万普平台iOS广告SDK开发者手册 (Ver2.0.8)

平台介绍

万普世纪移动营销服务平台(以下称为“万普平台”)的iOS 版SDK 提供了一套现成的开发包及Demo源代码，便于开发者在iOS应用中方便的集成万普平台的各项功能，包括万普统计、积分墙、互动广告、插屏广告等功能。
本文档描述了标准版SDK 的用途与用法，并提供了示例代码。

1.6.7 版特别注意：

为方便通过App Store审核，万普iOS SDK进行了重构，主要修改掉一些敏感方法命名，带来不便敬请谅解。

引入SDK包

步骤1:
下载iOS版SDK包，将Frameworks目录中的 WPLib.framework 复制到自定义路径。

步骤2:
引入 WPLib.framework，在工程中选择 -> TARGETS->Summary->Linked Frameworks and Libraries (Xcode5环境下为: TARGETS->Linked Frameworks and Libraries),
点“ + ”, 再点击 “ Add Other” 选中下载的 WPLib.framework
在所有调用万普SDK接口的代码文件中引入头文件：
#import "WPLib/AppConnect.h"

其他需要添加的依赖包如下(请在系统包中自行寻找添加)：

AdSupport.framework

QuartzCore.framework

Security.framework

CoreTelephony.framework

SystemConfiguratioin.framework

Libz.dylib

加入接口代码

1.数据统计接口

该接口是所有其他接口能正常使用的基础，在每次应用启动时，必须调用该接口，才能保证获得准确的统计数据。
在程序的启动方法中加入(必须)： （WAPS_ID在万普后台添加应用后获取）

```
//指定WAPS_ID(请将1bf390a13d540df7bf72418498dfe503替换成自己的WAPS_ID)和发布渠道编号PID(appstore,91)
[AppConnect getConnect:@“1bf390a13d540df7bf72418498dfe503” pid:@“appstore”];
```

或者(两种初始化方式只能选择一种)：

```
[AppConnect getConnect:@“1bf390a13d540df7bf72418498dfe503” pid:@“appstore” userID:@“user_007”];
//userID 为开发者自己给用户分配的标识，在用户获取积分后服务器端通知使用。
```

注意：如需开通服务端通知，请联系客服索取接口文档和提交服务端通知地址。

2.推荐列表接口

推荐列表（也称Offer，广告墙）是万普平台提供的一种集中展示型广告。开发者可在应用中合适的位置加入“推荐应用”、“免费赚积分”等类似字样的功能，获取更高的广告收益或参与流量交换。添加如下代码，即可显示万普平台推荐应用列表：

2.1 默认推荐列表

1. 显示推荐列表：

```
[AppConnect showList:viewController]; //支持横竖屏自动切换,原showOffers方法
```

或者：

```
[AppConnect showList:viewController showNavBar:NO]; //不显示导航条,原showOffers方法
```

从积分墙返回的事件回调方法（可选）：

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onListClosed:) name:WP_LIST_CLOSED object:nil];
```

```
-(void)onListClosed:(NSNotification*)notifyObj{  //原onOfferClosed方法
        NSLog(@"列表已关闭");
}
```

2. 关闭推荐列表接口

```
[AppConnect closeList]; //关闭同时会有WP_LIST_CLOSED通知事件,原closeOffers方法
```

3. 自定义推荐列表导航条

开发者可用自定义的UINavigationBar代替推荐列表中默认的导航条，方法如下：

```
[AppConnect showList:viewController navBar:bar]; //原showOffers方法
```

例子代码(详细代码请参照DEMO):

```
UINavigationBar *bar = [[UINavigationBar alloc] initWithFrame:CGRectMake(0, 20, self.view.frame.size.width, 60)];
UINavigationItem *_navBarTitle = [[UINavigationItem alloc] initWithTitle:@"推荐应用"];
UIBarButtonItem *_navBarLeftBtn = [[UIBarButtonItem alloc] initWithTitle:@"关闭" style:UIBarButtonItemStyleBordered target:self action:@selector(closeBtn)];
    [_navBarTitle setLeftBarButtonItem:_navBarLeftBtn];
    bar.items = [NSArray arrayWithObject:_navBarTitle];
[AppConnect showList:viewController navBar:bar];
```

2.2 广告墙自定义方式实现

除了默认的调用方法，开发者也可选择使用API的方式读取出基础数据，自己加工界面(详细代码请参照Demo，界面部分可直接使用Demo->OfferCustom目录中源代码)

1. 引入必要的.h文件

```
#import "WPLib/WPListCustom.h"
```

2. 在所需页面.h文件设置代理协议<WPListCustomDelegate>

3. 在所需页面.m文件实例化WPListCustom

```
WPListCustom *listCustom = [[WPListCustom alloc] init];//在viewDidLoad方法中
listCustom.delegate = self;//设置当前对象为listCumtom的代理
[listCustom loadCustomData]; //调用WPListCustom的loadCustomData方法,请求广告墙数据
```

4. 实现代理协议中的4个方法,都为可选方法，收到数据结果的2个方法实现其一即可：

