AdsMOGO iOS SDK 自定义广告平台使用说明

基于 AdsMOGO_iOS_SDK 使用说明，请先阅读此文档

一、什么是自定义广告平台

当您使用的广告平台在 AdsMogo 中不支持时，可以使用“自定义广告平台”，这是一个很好的解决办法。您可以使用“自定义平台”去启动任何代码。当您需要使用那些广告平台时，您需要在“自定义平台”中添加这个平台名称并填写相应的 key 或 id，开启它。在程序中通过一个 Adapter（适配器）来工作。

注：自定义平台只适用于芒果不支持的广告平台， 如果自定义芒果支持的平台有可能对芒果自有的系统造成影响，因此，请不要再自定义芒果已经支持的平台。

二、工作流程

首先，您需要在 App 配置页面添加一个“自定义广告平台”，填写相应平台 key。随后，您就可以开启使用该自定义平台。当这个“自定义平台”被轮换到时，就会被调用。在您的 Activity 中，即在您加载 AdsMogo 广告窗体的 Activity 中，定义返回您写好的 Adapter，AdsMogo 广告系统就可以使用这个 Adapter。您可以在这个方法中完成任何功能，当然也包括在这个方法里去执行别的广告平台，来将 AdsMogo 正在展示的 banner 内容换成其他平台的广告，或者任何其他广告。

三、后台配置

1，登录后台网站 http://my.adsmogo.com；
2，进入某一 App 的广告平台配置页面，点击“自定义广告平台”；
3，填写平台名称及该平台 key；
4，开启该平台并设置投放量后，点击保存。

请注意，每个广告平台列表中，投放总数必须为100%或者全部关闭，否则无法进行下一步操作。

保存 取消

四、自定义横幅接口介绍和使用方法

首先创建一个类作为 Adapter
引入 AdMoGoAdapterCustomEvent.h
继承 AdMoGoAdapterCustomEvent

1，需要在实现文件中重新实现三个方法：

```
+(void)load;
+(AdMoGoAdNetworkType)networkType;
-(void)getAd;
```

其中
a :**+(void)load** 是 NSObject 中的方法,在此重新实现是为了将自定义的 Adapter 注册进 Mogo 的 SDK 中，具体方法如下：

```
/**
 *注册adapter
 *将自定义的adapter注册入Mogo
 */
+(void)load{
    [super registerClass:self];
}
```

b: **+(AdMoGoAdNetworkType)networkType** 方法用来标记当前的 Adapter 对应的是哪个自定义平台,mogo 暂时支持最多 3 个自定义平台,他们的定义分别为：
//AdMoGoAdNetworkTypeCustomEventPlatform_1
//AdMoGoAdNetworkTypeCustomEventPlatform_2
//AdMoGoAdNetworkTypeCustomEventPlatform_3

将其对应作为返回值即可，具体方法如下：

```
+(AdMoGoAdNetworkType)networkType{
return      AdMoGoAdNetworkTypeCustomEventPlatform_1;
}
```

c: **-(void)getAd** 这个方法是用于初始化广告并开始请求的通知，一般的应该在此方法中初始化用户所需要添加平台的 AdView 等信息 ,需要注意的是由于需要适配多种屏幕尺寸，多数平台在初始化的同时需要选择一个合适的广告尺寸作为初始化参数来初始化一个合适大小的 View，这个尺寸是和 MogoSDK 所支持的广告类型所对应的，具体 Mogo 所能使用的广告类型在 MogoSDK 文档和 Sample 中都有说明，这里不在赘述。
这个参数可以通过方法-(AdViewType)customAdapterWillgetAdAndGetAdViewType;获得。与此同时，在执行此方法后 MogoSDK 会为您初始化一些参数，所以即使用户不需要 AdViewType，也务必向此方法发送消息。

下面是初始化多盟的示例：

```
//通知Mogo请求广告开始并且获得广告尺寸类型
//注：即便是不需要使用AdViewType,也要在此向
customAdapterWillgetAdAndGetAdViewType发一次消息
    AdViewType type = [super customAdapterWillgetAdAndGetAdViewType];

    //尺寸类型如下
    //AdViewTypeNormalBanner = 1,       //e.g. 320 * 50 ; 320 * 48...   iphone banner
    //AdViewTypeLargeBanner    = 2,     //e.g. 728 * 90 ; 768 * 110      ipad only
    //AdViewTypeMediumBanner = 3,       //e.g. 468 * 60 ; 508 * 80       ipad only
```

```
    //AdViewTypeRectangle      = 4,       //e.g. 300 * 250; 320 * 270      ipad only
    //AdViewTypeFullScreen     = 6,       //iphone full screen ad
    //AdViewTypeiPadNormalBanner = 8, //ipad use iphone banner
        CGSize size = CGSizeZero;

    //一般的,在这里通过type初始化用户展示广告的Adview 或者选择合适的广告尺寸定义,以便于请求相应的广告
    switch (type) {
        case AdViewTypeNormalBanner:
        case AdViewTypeiPadNormalBanner:
            size = DOMOB_AD_SIZE_320x50;
            break;
        case AdViewTypeMediumBanner:
            size = DOMOB_AD_SIZE_488x80;
            break;
        case AdViewTypeLargeBanner:
            size = DOMOB_AD_SIZE_728x90;
            break;
        default:
            break;
    }

    //初始化单一平台的AdView
    //单一平台的id为对象 key 以 NSDictionary 形式存在,其中两个两个KEY分别被定义为APPID_F,APPID_S.具体内容是由用户在后台的配置决定的;
    DMAdView *adview = [[DMAdView alloc] initWithPublisherId:[key objectForKey:APPID_F] size:size autorefresh:NO];

    //此处由于单一需要当前的UIViewController对象 可以通过
    //[adMoGoDelegate viewControllerForPresentingModalView] 获取
    //此方法对应的是-(UIViewController *)viewControllerForPresentingModalView;
    //所以使用此方法获取 UIViewController 前务必实现此回调
    adview.rootViewController = [adMoGoDelegate viewControllerForPresentingModalView];
    adview.delegate = self;
    [adview loadAd];
```

```
    //记录当前的adView,在初始化之后必须要做
    self.adNetworkView = adview;
      [adview release];
```

