

# InMobi iOS Native Reboot SDK 集成指南[7.2.8]

此文档将向您介绍如何集成轻量级的iOS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. 初始化SDK
3. 集成原生广告（支持信息流 / 开屏 / 前贴等样式）
4. 集成插屏（全屏）、插屏视频广告
5. 集成激励性视频广告
6. 集成Banner广告（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：** InMobi SDK的最新版本支持iOS 8及更高版本。另外，这个版本的iOS SDK需要XCode 9.0或更高版本。

中国版SDK下载地址：[http://sdk-china.s3.cn-north-1.amazonaws.com.cn/SDK/InMobi\\_iOS\\_SDK.zip](http://sdk-china.s3.cn-north-1.amazonaws.com.cn/SDK/InMobi_iOS_SDK.zip)

国际版SDK下载地址：[https://dl.inmobi.com/SDK/InMobi\\_iOS\\_SDK.zip](https://dl.inmobi.com/SDK/InMobi_iOS_SDK.zip)

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

向xCode中添加框架和额外的Build Settings来配置，步骤如下：

步骤1：添加下列**必需框架**到xCode工程中

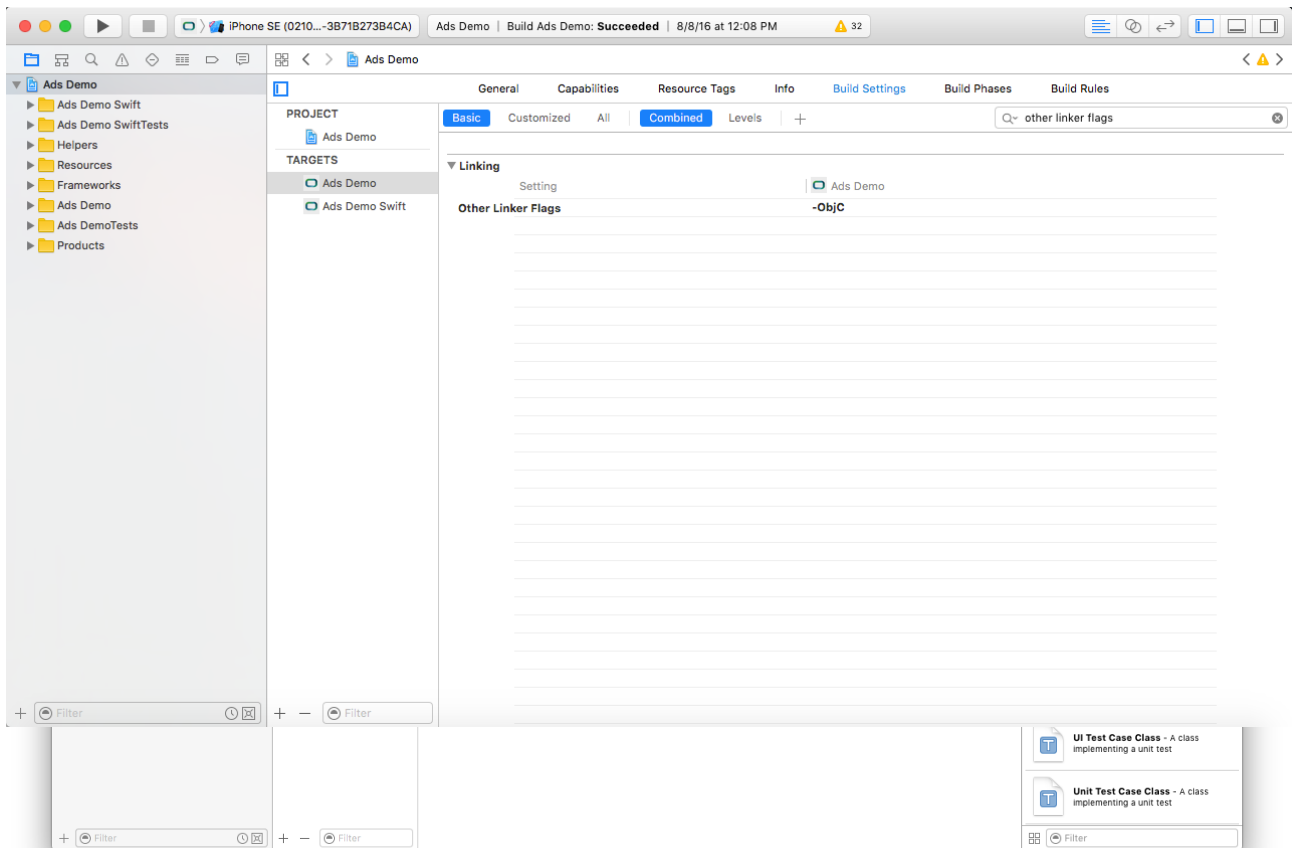
- InMobiSDK.framework （可从InMobi SDK包中获取）
- libsqlite3.0.tbd
- libz.tbd
- libxml2.tbd
- WebKit.framework