```
//参数:flg为真时,表明正在请求数据;flg为假时,数据请求完成.
- (void)isLoading:(BOOL)flg;
//参数:offersJson为下载完成后的字符串数据.
- (void)getCustomDataWithJson:(NSString *)offersJson;
//参数:offersArray为下载完成后的数组数据,数组中的内容为字典格式.
- (void)getCustomDataWithArray:(NSArray *)offersArray;
//参数:errorInfo为下载失败后的信息.
- (void)getCustomDataFaile:(NSString *)errorInfo;
```

5. 点击事件中提交点击内容的click_url到指定方法：

```
//参数:url为返回数据中click_url属性所对应的字符串.
- (void)listOnClick:(NSString *)url;
```

备注：

1. 以上功能在Demo中有完整实现，可直接复制使用
2. WapsDemo可以在ARC和非ARC模式中使用,建议使用ARC模式,如果使用非ARC需要开发者自己手动管理内存。

3.虚拟货币接口

在应用中合理设置虚拟货币及消费机制，可促进用户参与应用内购买或参与广告活动，增强应用粘性，大幅提升收益。如果您的应用开启了虚拟货币功能，需要使用该接口和服务器同步用户的虚拟货币余额。

(1)获取用户虚拟货币：

```
[AppConnect getPoints];
```

(2)花费用户虚拟货币：

```
[AppConnect spendPoints:(int)amount];
```

(3)奖励用户虚拟货币：

```
[AppConnect awardPoints:(int)amount];
```

(4)虚拟货币状态处理：

注意：WP_UPDATE_POINTS在所有虚拟货币操作成功后都进行通知，1.6.1版本之前的WP_GET_POINTS_SUCCESS和WP_GET_POINTS_FAILED只在调用getPoints后通知，其他积分操作不做getPoints通知

```
//只要有虚拟货币操作成功就有此通知，1.6.1新增
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUpdatedPoints:) name:WP_UPDATE_POINTS object:nil];
//只有getPoints调用成功时通知
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onGetPointsSuccess:) name:WP_GET_POINTS_SUCCESS object:nil];
//只有spendPoints调用成功时通知
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onSpendPointsSuccess:) name:WP_SPEND_POINTS_SUCCESS object:nil];
//只有awardPoints调用成功时通知
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onAwardPointsSuccess:) name:WP_AWARD_POINTS_SUCCESS object:nil];
```

虚拟货币通知处理方法：

```
-(void)onUpdatePoints:(NSNotification*)notifyObj{
    WPUserPoints *userPoints = notifyObj.object;
    NSString *pointsName=[userPoints getPointsName];
    int  pointsValue=[userPoints getPointsValue];
}
-(void)onGetPointsSuccess:(NSNotification*)notifyObj{
    WPUserPoints *userPoints = notifyObj.object;
    NSLog(@"成功获取积分:%@", [userPoints getPointsValue]);
}
-(void)onSpendPointsSuccess:(NSNotification*)notifyObj{
        NSLog(@"成功消费积分:%@", notifyObj.object);
 }
-(void)onAwardPointsSuccess:(NSNotification*)notifyObj{
        NSLog(@"成功赚取积分:%@", notifyObj.object);
 }
```

4.互动广告接口

互动广告是一个显示在应用内固定位置高度为50像素广告条，将自动显示万普平台提供的Banner广告。使用下面一行代码即可显示万普互动广告条：

```
[AppConnect displayAd:viewController];
```

以上默认情况下，广告条将显示在屏幕顶端，默认为320*50像素。如需在其他位置显示广告条，可使用以下方法在指定位置创建广告：

```
[AppConnect displayAd:viewController showX:(int)x showY:(int)y];
```

关闭互动广告：

```
[AppConnect closeBannerAd];
```

5.插屏广告接口

插屏广告是在1.2版本后新加入的广告形式，将弹出显示万普平台提供的插屏广告。下面是使用方法:

步骤1 初始化

初始化(预先加载)广告数据:

```
[AppConnect initPop];
```

注意：请在 [AppConnect getConnect:[WAPS_ID] pid:[channel]];方法之后调用。

步骤2 显示插屏广告

```
[AppConnect showPop:viewController];
```

步骤3 手动关闭插屏广告(可选)

```
[AppConnect  closePop];
```

步骤4 处理插屏广告状态

声明插屏广告各种状态通知处理(可选,附录中有所有通知说明)

6.信息流广告接口

信息流广告是在2.0.6版本后新加入的广告形式，可以在列表总显示万普平台提供的信息流广告。调用方式采用API形式(可以参考服务器文档接口来获取数据)下面是使用方法:

步骤1 获取icon接口

```
NSString *strUrl = [NSString stringWithFormat:@"%@&idfa=%@",imageViewUrl,[self getIDFA]];
```

注意：请在 [AppConnect getConnect:[WAPS_ID] pid:[channel]];方法之后调用。

步骤2 获取大图接口，横幅图片需要在地址后面添加参数`res_type=16_9`或`res_type=4_3`，获取不同比例的图片