2．广告请求展示失败等接口介绍

a：-(void)adMoGoCustom:(AdMoGoAdNetworkAdapter *)adapter didReceiveAdView:(UIView *)adView
通知 MogoSDK 广告请求成功并且将 AdView 加入到 Mogo 视图上。

b: -(void)adMoGoCustom:(AdMoGoAdNetworkAdapter *)adapter didFailAd:(NSError *)error;
通知 MogoSDK 广告请求失败,继续轮换其他的平台。

五、自定义全插屏接口介绍和使用方法

首先创建一个类作为 Adapter
引入 AdMoGoFullScreenAdapterCustomEvent.h
继承 AdMoGoFullScreenAdapterCustomEvent

1，需要在实现文件中重新实现五个方法：

```
+(void)load;
+(AdMoGoAdNetworkType)networkType;
-(void)getAd;
- (BOOL)isReadyPresentInterstitial;
- (void)presentInterstitial;
```

其中
a :**+(void)load** 是 NSObject 中的方法,在此重新实现是为了将自定义的 Adapter 注册进 Mogo 的 SDK 中，具体方法如下：

```
/**
 *注册adapter
 *将自定义的adapter注册入Mogo
 */
+(void)load{
    [super registerClass:self];
```

```
}
```

b: **+(AdMoGoAdNetworkType)networkType** 方法用来标记当前的 Adapter 对应的是哪个自定义平台,mogo 暂时支持最多 3 个自定义平台,他们的定义分别为：

//AdMoGoAdNetworkTypeCustomEventPlatform_1
//AdMoGoAdNetworkTypeCustomEventPlatform_2
//AdMoGoAdNetworkTypeCustomEventPlatform_3

将其对应作为返回值即可，具体方法如下：

```
+(AdMoGoAdNetworkType)networkType{
return      AdMoGoAdNetworkTypeCustomEventPlatform_1;
}
```

c: **-(void)getAd** 这个方法是用于初始化广告并开始请求的通知，一般的应该在此方法中初始化用户所需要添加平台的 AdView 等信息，与横幅不同这里需要区别的不是广告尺寸而是展示设备，即 iPhone 全屏/iPad 全屏，这个类型是和 MogoSDK 所支持的广告类型所对应的，具体 Mogo 所能使用的广告类型在 MogoSDK 文档和 Sample 中都有说明，这里不在赘述。
这个参数可以通过方法-(AdViewType)customAdapterWillgetAdAndGetAdViewType;获得。与此同时，在执行此方法后 MogoSDK 会为您初始化一些参数，所以即使用户不需要 AdViewType，也务必向此方法发送消息。

下面是初始化 AdMob 的示例：

```
//通知Mogo请求广告开始并且获得广告尺寸类型
//注：即便是不需要使用AdViewType,也要在此向
customAdapterWillgetAdAndGetAdViewType发一次消息
    AdViewType type =[self customAdapterWillgetAdAndGetAdViewType];

      if (type == AdViewTypeFullScreen || type==AdViewTypeiPadFullScreen) {
            gadinterstitial = [[GADInterstitial alloc] init];
            gadinterstitial.delegate = self;
            gadinterstitial.adUnitID = [key objectForKey:APPID_FS_F];
            GADRequest *request = [GADRequest request];
            request.testDevices = [NSArray arrayWithObjects:nil];
            [gadinterstitial loadRequest:request];

            //did start(requirement)
            [self adMoGoFSCustom:self didStartRequestAd:gadinterstitial];
```

```
        }else{

            [self adMoGoFSCustom:self didFailAd:nil];

         }
```

2．全屏广告请求成功失败等接口介绍

a：- (void)adMoGoFSCustom:(AdMoGoFullScreenAdapterCustomEvent *)adapter didStartRequestAd:(id)ad;
通知 MogoSDK 广告开始请求。

b: - (void)adMoGoFSCustom:(AdMoGoFullScreenAdapterCustomEvent *)adapter didReceiveInterstitialScreenAd:(id)ad;
通知 MogoSDK 广告请求成功,MogoSDK 也会通过接口 adsMoGoInterstitialAdReceivedRequest 通知广告接收成功，和其他平台一样，在这个回调后是展示全屏广告的时机。

c: - (void)adMoGoFSCustom:(AdMoGoFullScreenAdapterCustomEvent *)adapter didFailAd:(NSError *)error;
通知 MogoSDK 广告请求失败，SDK 会按照既定逻辑继续进行轮换。

d: - (void)adMoGoFSCustom:(AdMoGoFullScreenAdapterCustomEvent *)adapter didDismissScreen:(id)ad;
通知 MogoSDK 广告被关闭，此时 SDK 会根据逻辑继续或在预设时间后请求下一轮广告，

如果没有调用此接口，则不会再进行轮换。

e: - (void)adMoGoFSCustom:(AdMoGoFullScreenAdapterCustomEvent *)adapter willPresent:(id)ad;
通知 MogoSDK 广告将要被展示

f: - (void)adMoGoFSCustomCountClick
通知 MogoSDK 广告被点击，此时 SDK 会记录一次全屏的点击数。

注：以上为自定义平台接口，需要开发者在对于的时机调用的，而不

是提供给开发者的代理方法！

详细的使用方式请见Sample。