步骤2：添加 **-ObjC** flag到**Other Linker Flags**：

- 在xCode工程中选择 Application Target > Build Settings
- 在搜索框中，搜索**Other Linker Flags**
- 添加 **-ObjC** flag

步骤3：配置**ATS**属性

我们建议在Apple强制启用https链接之前，暂时关闭ATS（Application Transport Security）属性以获取更好的广告填充。同时我们正在努力和所有的广告主以及监测公司对所有未达标的链接进行升级。在此过程中，一些广告还没法做到完全使用https的链接，敬请谅解。



您也可以在app的

**Info.plist**中添加下列代码片段来关闭ATS Flag

关于添加ATS白名单的说明：

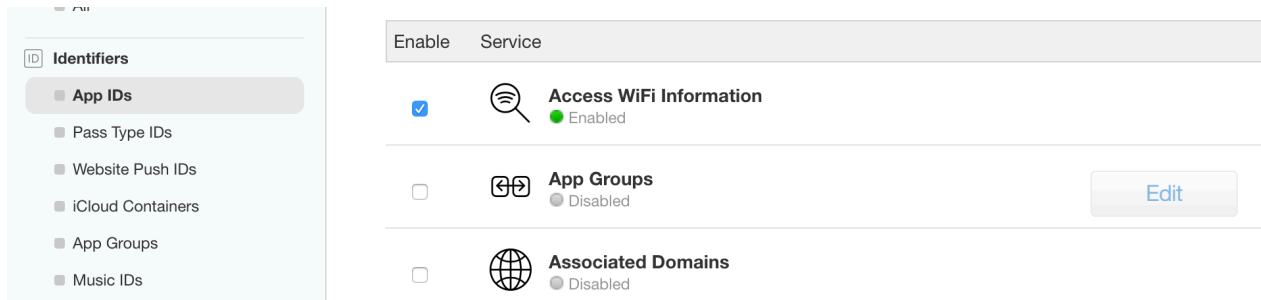
在关闭了ATS属性后，您仍然可以配置ATS的白名单来满足一些特殊需求，但我们建议不要添加任何InMobi的域名，以及和广告业务相关的其他三方域名（如CDN），以免影响广告在app内的投放及展示。

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
    <true/>
<key>NSExceptionDomains</key>
    <dict>
        <key>example.com</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
        </dict>
    </dict>
</dict>
</dict>
```

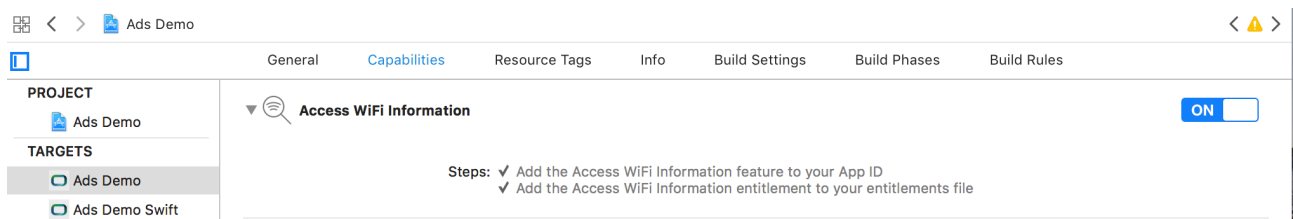
#### 步骤4：iOS 12 WiFi设置

从ios12开始，苹果公司就引入了隐私设置来访问WiFi细节。为了提高盈利和相关用户体验，我们鼓励共享目标广告的WiFi详细信息。

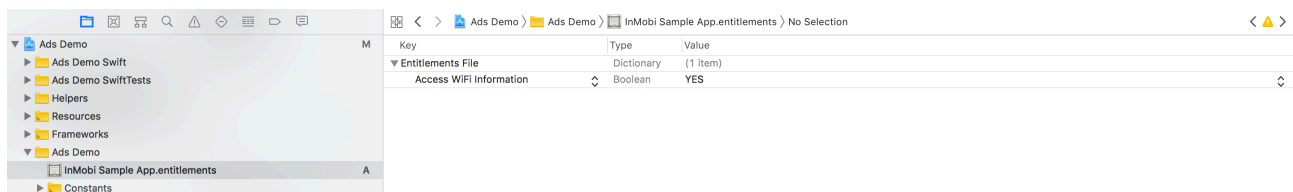
第1步：在您的应用ID上启用“访问WiFi信息”。



第2步：从XCode功能启用对目标应用的访问“访问WiFi信息”



第3步：确保将WiFi Access添加到App.entitlements文件中。



## Notes：WKWebView中的音频处理

对于运行任何音频或视频媒体作为主要内容的开发者，我们建议您通过观察AVAudioSession发布的AVAudioSessionInterruptionnotification来监听中断。这有助于优化各种音频夺权问题，提高用户体验。有关更多信息，请参照这些Apple文档：[Article 1](#), [Article 2](#)。

## 2. 初始化SDK

步骤一：在AppDelegate头文件中导入InMobi SDK