```
NSString *strUrlBanner = [NSString stringWithFormat:@"%@&idfa=%@&%@",imageViewUrlBanner,[self getIDFA],@"res_type=16_9"];
```

7.在线配置接口

开发者可以通过万普平台管理后台的“在线配置”功能设置各种参数值，便于在线修改应用的各项配置。其中参数key为在线配置的参数ID，通过次方法可获取到对应的值。

注意：该方法将在统计器初始化成功时返回所有参数值，缓存到本地调用。因此在线参数的初始化需要在数据统计接口的连接成功回调函数中进行。

计数器连接成功的方法中获取:

```
-(void)onConnectSuccess:(NSNotification*)notifyObj{
    NSLog(@"连接成功");
  //启动并连接成功后获取在线配置,key设置为在线参数ID
  NSString* value=[[AppConnect getConfigItems] objectForKey:@"key"];
}
```

8.用户反馈接口

用户反馈接口的作用便于开发者收集到用户对应用的使用反馈，用户反馈的内容开发者可在万普平台管理后台的“用户反馈”中进行管理，也可通过管理后台给用户及时反馈。
调用方法：

```
[AppConnect showFeedBack:viewController];
```

9.远程推送接口

步骤1:p12证书的制作,请参考压缩包中的远程推送说明文档(万普世纪).pdf

步骤2:在开发者后台上传p12证书,如下如所示:

步骤3:在系统函数入口处,必须调用以下接口

```
[AppConnect setupWithOption:launchOptions viewController:rootViewController];
//launchOptions 为应用函数入口的参数，rootViewController为根控制器。
[AppConnect registerForRemoteNotificationTypes];
//此函数为注册远程通知类型,分别为sounds,alter,badge。
```

步骤4:在以下系统接口中调用处理推送信息的接口

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken{
   [AppConnect registerDeviceToken:deviceToken];
   // 上传deviceToken值到服务器
}

- (void)application:(UIApplication *)application
didFailToRegisterForRemoteNotificationsWithError:(NSError *)error{
   [AppConnect handFailToRegisterForRemoteNotificationsWithError:error];
   //处理deviceToken出错的接口
}

-  (void)application:(UIApplication *)application didReceiveRemoteNotification:
   (NSDictionary *)userInfo {
    [AppConnect handleRemoteNotification:userInfo];
     //处理远程推送的信息的接口。
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult
result))completionHandler{
    [AppConnect handleRemoteNotification:userInfo];
    //ios7以上要调用的函数。
}
```

步骤5:可以在系统接口中通过参数userinfo字典或者launchOptions字典，获取后台添加的键值信息

10.事件通知接口

调用例子:

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUpdatedPoints:) name:[] object:nil];
```

可获取的事件通知(所有事件KEY在1.6.7版本里做了重构，请注意):

事件KEY	类型	说明	返回对象
WP_CONNECT_SUCCESS	计数器	计数器链接成功	nil
WP_CONNECT_FAILED	计数器	计数器链接失败	nil
WP_LIST_CLOSED	广告墙	推荐列表关闭	nil
WP_UPDATE_POINTS	积分	所有积分操作成功均有此通知	WapsUserPoints *user = notifyObj.object;
WP_GET_POINTS_SUCCESS	积分	获取积分成功,仅getPoints调用通知	WapsUserPoints *user = notifyObj.object;
WP_GET_POINTS_FAILED	积分	获取积分失败,仅getPoints调用通知	nil
WP_SPEND_POINTS_SUCCESS	积分	消费积分成功	NSString * ret=notifyObj.object //消费数额
WP_SPEND_POINTS_FAILED	积分	消费积分失败	nil
WP_AWARD_POINTS_SUCCESS	积分	奖励积分成功	NSString * ret=notifyObj.object //奖励数额
WP_AWARD_POINTS_FAILED	积分	奖励积分失败	nil
WP_POINTS_EARNED	积分	赚取积分	nil
WP_BANNERAD_SHOW	广告条	广告条成功显示	nil
WP_BANNERAD_CLICK	广告条	广告条点击	nil
WP_BANNERAD_FAILED	广告条	广告条显示失败	NSString * ret=notifyObj.object //NO代表服务端关闭
WP_BANNERAD_CLOSED	广告条	广告条手动关闭	nil
WP_POPAD_INIT_SUCESS	插屏广告	插屏广告初始化成功	nil
WP_POPAD_INIT_NULL	插屏广告	插屏广告无广告内容	nil
WP_POPAD_INIT_FAILED	插屏广告	插屏广告初始化失败	nil
WP_POPAD_SHOW_SUCESS	插屏广告	插屏广告显示成功	nil
WP_POPAD_SHOW_FAILED	插屏广告	插屏广告显示失败	NSString * ret=notifyObj.object //NO代表服务端关闭
WP_POPAD_CLOSED	插屏广告	插屏广告关闭	nil
WP_POPAD_CLICKED	插屏广告	插屏广告点击	nil
WP_PUSH_UPLOADDEVICE_SUCCESS	推送	上传DeviceToken值成功	nil
WP_PUSH_UPLOADDEVICE_FAILED	推送	上传DeviceToken值失败	NSString *objStr = [notify object];