(@NonNull InMobiNative inMobiNative) {
    Log.e(TAG, "=onAdImpressed. " );
}
//广告点击上报成功的回调
@Override
public void onAdClicked(@NonNull InMobiNative inMobiNative) {
    Log.e(TAG, "=onAdClicked. " );
}

//PreRoll视频广告播放完毕的回调
@Override
public void onMediaPlaybackComplete(@NonNull InMobiNative inMobiNative) {
    Log.e(TAG, "=onMediaPlaybackComplete. " );
}
//下载器下载进度回调
@Override
public void onAdStatusChanged(@NonNull InMobiNative inMobiNative) {
    Log.e(TAG, "=onAdStatusChanged.");
}
//用户点击跳过按钮回调
@Override
public void onUserSkippedMedia(@NonNull InMobiNative inMobiNative) {
    Log.e(TAG, "=onUserSkippedMedia.");
}
}

```

```
}
```

步骤二：启用Downloader

您需要在创建InMobi原生广告对象时启用下载器

```
nativeAd.setDownloaderEnabled(true);
```

每次下载器更改状态时将触发监听器的下载器回调onAdStatusChanged。

可以通过[getDownloadStatus\(\)](#)方法查询下载状态。可能的下载状态编码如下：

```
STATE_UNINITIALIZED = -2;
STATE_INITIALIZING = -1;
STATE_DOWNLOADING = 0;
STATE_DOWNLOADED = 1;
STATE_ERROR = 2;
```

当下载器下载状态为 STATE_DOWNLOADING = 0，您可以使用getDownloadProgress()方法访问下载进度，如展示进度条，可以使用此进度更新。

注意：每次下载百分比更新都会回调[onAdStatusChanged](#)。示例如下：

```
public void onAdStatusChanged(@NonNull InMobiNative nativeAd) {
    Log.d(TAG, "Ad Status has changed");
    Log.d("InMobi Downloader status is", "value: " +
nativeAd.getDownloader().getDownloadStatus());
    if (nativeAd.getDownloader().getDownloadStatus() ==
InMobiNative.Downloader.STATE_DOWNLOADING) {
        <<your progressbar>>.setProgress(nativeAd.getDownloader().getDownloadProgress());
        Log.d("The Download Progress is", "value: " +
nativeAd.getDownloader().getDownloadProgress());
    }
}
```

步骤三：加载原生广告

调用InMobiNative对象的load方法来加载原生广告

```
nativeAd.load();
```

调用之后，可在回调方法中获取广告加载结果。广告加载成功将执行onAdLoadSucceeded中的逻辑，广告加载失败时将执行onAdLoadFailed方法，您可以打印InMobiAdRequestStatus的信息查看具体内容。

步骤四：获取广告元素并填充广告

当广告加载成功时，您需要调用InMobiNative.getXX()获取原生广告相应的元素，解析后按需填充到App的相应位置中。

方法	描述
getAdTitle()	返回广告的Title
getAdDescription()	返回广告的描述
getAdIconUrl()	返回Icon的Url链接

方法	描述
getAdCtaText()	返回点击按钮的文字描述
getAdRating()	返回广告的评分
getAdLandingPageUrl()	返回广告的落地页链接
isAppDownload()	判断是否是下载类广告（可以用此判断是否展示下载提示框）
getPrimaryViewOfWidth(context, convertView, parent, width)	返回一个内容的View,广告的主体，广告的展示和点击这个view都全部内部处理了。开发者可以设置这个view的大小， FEED的广告类型中，请设置width=75,且view的容器设置1px*1px。
load()	拉取广告
isReady()	判断广告是否已经拉取到本地
reportAdClick()	上报点击，为了开发者其他的组件进行点击操作。
reportAdClickAndOpenLandingPage()	上报点击且打开落地页，默认是外部浏览器打开(如果开启了下载器，会触发下载)

2.1 Infeed 广告集成

2.1.1 定义你想展示广告的位置，例如广告展示在第4个位置：

```
private static final int AD_POSITION = 4;
```

2.1.2 创建Listener InMobiNative.NativeAdListener 如上：

```
private final class InMobiNativeAdListener implements InMobiNative.NativeAdListener{...}
```

2.1.3 创建InmobiNative实例：

```
InMobiNative InMobiNativeAd = new InMobiNative(
<<Activity Instance>>,
<<Your Placement ID>>,
<<InMobiNativeAdListener created in step 1>>);
```

2.1.4 拉取广告：

```
InMobiNativeAd.load();
```

2.1.5 例如你想把视频广告放在ListView里面：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    InMobiNative InMobiNativeAd = new InMobiNative(
```

```

        <<Activity Instance>>,
        <<Your Placement ID>>,
        <<InMobiNativeAdListener created in step 1>>);
        InMobiNativeAd.load();
        //Your App code here
    }