```
@import InMobiSDK;
```

步骤二：用InMobi Account ID在App的delegate.m文件中的didFinishLaunchingWithOptions回调方法中初始化InMobi SDK

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    //使用InMobi Account ID初始化SDK
```

```

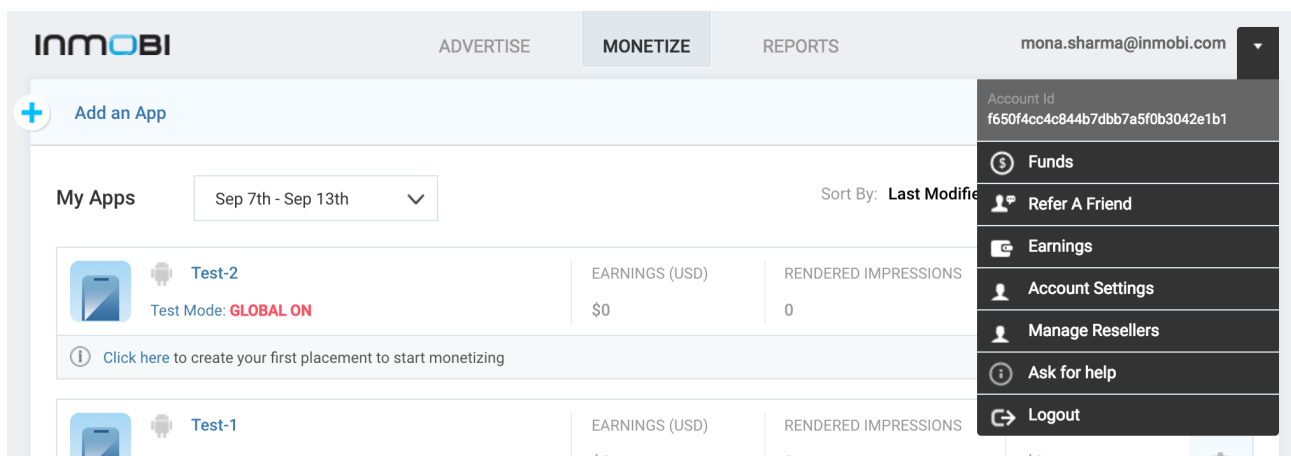
[IMSdk initWithAccountID:@"在此输入InMobiAccountID"];
[IMSdk initWithAccountID:@"Insert InMobi account ID here"consentDictionary:@{<Insert
consentObject dictionary here>}]; (欧盟国家流 app必须使 此初始化方法)

// 其他逻辑
return YES;
}

```

**注意!** consentObject - [consentObject](#) 是个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 户授权以符合GDPR约束。  
GDPR 详情可参考 [here](#).

