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：

一： 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 7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：

```
http://sdk-china.s3.cn-north-1.amazonaws.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
```

1. 添加SDK到您的工程中

1.1 添加SDK的jar包

方法一：通过jCenter获取最新的InMobi Android SDK(海外版)

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.X.X'
```

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。

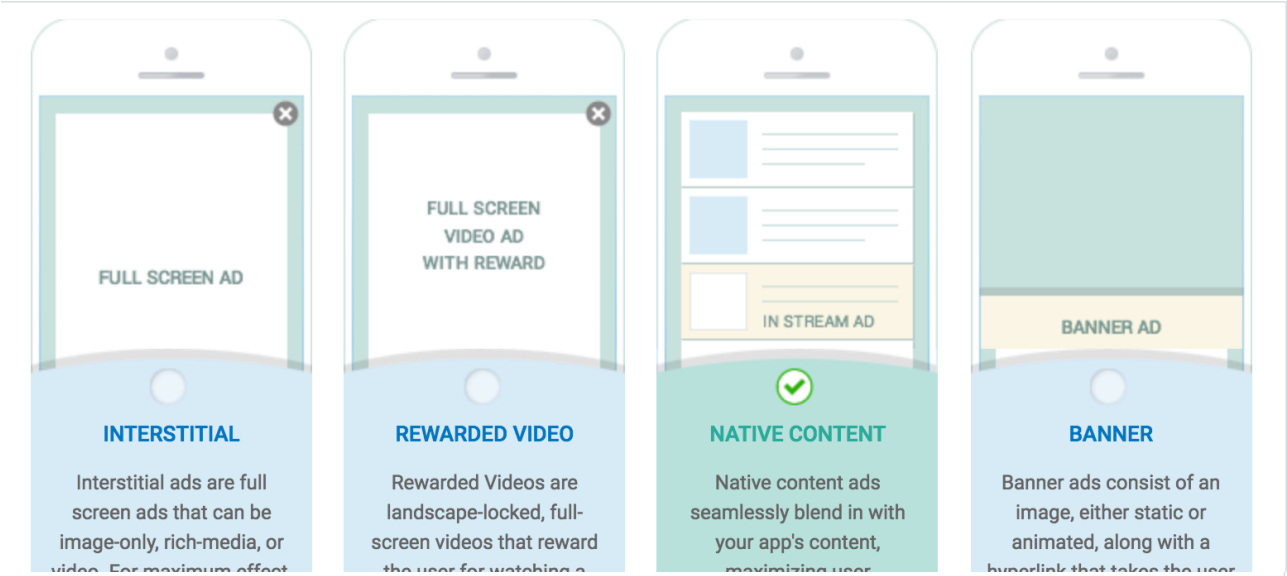
步骤二：添加下列内容到工程application module的build.gradle中的dependencies下