```

2.1.6 如果广告加载成功，你可以在onAdLoadSucceeded回调函数中获取广告元素：

```

@Override
public void onAdLoadSucceeded(@NonNull InMobiNative nativeAd) {
    AdFeedItem nativeAdFeedItem = new AdFeedItem(nativeAd);
    mFeedItems.add(AD_POSITION, nativeAdFeedItem);
    mFeedAdapter.notifyDataSetChanged();
    Log.d(TAG, "The ad is ready to be shown");
}

```

2.1.7 到目前为止，我们已经拿到了广告，并添加到数据源里面，后面就是展示广告，例如在ListView里面：

```

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if (itemViewType == VIEW_TYPE_CONTENT_FEED) {
        //Your app code
    } else {
        //获取原生广告元素
        final InMobiNative currentNativeAd = ((AdFeedItem) feedItem).mNativeAd;
        //获取主要content view
        AdViewHolder.primaryViewHolder = currentNativeAd.getPrimaryViewOfWidth(context,
        convertView, parent, parent.getWidth());

        Picasso.with(context).load(currentNativeAd.getAdIconUrl()).into(AdViewHolder.iconImageView);
        AdViewHolder.title.setText(currentNativeAd.getAdTitle());
        AdViewHolder.description.setText(currentNativeAd.getAdDescription());
        AdViewHolder.ctaButton.setText(currentNativeAd.getAdCtaText());
        return convertView;
    }
}

```

注意：请提供一个非null的 parent(ViewGroup)参数，主要是为了要从这个参数中获取广告展示view的ViewGroup.LayoutParams参数

2.1.8 刷新广告，为了最大的提升广告收入，开发者需要做好刷新机制，每次刷新，都需要重新拉取广告，移除之前的广告。

```

private void refreshAd() {
    Iterator<FeedItems> feedItemIterator = mFeedItems.iterator();
    while (feedItemIterator.hasNext()) {
        final FeedItem feedItem = feedItemIterator.next();
        if (feedItem instanceof AdFeedItem) {
            feedItemIterator.remove();
        }
    }
    // refresh feed
    mFeedAdapter.notifyDataSetChanged();
    // destroy InMobiNative object
}

```

```

InMobiNativeAd.destroy();
// create InMobiNative object again
InMobiNative InMobiNativeAd = new InMobiNative(
    <<Activity Instance>>,
    <<Your Placement ID>>,
    <<InMobiNativeAdListener created in step 1>>);
InMobiNativeAd.load();
}

```

注意：广告从你的数据列表中移除时，都需要destroy掉这个对象。

1. 一个实例创建的只是一个广告，一个广告有效展示一次和点击一次，多次展示和点击会记为无效
2. 如果想要多个广告，请创建响应个数的广告实例，建议创建为类变量，不要创建为函数变量。
3. 在activity的生命周期函数中调用响应的函数，`nativeAd.destroy()`; `nativeAd.pause()`; `nativeAd.resume()`;

2.2 Splash 广告集成

开屏广告集成，方法和Infeed的广告一致，有几点需要注意：

2.2.1 在主线程中，可能很忙，在onloadSucceed函数中不能及时的拿到广告返回，这时，可以通过`InMobiNativeAd.isReady()`方法去查看广告是否可以准备好展示。

2.2.2 【重要】当视频的第二页展示时，不要destroy或者刷新开屏广告，你需要检查第二详情页是否正在展示，如下：

```

@Override
public void onAdFullScreenDisplayed(InMobiNative nativeAd) {
    Log.d(TAG, "Full screen displayed");
    isSecondScreenDisplayed = YES;
}
public void dismissAd() {
    if(isSecondScreenDisplayed){
        Log.d(TAG, "DO NOT DISMISS THE AD WHILE THE SCREEN IS BEING DISPLAYED");
    }
    else
    {
        SplashAdView.setVisibility(View.GONE);
        InMobiNativeAd.destroy();
    }
}
}
/**

```

* 为了适应9：18的全面屏设备 开屏有3种推荐方式：

- * 1. 开屏的container设置垂直居中加载PrimaryView
 - * 2. `getPrimaryViewOfWidth`中的最后一个参数width值可以传：（素材的宽度*设备的高度/素材的高度）填充整个屏幕【素材是按照一定比例的，调整宽度，高度会按照比例自动调整】
 - * 3. 调整自己的logo高度 【如果开屏展示自家的logo】
- */

2.3 Preroll 广告集成

前贴广告集成，方法和Infeed的广告一致，有几点需要注意：

2.3.1 如果你需要关闭展示页面，你得在视频播放完的回调函数中操作：

```

@Override

```

```
public void onMediaPlaybackComplete(@NonNull InMobiNative nativeAd) {
    Log.d(TAG, "Media playback complete " + mPosition);
}
```

2.3.2 前贴视频广告关闭的方法如下：

```
public void dismissAd() {
    SplashAdView.setVisibility(View.GONE);
    InMobiNativeAd.destroy();
}
```

3. 集成插屏（全屏）、插屏视频广告

InMobi插屏（全屏）广告，包含静态插屏、旋转木马、普通插屏视频和激励性视频等多种广告样式。在开始集成插屏广告之前，请联系InMobi的商务同事获取对应类型的Placement ID