点击InMobi开发者后台右上角的邮箱地址处获取AccountID，或联系和您对接的商务同事获取。



### 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:

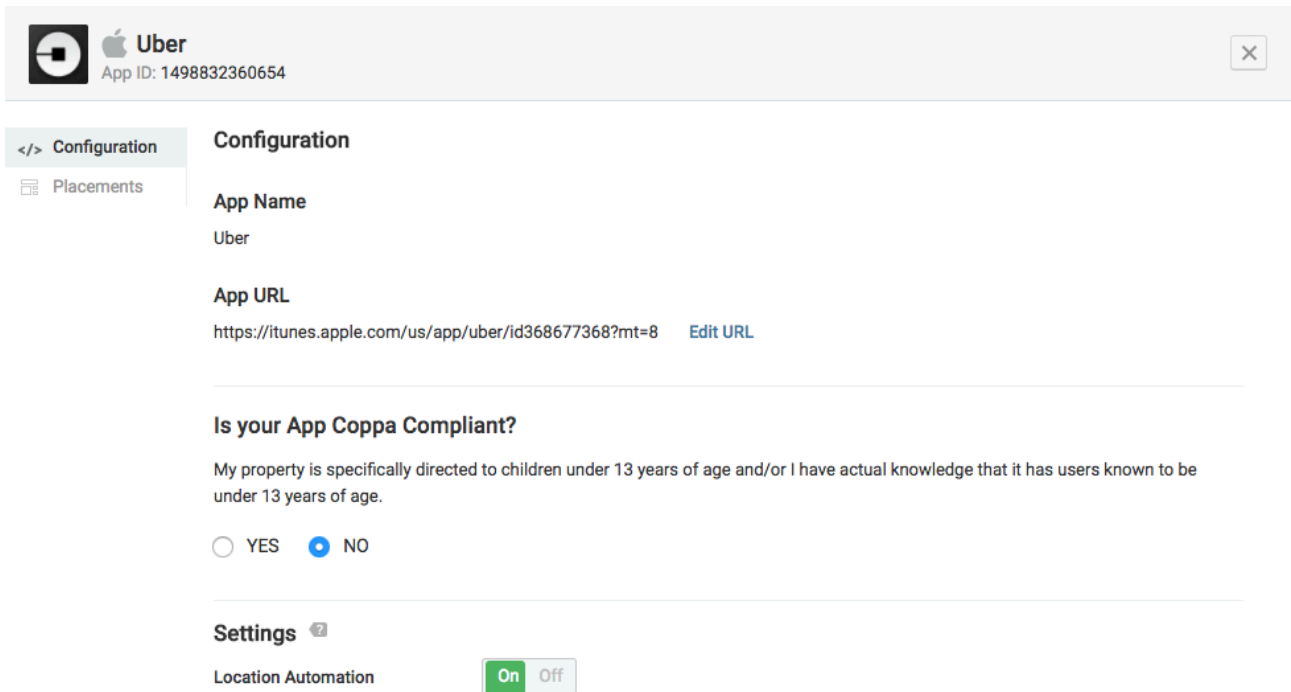
- 每个session对话中都需要提供consentObject. SDK 会永久保存授权信息，只会把consentObject存在内存中. 如果应 重启或崩溃consentObject就会丢失
- 如果运行中consentObject值有改变，可以按照下列方式更新授权信息  
`IMSdk.updateGDPRConsent(<Insert consentObject dictionary here>)`

步骤三：传地理位置信息

**重要！** 如果您的应用使用了用户地理位置信息，我们建议您将这部分信息传递给SDK，因为具备地理位置的广告位更具变现优势。

方法一：通过开发者后台设置传入地理位置信息：

开关设置为On时，如果您的应用有可用的地理位置信息，广告SDK即可获取该地理位置：



The screenshot shows the configuration page for the Uber app in the Google Play Console. The app ID is 1498832360654. The configuration includes the app name 'Uber' and the app URL 'https://itunes.apple.com/us/app/uber/id368677368?mt=8'. Under the 'Settings' section, the 'Location Automation' toggle is turned 'On'.

方法二：代码实现传入地理位置信息：

在您工程的info.plist添加NSLocationWhenInUseUsageDescription（location权限请求描述），并实现以下代码。

```
//使用 <CoreLocation/CoreLocation.h> 获取坐标
CLLocationManager *mgr = [[CLLocationManager alloc] init];
CLLocation *loc = mgr.location;
[IMSdk setLocation:loc];
```

步骤四：传性别、年龄信息

**重要！** 如果您的应用已获得用户授权使用其年龄、性别，我们建议您将这部分信息传递给SDK，获得更好的广告定向。

```
// 性别，年龄：根据用户实际情况，选择合适的值传入
[IMSdk setGender:kIMSDKGenderFemale];
[IMSdk setAge:20];
```

步骤五(可选)：通过设置debug log level打印广告详情便于调试，正式上线前请关闭。

```
//在初始化SDK前设置debug log level
[IMSdk setLogLevel:kIMSDKLogLevelDebug];
```

### 3. 集成原生广告 - 支持信息流 / 开屏 / 前贴等样式

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，可以用于实现信息流 / 开屏 / 前贴等广告形式（视频或图片）。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。

API介绍：

IMNative Objects:	
NSString* adTitle	返回广告标题
NSString* adDescription	返回广告描述
UIImage* adIcon	返回广告图标
NSString* adCtaText	返回广告点击行为描述，如“免费安装”等
NSString* adRating	返回广告评分，如5.0 (非必含项)
NSURL* adLandingPageUrl	返回广告落地页地址链接
BOOL isAppDownload	YES表示为下载类广告；NO表述为非下载类广告
NSString* customAdContent	广告返回的内容（json格式）