```
compile fileTree(dir: 'libs', include: ['*.jar'])
```

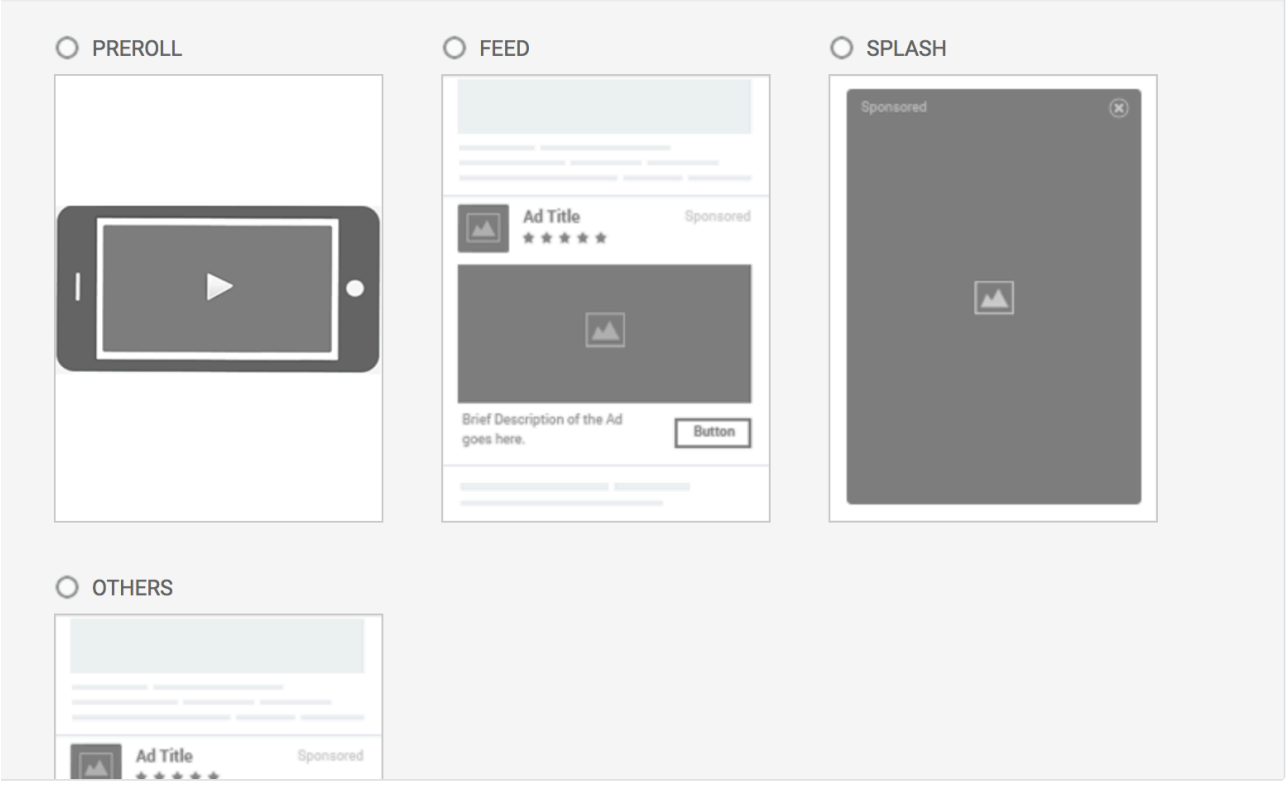
1.2 添加.so文件(重要)

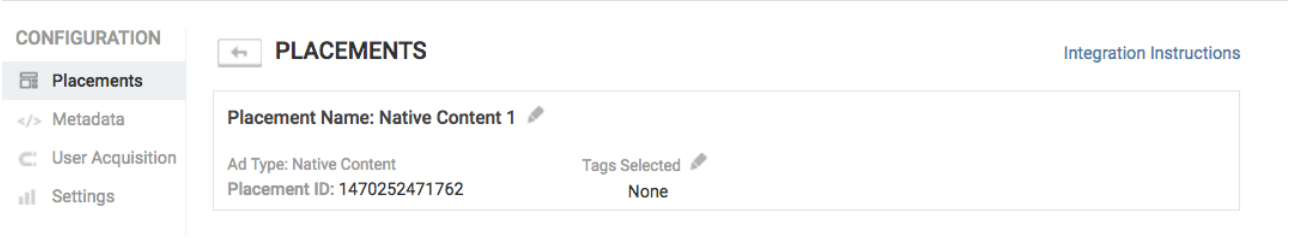
SDK ‘libs/jniLibs’ 目录下对应不同CPU架构下的库文件，建议拷贝该目录下所有到工程**jniLibs**文件夹下，如果考虑APK大小，至少要包含主流架构：`armeabi`、`armeabi-v7a`、`arm64-v8a`。

2. 创建广告位



Select Native Ad Layout





3. 在工程内添加必要的依赖包

3.1 GOOGLE PLAY SERVICES(国内应用可省略)

3.1.1 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-base:11.8.0'
compile 'com.google.android.gms:play-services-ads:11.8.0'
compile 'com.google.android.gms:play-services-location:11.8.0'
compile 'com.google.android.gms:play-services-plus:11.8.0'
```

3.1.2 同步一下工程， 确保依赖库成功添加

3.1.3 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" />
```

3.2 Picasso

3.2.1 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'
```

3.2.2 同步一下工程， 确保依赖库成功添加

注意⚠️：picasso 依赖库添加失败会导致插屏广告(interstitial ads)请求失败，进而影响变现。

3.3 Support Library

为了支持图片或者广告页面的滑动，需在工程application module的build.gradle文件中添加以下内容：

```
compile 'com.android.support:support-v4:27.1.0'
```

3.4 RecyclerView

为了支持图片或者广告的自由滑动，需在工程application module的build.gradle文件中添加以下内容：

```
compile 'com.android.support:recyclerview-v7:27.1.0'
```