步骤一：创建InMobiInterstitial对象

```
InMobiInterstitial interstitialAd = new InMobiInterstitial(InterstitialAdsActivity.this, PlacementID,
mInterstitialAdListener);
```

参数说明

InterstitialAdsActivity.this: 从SDK6.0.0开始，创建插屏广告对象时，必须在对象的构造函数中传入Activity Context，禁止使用Application或服务Service的Context

PlacementID（Long型）：对应形式的插屏广告PlacementID

mInterstitialAdListener：您需要实现InterstitialAdListener2监听插屏广告的后续行为

注意：InMobiInterstitial对象必须创建在UI线程，其他和插屏广告相关的方法也要在UI线程中执行，否则有可能造成一些不可预估的问题。

主要的插屏广告回调如下：（注意，此处的接口名为InterstitialAdListener2）

```
public interface InterstitialAdListener2 {
    // status
    void onAdLoadFailed(InMobiInterstitial ad, InMobiAdRequestStatus status); // onAdReceived
    void onAdReceived(InMobiInterstitial ad); // onAdLoadSucceeded
    void onAdLoadSucceeded(InMobiInterstitial ad); // onAdRewardActionCompleted
    void onAdRewardActionCompleted(InMobiInterstitial ad, Map<Object, Object> rewards); //
    void onAdDisplayFailed(InMobiInterstitial ad);
}
```

步骤二：加载插屏广告

您需要直接调用 interstitialAd.load(); 方法来请求插屏广告。

如果请求成功，onAdReceived回调方法会被调用；之后onAdLoadSucceeded和onAdLoadFailed方法都有可能被执行。若请求失败，您可以打印InMobiAdRequestStatus对象中的信息来查看失败原因；若请求成功，您可以在任意时刻调用InMobiInterstitial的show方法来展示插屏广告。

注意：在插屏广告请求成功后，若再次调用load方法，将不会再重新请求一条广告。只有当SDK废弃了当前请求到的广告或广告超时时，调用load方法才会再会请求一条新的广告。

步骤三：展示插屏广告

当onAdLoadSucceeded回调中的逻辑被执行后，此时可以调用show()方法展示广告。

```
interstitialAd.show();
```

4. 集成激励视频广告

您需要使用激励视频类型的Placement ID并完整参考3中插屏广告的内容进行集成，当广告展示完毕时，SDK会调取下面的回调方法通知开发者给予用户奖励：

```
//当激励视频播放完毕时，会执行onAdRewardActionCompleted回调，此时可以发放奖励
void onAdRewardActionCompleted(InMobiInterstitial ad, Map<Object, Object> rewards);
```

5. 集成横幅广告

在集成Banner广告之前，请先联系InMobi的商务同事或自行在开发者后台创建Banner广告的Placement ID。

***Banner的高度是一定的，320*50或者640*100 不能传递别的尺寸**

方式一：通过XML布局文件创建并展示Banner广告

步骤一：请参考下列XML布局文件示例代码并添加InMobi Banner

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:ads="http://schemas.android.com/apk/lib/com.inmobi.ads"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context="com.inmobi.samples.BannerXMLActivity">

    <TextView android:text="@string/banner_xml_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <com.inmobi.ads.InMobiBanner
        android:layout_width="320dp"
        android:layout_height="50dp"
        android:id="@+id/banner"
        ads:placementId="输入你的Banner PlacementID"
        ads:refreshInterval="60"/>

</LinearLayout>
```

步骤二：引入banner对象

```
InMobiBanner bannerAd = findViewById(R.id.banner);
```

方式二：

步骤一：在代码中获取InMobiBanner对象

```
InMobiBanner bannerAd = new InMobiBanner(BannerAdsActivity.this, Banner PlacementID);
```

注意：InMobiBanner对象必须创建在UI线程，其他和Banner广告相关的方法也要在UI线程中执行，否则有可能造成一些不可预估的问题。

步骤二：当Banner广告对象创建之后，您可以参考下面的代码将广告添加到App的View中

```
RelativeLayout adContainer = (RelativeLayout) findViewById(R.id.ad_container);
```

//View宽高的对象为pixel

```
RelativeLayout.LayoutParams bannerLp = new RelativeLayout.LayoutParams(640, 100);  
bannerLp.addRule(RelativeLayout.ALIGN_PARENT_BOTTOM);  
bannerLp.addRule(RelativeLayout.CENTER_HORIZONTAL);  
adContainer.addView(bannerAd, bannerLayoutParams);
```

步骤三：加载Banner广告

当完成步骤三后，调用bannerAd.load()方法来加载Banner广告

```
bannerAd.load();
```

测试

可参考的 Debug 信息

调试的时候,在初始化 account ID 的地方可以开启调试开关:

```
InMobiSdk.setLogLevel(InMobiSdk.LogLevel.DEBUG);
```

```
InMobiSdk.setLogLevel(LogLevel.DEBUG);
```