关于判断当前返回的广告素材类型（视频/静态）说明

可以通过NSString\* customAdContent解析出广告的素材类型，以下是json解析的代码：

```
NSString *jsonString =native.customAdContent;
NSData *data = [jsonString dataUsingEncoding:NSUTF8StringEncoding];
NSDictionary* jsonDict = [NSJSONSerialization JSONObjectWithData:data options:0
error:nil];
BOOL isVideo = [[jsonDict objectForKey:@"isVideo"] boolValue];
```

IMNative functions:	
-(UIView*)primaryViewOfWidth:(CGFloat)width	此方法将返回根据您提供的width生成的UI View。这个UIView即是广告的主体素材，该素材的横纵比和您在InMobi后台建立广告位时选择的横纵比一致。当这个UIView被展示时，SDK将自动上报广告的曝光事件；当这个UIView被点击时，SDK将自动处理广告跳转和上报点击事件。 如果您在创建广告位时选择的是 <b>Feed</b> 样式，您也需要通过此方法获取一个AdChoices图标(设width为25)，并通过展示这个小图标来保证广告的曝光上报。

IMNative functions:	
-(void)load	请求广告，同一广告实例 <b>不支持</b> 重复调用load方法，详见集成步骤六[广告刷新]。
-(BOOL)isReady	返回为真表示广告准备完毕，可以用于展示
-(void)reportAdClick	此方法仅上报广告的点击事件，需要开发者自行实现landingURL落地页链接打开的方法。开发者自己添加点击跳转控件时可以使用此方法。
-(void)reportAdClickAndOpenLandingPage	此方法既上报广告的点击事件，同时自动处理广告的落地页跳转。开发者自己添加点击跳转控件时可以使用此方法。
-(void)recyclePrimaryView	广告滑出可视区域时调用此方法回收先前的Primary View。

## InFeed - 用于集成信息流广告样式

步骤一：导入头文件，定义广告在信息流中的位置，在ViewController.m文件中创建原生广告对象，参考如下：

```
#import <InMobiSDK/InMobiSDK.h>
//例如在第信息流4个位置展示广告
#define IM_AD_INSERTION_POSITION 4

@interface TableViewController () <IMNativeDelegate>
@property(nonatomic,strong) IMNative *InMobiNativeAd;
@end
```

步骤二：在ViewController.m文件中创建原生广告对象，设置IMNative代理为当前ViewController自身，调用load加载广告：

```
-(void)viewDidLoad {
    [super viewDidLoad];
    self.InMobiNativeAd = [[IMNative alloc] initWithPlacementId:<此处输入原生广告Placement ID>];
    self.InMobiNativeAd.delegate = self;
    [self.InMobiNativeAd load];
}
```

**注意！** IMNative广告对象不支持重复调用load方法，请求新广告请重新生成新的IMNative广告对象并调用load，详情请参考步骤六[广告刷新]。

步骤三：本次请求如果有广告填充将会在nativeDidFinishLoading回调之前进入此回调，没有广告返回将不会进入此回调。（开发者可以自由选择是否使用）

```
-(void)nativeAdIsAvailable:(IMNative*)native{
    NSLog(@"Native Ad Is Available"); // 广告有返回但还没有渲染好
}
```

步骤四：实现IMNativeDelegate的成功回调nativeDidFinishLoading方法，该回调触发说明广告已经加载完毕并可以用于展示，在这个回调里建议您把广告对象InMobiNativeAd添加到TableView的data source.

```
-(void)nativeDidFinishLoading:(IMNative*)native{
    [self.tableData insertObject:native atIndex:IM_AD_INSERTION_POSITION];
    [self.tableView reloadData];
    NSLog(@"Native Ad did finish loading");
}
```

步骤五：广告展示和曝光、点击事件上报

成功获取广告后（走到nativeAdDidFinishLoading或isReady为真），可以在tableView的cellForRowAtIndexPath回调中实现广告加载，参考如下：