注意⚠️：RecyclerView依赖库添加失败会导致插屏广告(interstitial ads)请求失败，进而影响变现。

3.5 Multidex(非必需)

根据您项目的实际情况，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

3.5.1 修改 defaultConfig, 使工程支持multidex:

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

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

```
compile 'com.android.support:multidex:1.0.1'
```

4. 更改Manifest文件

4.1 权限配置

4.1.1 务必在清单中加入以下必需权限，缺少此权限会引起变现失败

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

注意⚠️：针对第三条权限，若流量在国内必须添加，流量在海外则不必

4.1.2 除了上述必需权限，强烈建议您添加 ACCESS_COARSE_LOCATION 或 者 ACCESS_FINE_LOCATION 权限，该权限虽不是必需权限，却有助于广告精准投放，增加收益。

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

4.1.3 为了更精准定向，强烈建议您添加 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" />
```

4.1.4 如果您要请求富媒体广告，则需添加以下权限：

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

4.1.5 请务必在清单中配置InMobiAdActivity

```
<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" />
```

4.2 APP下载广告的属性

4.2.1 添加下载服务

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

4.2.2 需要写入外部存储权限

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

4.3 Android 7(Nougat 及以上)

从Android N起，需要进行新的更改，因为文件共享设计已更改。其他应用程序无法访问您的应用程序的文件，因此我们需要PackageInstaller应用程序来安装我们下载的APK文件。具体参考：https://developer.android.com/training/secure-file-sharing/setup-sharing

4.3.1 在 res/xml/provider.xml中添加：

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
  <external-path name="external_files" path="."/>
</paths>
```

4.3.2 在AndroidManifest.xml中添加：

```
<provider
  android:name="android.support.v4.content.FileProvider"
  android:authorities="{appPackageName}.fileprovider"
  android:exported="false"
  android:grantUriPermissions="true">
  <meta-data
     android:name="android.support.FILE_PROVIDER_PATHS"
     android:resource="@xml/provider" />
</provider>
```

4.3.3 对于Android Oreo(8及以上)，需要在AndroidManifest.xml中添加：

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

4.3.4 硬件加速允许HTML5视频广告的播放，您需要在application标签中添加以下代码：
hardwareAccelerated:true

注意⚠️：InMobi SDK兼容Android P(API28)——-Android平台的最新版本。Android P 以 上的应用程序默认情况下通过HTTPS进行连接。InMobi支持采用HTTPS，同时我们正在努力和所有的广告主以及监测公司对未达标的链接进行升级。在此过程中，一些广告主还没法做到完全使用https的连接，为避免对收入有影响，建议开发者可以同时支持HTTP和HTTPS连接。

5. 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.** { *; }
-dontwarn 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.** {*;}
```

6. 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

注意⚠️：

i. 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

ii. 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(locationObj);
InMobiSdk.setLocationWithCityStateCountry("city","state","country");
InMobiSdk.setGender(InMobiSdk.Gender.MALE); // or InMobiSdk.Gender.FEMALE
InMobiSdk.setAge(age);
InMobiSdk.setAgeGroup(InMobiSdk.AgeGroup.BELOW_18);
        // other enums: BETWEEN_18_AND_20, BETWEEN_21_AND_24,
        BETWEEN_25_AND_34
        // BETWEEN_35_AND_5, ABOVE_55
```

二. 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

```
public class NativeAdsActivity extends Activity {

        private final List<InMobiNative> mNativeAds = new ArrayList<>();
        InMobiNative nativeAd = new InMobiNative(NativeAdsActivity.this,
YOUR_PLACEMENT_ID, nativeAdeEvenListener);
        nativeAd.load();
        mNativeAds.add(nativeAd);
        ……
```

[重要！]: inmobiNative对象必需是强引用，否则可能被垃圾回收(garbage collection)，进而引起接受不到任何关于加载成功或失败的回调信息，和填充率低的问题

nativeAdeEvenListener 是抽象类NativeAdEventListener的实现类， 用于监听原生广告的后续行为

```
/**
* A listener for receiving notifications during the lifecycle of a Native ad.
*/
public abstract class NativeAdEventListener {

  public void onAdLoadSucceeded(InMobiNative ad) {}

   public void onAdLoadFailed(InMobiNative ad, InMobiAdRequestStatus requestStatus) {}

   public void onAdFullScreenDismissed(InMobiNative ad) {}

   public void onAdFullScreenWillDisplay(InMobiNative ad) {}

   public void onAdFullScreenDisplayed(InMobiNative ad) {}

   public void onUserWillLeaveApplication(InMobiNative ad) {}

   public void onAdImpressed(InMobiNative ad) {}

   public void onAdClicked(InMobiNative ad) {}

   public void onAdStatusChanged(InMobiNative nativeAd) {}

   public void onRequestPayloadCreated(byte[] requestPayload) {}

   public void onRequestPayloadCreationFailed(InMobiAdRequestStatus status) {}
}
```