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    InFeedTableCell *cell = (InFeedTableCell *)[tableView
    dequeueReusableCellWithIdentifierWithIdentifier:@"InFeedTableCell"];
    NSArray *nib = [[NSBundle mainBundle] loadNibNamed:@"InFeedTableCell" owner:self
    options:nil];
    cell = [nib objectAtIndex:0];
    id slide = [self.tableData objectAtIndex:indexPath.row];
    if ([self isAdAtIndex:indexPath.index]) {
        IMNative * currentNativeAd = slide;
        cell.imageView.image = currentNativeAd.adIcon;
        cell.titleLabel.text = currentNativeAd.adTitle;
        cell.subtitleLabel.text = @"Sponsored";
        cell.descriptionLabel.text = currentNativeAd.adDescription;
        cell.ctaLabel.text = currentNativeAd.adCtaText;
        //Calculate your feed's primaryImageViewWidth and use it to fetch and Ad Primary
        view of same width.
        UIView* AdPrimaryViewOfCorrectWidth = [currentNativeAd
        primaryViewOfWidth:primaryImageViewWidth];
        //Set the frame of Ad Primary View same as that of your feed's Primary View
        AdPrimaryViewOfCorrectWidth.frame = primaryImageViewFrame;
        [cell addSubview:AdPrimaryViewOfCorrectWidth];
        UITapGestureRecognizer *singleTapAndOpenLandingPage = [[UITapGestureRecognizer
        alloc] initWithTarget:currentNativeAd action:@selector(reportAdClickAndOpenLandingPage)];
        cell.ctaLabel.userInteractionEnabled = YES;
        [cell.ctaLabel addGestureRecognizer:singleTapAndOpenLandingPage];
    }
    else
    {
        //Your App's TableCell implementation
    }
    return cell;
}
```



**重要！** 仅当primary view屏幕可见时（满足50%以上可见且持续时间不短于1秒），广告的曝光事件才会自动上报。

另外，如果您自己添加了广告点击跳转控件，选择不自动跳转落地页（InMobi技术帮您操作屏蔽自动跳转），需要您自行实现landingURL落地页链接的打开，参考如下：

```
if (landingUrl != nil){
    NSURL* landingPageURL = [NSURL URLWithString:landingUrl];
    [[UIApplication sharedApplication] openURL:landingPageURL];
}
```

#### 步骤六：回收Primary view

为了优化内存，每次当广告滑出可视区域时，都需要调用recyclePrimaryView方法来回收Primary view。以TableView为例，可以在didEndDisplayingCell方法中调用：

```
-(void)tableView:(UITableView *)tableView didEndDisplayingCell:(UITableViewCell *)cell
forRowAtIndexPath:(NSIndexPath *)indexPath {
    if([self isAdAtIndexPath:indexPath]){
        [self.InMobiNativeAd recyclePrimaryView];
    }
}
```

注意：调用recyclePrimaryView方法后，因为当前的Primary view已经被回收，如需再次展示广告则需要调用[self.InMobiNativeAd primaryViewOfWidth:<custom width>]重新获取。

如果您已经采用了步骤四中的方案（即cellForRowAtIndexPath中获取Primary view），则无需重复操作。

#### 步骤七：广告刷新

实现恰当的广告刷新逻辑能够优化广告的展示率。需要注意的是，在调用load刷新广告前，需要将IMNative广告对象从tableData中移除，并通过调用recyclePrimaryView回收当前的Primary View。IMNative广告对象不支持重复调用load方法，请求新广告请重新生成新的IMNative广告对象并调用load。

建议您采用样例代码中的方法调用顺序来进行广告刷新：

```
-(void)refreshInMobiStrandAd {
    [self.tableData removeObject:self.InMobiNativeAd];
    [self.tableView reloadData];
    [self.InMobiNativeAd recyclePrimaryView];

    self.InMobiNativeAd = [[IMNative alloc] initWithPlacementId:<此处输入原生广告Placement ID>];
    self.InMobiNativeAd.delegate = self;
    [self.InMobiNativeAd load];
}
```

#### 步骤八：转屏处理

如果您的应用支持转屏，则需要用屏幕旋转后的新广告位宽度，再次调用[self.InMobiNativeAd primaryViewOfWidth:<custom width>] 方法获取新的Ad Primary View来替换当前的广告视图，否则可能因为广告展示不全导致广告展示异常。

#### 步骤八：销毁广告对象

您应当在ViewController 的dealloc 方法中通过将InMobiNativeAd广告对象和代理置为nil进行销毁，但请务必确保在dealloc销毁前您已经通过调用recyclePrimaryView进行了回收。