VideoEventListener类可用于监听原生视频广告的后续行为，使用以下方式注册代码

```
nativeAd.setVideoEventListener (videoEventListener);
```

```
/**
* A listener for receiving notifications during the lifecycle of a Native ad.
*/
public abstract class VideoEventListener {

   public void onVideoCompleted(InMobiNative ad) {  }

   public void onVideoSkipped(InMobiNative ad) { }

   public void onAudioStateChanged(InMobiNative inMobiNative, boolean isMuted) {  }
}
```

步骤二：启用Downloader

在中国，由于缺少Google Play, 需要使用InMobi Downloader直接下载APK。只有流量在中国的开发者需要使用InMobi Downloader，它管理了广告主应用的下载和安装。

您需要在创建InMobi原生广告对象时启用下载器

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        InMobiNative nativeAd = new InMobiNative(
        <<Activity Instance>>,
        <<Your Placement ID>>,
        <<InMobiNativeAdListener>>);
        nativeAd.setDownloaderEnabled(true);
        nativeAd.load();
        //Your App code here
}
```

每次下载器更改状态时将回调监听器的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，您可以通过getDownloadProgress()方法获取下载进度，并使用该进度，更新进度条。

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("InMobi Downloader status is", "value: " +
nativeAd.getDownloader().getDownloadStatus());
        if (nativeAd.getDownloader().getDownloadStatus() ==
InMobiNative.Downloader.STATE_DOWNLOADING) {
        progressBar.setProgress(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
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）
getCustomAdContent()	获取广告部分内容（json）
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=1,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)

步骤五：区分视频广告

广告获取成功的回调中使用以下代码，获取标记当前广告是否为视频的标识值。

```
JSONObject customContent = inMobiNative.getCustomAdContent();
boolean isVideo = customContent.getBoolean("isVideo");
```

对接开屏广告时候，建议视频广告展示时长5秒

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：

```
@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
}
```

2.1.4 拉取广告：

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

2.1.5 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");
 }
```

2.1.6 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

2.1.7 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. 如果想要多个广告，请创建相应个数的广告实例，建议创建为类变量，不要创建为函数变量。
2. 在activity的生命周期函数中调用相应的函数，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 主线程可能很忙，在onloadSucceed函数中有时不能及时拿到广告返回，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 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();
   }
}
```

注意：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();
}
```

2.4 Lock Screen 锁屏广告集成

集成锁屏广告，如果您想要预拉取广告，并且不想绑定在任何activity上，您可在Appilcation的子类中创建native ad units(原生广告单元). 要变现的app, 锁屏位置上则可使用该native ad units(原生广告单元)

代码如下：

```
public class YourApplication extends Application {

    ...
    private InMobiAdRequest mInMobiAdRequest;
    private NativeAdRequestListenermListener;
    Private InMobiNative mNativeAd;
    ...
    public void onCreate() {

        mInMobiAdRequest = new
InMobiAdRequest.Builder(PlacementId.YOUR_PLACEMENT_ID)
                .build();
        mListener = new InMobiNative.NativeAdRequestListener() {
            @Override
            public void onAdRequestCompleted(InMobiAdRequestStatus status, InMobiNative
nativeAd) {
                if (status.getStatusCode() == NO_ERROR && nativeAd != null) {
                    mNativeAd = nativeAd;
                }
            }
        };
        InMobiNative.requestAd(this, mInMobiAdRequest, mListener);

        ...
    }
    public InMobiNative getNativeAd() {
        return mNativeAd;
    }
}
```

三. 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, Placement
ID, mInterstitialAdEventListener);
```

ImInterstitialAdEventListener是 InterstitialAdEventListener的实现类，

InterstitialAdEventListener用于监听插屏广告的后续行为

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

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InterstitialAdEventListener）

```
public abstract class InterstitialAdEventListener {
  public void onAdLoadSucceeded(InMobiInterstitial ad) {}
   public void onAdLoadFailed(InMobiInterstitial ad, InMobiAdRequestStatus status) {}
   public void onAdReceived(InMobiInterstitial ad) {}
   public void onAdClicked(InMobiInterstitial ad, Map<Object, Object> params) {}
    public void onAdWillDisplay(InMobiInterstitial ad) {}
    public void onAdDisplayed(InMobiInterstitial ad) {}
   public void onAdDisplayFailed(InMobiInterstitial ad) {}
   public void onAdDismissed(InMobiInterstitial ad) {}
    public void onUserLeftApplication(InMobiInterstitial ad) {}
  public void onRewardsUnlocked(InMobiInterstitial ad, Map<Object, Object> rewards) {}
}
```

步骤二：加载插屏广告

```
interstitialAd.load();
```

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后，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，可参考以下代码段。

```
InterstitialAdEventListener mInterstitialAdEventListener = new InterstitialAdEventListener
() {
      @Override
      public void onAdLoadSucceeded(InMobiInterstitial inMobiInterstitial) {
         Log.d(TAG, "Ad can now be shown!");
         mCanShowAd = true;
      }
   };

  void init() {
     InMobiInterstitial interstitialAd = new InMobiInterstitial(GameActivity.this, PlacemenID,
mInterstitialAdEventListener);
  }

  void prepareGameLevel() {
     interstitialAd.load();
  }

  void handleGameLevelCompleted() {
      If (mCanShowAd) interstitialAd.show();
  }
```

广告展示成功，则会回调InterstitialAdEventListener中的 onAdWillDisplay(),展示失败，则会回调 onAdDisplayFailed(), 这时可通过load() 重新请求广告。广告关闭或消失，则回调 onAdDismissed()

步骤四：插屏的动画效果

插屏出现和消失时的动画效果，可通过以下代码实现

```
interstitialAd.show(R.anim.anim_entry, R.anim.anim_exit);
```

四. 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，代码如下

```
InterstitialAdEventListener mInterstitialAdEventListener = new InterstitialAdEventListener() {

    @Override
    public void onAdRewardActionCompleted(InMobiInterstitial inMobiInterstitial,
Map<Object, Object> map) {
        Log.d(TAG, "Ad rewards unlocked!");
        for (Object key : map.keySet()) {
            Object value = map.get(key);
            Log.v(TAG, "Unlocked " + value + " " + key);
        }
    }
};
```

五. 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="plid-1431977751489005"
        ads:refreshInterval="60"
    />
</LinearLayout>
```

注意⚠️：开发者如果正在使用Android SDK 720版本及以上，使用XML布局的方式集成banner, 则需在placementId标签前添加“plid-”，代码如下：

```
ads:placementId="plid-1431977751489005"
```

步骤二：引入banner对象

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

方式二：代码实现

步骤一：在代码中获取InMobiBanner对象

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

    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：bannerAd.load()方法来加载Banner广告

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

注意⚠：请务必指明banner布局的宽和高，如果宽高值传入WRAP_CONTENT会引起程序化的错误，该规则使用于上述两种实现方式。

步骤四：设置banner的自动刷新

第一次请求完，banner会自动刷新，可以通过以下代码设置刷新时长

```
setRefreshInterval(int)
```

步骤五：动画刷新banner

通过以下代码，可以选择banner刷新时的动画效果，默认的风格为AnimationType.ROTATE_HORIZONTAL_AXIS，也可以通过AnimationType.ANIMATION_OFF关掉该动画效果

```
setAnimationType(InMobiBanner.AnimationType);
```

步骤六：可以通过BannerAdListener监听banner的后续行为

```
public abstract class BannerAdEventListener {
    public void onAdLoadSucceeded(InMobiBanner ad) {}
    public void onAdLoadFailed(InMobiBanner ad, InMobiAdRequestStatus status) {}
    public void onAdClicked(InMobiBanner ad, Map<Object, Object> params) {}
    public void onAdDisplayed(InMobiBanner ad) {}
    public void onAdDismissed(InMobiBanner ad) {}
    public void onUserLeftApplication(InMobiBanner ad) {}
    public void onRewardsUnlocked(InMobiBanner ad, Map<Object, Object> rewards) {}
 }
```

测试

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);
```