```
-(void)dealloc {
    [self.InMobiNativeAd recyclePrimaryView];
    self.InMobiNativeAd.delegate = nil;
    self.InMobiNativeAd = nil;
}
```

## Splash - 用于集成开屏广告样式

IMNative同样可以被用于实现开屏，除上面介绍过的操作外，以下几点需要您的特别注意：

### 第一点：广告加载成功的判断

开屏建议您通过self.InMobiNativeAd.isReady来判断广告是否已经加载完毕以用于展示。由于nativeDidFinishLoading回调可能会因为主线程阻塞导致接收延迟，因此建议您在开屏广告加载等待时间结束前，通过调用**isReady**主动检查广告的加载情况。

### 第二点：避免误删广告

用户点击开屏后，在SDK完成点击上报和落地页跳转前，请勿强行移除广告。

您可以通过广告对象的nativeWillPresentScreen回调判断广告详情页的展示状态，参考如下：

```
-(void)nativeWillPresentScreen:(IMNative*)native{
    NSLog(@"Native Ad will present screen");
    isSecondScreenDisplayed = YES;
}
```

### 第三点：回收Primary View

您应当在消除开屏广告前回收广告的Primary View，参考如下：

```
-(void)dismissAd{
    if(isSecondScreenDisplayed){
        NSLog(@"DO NOT DISMISS THE AD WHILE THE SCREEN IS BEING DISPLAYED");
    }
    else
    {
        self.SplashAdView.hidden = true;
        [self.InMobiNativeAd recyclePrimaryView];
        self.InMobiNativeAd = nil;
    }
}
```

## Preroll - 用于集成前贴广告样式

IMNative同样可以被用于实现前贴，除上面介绍过的操作外，以下几点需要您的特别注意：

### 第一点：消除广告的时机

您应当在nativeDidFinishPlayingMedia视频播放完毕回调触发后再移除广告，参考如下：

```
-(void)nativeDidFinishPlayingMedia:(IMNative *)native{
    [self dismissAd];
}
```

第二点：回收Primary View

您应当在消除前贴广告前回收广告的Primary View，参考如下：

```
-(void)dismissAd{
    self.PrerollAdView.hidden = true;
    [self.InMobiNativeAd recyclePrimaryView];
    self.InMobiNativeAd = nil;
}
```

## 高级广告控制

IMNative广告支持的所有回调如下，出了上文中提及的，您还可以根据实际需要自行选取使用：

```
-(void)nativeAdIsAvailable:(IMNative*)native{
    NSLog(@"Native Ad Is Available"); // 广告有返回但还没有渲染好
}
-(void)nativeDidFinishLoading:(IMNative*)native{
    NSLog(@"Native Ad load Successful"); // 广告已经准备好用于展示
}
-(void)native:(IMNative*)native didFailToLoadWithError:(IMRequestStatus*)error{
    NSLog(@"Native Ad load Failed"); // 广告获取失败，请打印error查看详情
}
-(void)nativeWillPresentScreen:(IMNative*)native{
    NSLog(@"Native Ad will present screen"); //广告详情页即将展示
}
-(void)nativeDidPresentScreen:(IMNative*)native{
    NSLog(@"Native Ad did present screen"); //广告详情页展示
}
-(void)nativeWillDismissScreen:(IMNative*)native{
    NSLog(@"Native Ad will dismiss screen"); //广告详情页即将消失
}
-(void)nativeDidDismissScreen:(IMNative*)native{
    NSLog(@"Native Ad did dismiss screen"); //广告详情页消失
}
-(void)userWillLeaveApplicationFromNative:(IMNative*)native{
    NSLog(@"User leave"); //用户即将离开当前应用
}
-(void)native:(IMNative *)native didInteractWithParams:(NSDictionary *)params{
    NSLog(@"User clicked"); //广告被点击
}
-(void)nativeAdImpressed:(IMNative *)native{
    NSLog(@"User viewed the ad"); //广告被展示
}
-(void)nativeDidFinishPlayingMedia:(IMNative*)native{
```

```

    NSLog(@"The Video has finished playing"); // 视频播放完毕，用于前贴广告类型
}
-(void)userDidSkipPlayingMediaFromNative:(IMNative*)native{
} // 用户点击跳过按钮
-(void)native:(IMNative *)native rewardActionCompletedWithRewards:(NSDictionary *)rewards{
    NSLog(@"Rewarded"); // 发放用户奖励，用于激励广告
}
-(void)native:(IMNative*)native adAudioStateChanged:(BOOL)audioStateMuted {
    if (audioStateMuted) {
        NSLog(@"Inline video-ad audio state changed to mute");
    } else {
        NSLog(@"Inline video-ad audio state changed to unmute");
    }
}
//视频信息流广告视频音频状态发生变化时调用此方法。
}

```

## 4. 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：导入头文件，在ViewController.h文件中创建原生广告对象，参考示例如下：

```

#import <UIKit/UIKit.h>
#import <InMobiSDK/InMobiSDK.h>
@interface ViewController : UIViewController <IMInterstitialDelegate>
@property (nonatomic, strong) IMInterstitial *interstitial;
@end

```

步骤二：创建IMInterstitial对象，我们建议您在ViewController.m文件中的viewDidLoad回调中创建

```

self.interstitial = [[IMInterstitial alloc] initWithPlacementId: <Placement ID, Long型>];

```

步骤三：在调用load方法之前，需先实现IMInterstitial属性的代理以获取广告的各种状态，具体回调方法可在IMInterstitialDelegate.h文件中查看

注意：每个Placement ID只允许创建一个与之对应的IMInterstitial对象

```

self.interstitial.delegate = self;

/**
 * 服务器已经成功返回广告。此时素材可能还未完全加载完毕，请用interstitialDidFinishLoading
 * 回调方法来获取广告素材的加载情况
 */
-(void)interstitialDidReceiveAd:(IMInterstitial *)interstitial;

/**
 * 此回调表示广告返回成功且素材加载完毕，此时可以展示全屏广告
 */
-(void)interstitialDidFinishLoading:(IMInterstitial*)interstitial;

```

```

/**
 * 此回调表示插屏广告在加载过程中遇到了一些问题，请打印IMRequestStatus
 * 查看具体内容
 */
-(void)interstitial:(IMInterstitial*)interstitial didFailToLoadWithError:(IMRequestStatus*)error;

-(void)interstitialWillPresent:(IMInterstitial*)interstitial;

-(void)interstitialDidPresent:(IMInterstitial *)interstitial;

-(void)interstitial:(IMInterstitial*)interstitial didFailToPresentWithError:(IMRequestStatus*)error;

-(void)interstitialWillDismiss:(IMInterstitial*)interstitial;

-(void)interstitialDidDismiss:(IMInterstitial*)interstitial;

-(void)interstitial:(IMInterstitial*)interstitial didInteractWithParams:(NSDictionary*)params;

-(void)userWillLeaveApplicationFromInterstitial:(IMInterstitial*)interstitial;
-(void)interstitial:(IMInterstitial*)interstitial rewardActionCompletedWithRewards:
(NSDictionary*)rewards;

```

#### 步骤四：加载插屏广告

您需要先加载一个插屏广告并通过回调检查素材加载情况，当广告和素材都加载成功后，才能展示插屏广告。

加载插屏广告： [self.interstitial load];

viewController文件代码片段如下：

```

-(void)viewDidLoad {
    [super viewDidLoad];
    self.interstitial = [[IMInterstitial alloc] initWithPlacementId:<Insert InMobi placement ID here>];
    self.interstitial.delegate = self;
    [self.interstitial load];
}

```

当您收到广告加载成功的回调后，可以调用IMInterstitial的isReady方法来查看广告素材是否加载完成。当广告素材加载完毕时，调用下面的方法展示全屏广告：

```
[self.interstitial showFromViewController:self];
```

## 5. 集成激励性视频广告

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：

```

/**
 * 此回调表示激励视频广告播放完毕，开发者此时可以将奖励发放给用户

```

```
*/  
-(void)interstitial:(IMInterstitial*)interstitial rewardActionCompletedWithRewards:  
(NSDictionary*)rewards;
```

## 6. 集成BANNER广告（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。

步骤一：导入头文件，之后在您的ViewController头文件中定义Banner对象

```
#import <UIKit/UIKit.h>  
#import <InMobiSDK/InMobiSDK.h>  
@interface ViewController : UIViewController <IMBannerDelegate>  
@property (nonatomic, strong) IMBanner *banner;  
@end
```

步骤二：我们建议您在ViewController.m文件中的viewDidLoad方法中初始化Banner对象。

```
self.banner = [[IMBanner alloc] initWithPlacementId:<此处为Long型Banner Placement ID>];
```

步骤三：设置IMBanner的代理

```
self.banner.delegate = self;
```

步骤四：在加载广告之前，将广告的adview添加到主界面上

```
[self.view addSubview:self.banner];
```

步骤五：加载Banner广告

```
[self.banner load];
```

ViewController文件示例：

```
-(void)viewDidLoad {  
    [super viewDidLoad];  
    self.banner = [[IMBanner alloc] initWithFrame:CGRectMake(0, 0, 320, 50)  
                placementId:@"Insert your placement ID here"];  
    self.banner.delegate = self;  
    [self.view addSubview:self.banner];  
    [self.banner load];  
}
```