

MagicWindowSDK

使用文档 (iOS)

——mLink

目录

一	集成准备	4
1.1	获取魔窗 AppKey	4
1.2	导入 SDK	4
1.2.1	使用 Cocoapods 安装 SDK	4
1.2.2	下载 SDK 并集成	5
1.3	初始化 SDK	7
二	mLink 的设置	8
2.1	mLink 基础配置	8
2.1.1	配置 App 的 URL Scheme	8
2.1.2	配置 Universal link	8
2.2	配置相关代码	11
2.2.1	通过 URL Scheme 和 Universal link 唤起 app	11
2.2.2	配置深度链接，一键直达具体页	12
2.2.3	场景还原	14
三	高级设置	14
3.1	ABA 跳转	14
3.1.1	ABA 的实现原理和具体实现	14
3.1.2	ABA 返回浮层按钮	14
3.2	无码邀请	16
四	基础指标统计	17

4.1	页面统计.....	17
4.2	计数统计.....	17
4.3	设置用户信息.....	18
五	使用多渠道分析.....	18
六	注意事项.....	19
6.1	如何防止 app 因获取 IDFA 被 App Store 拒绝.....	19
6.2	支持 ATS.....	20
6.3	如何验证 SDK 已经对接成功.....	20
七	FAQ.....	20

一 集成准备

1.1 获取魔窗 AppKey

登录魔窗后台管理 (<http://mgnt.magicwindow.cn/>), 按照步骤提示注册应用, 可获得 AppKey。



1.2 导入 SDK

导入 SDK 有以下两种方法, 第一种使用 Cocoapods 安装 SDK, 第二种直接下载 SDK 并集成, 选择其中一种即可。

注意: SDK 中包含了微信分享 SDK, 如果您的工程里面也同样包含了微信分享 SDK, 请将重复的删除, 保留最新版的微信分享 SDK 即可, 不会影响 SDK 的正常使用。

1.2.1 使用 Cocoapods 安装 SDK

使用 Cocoapods 可以方便的统一管理第三方库: <https://cocoapods.org>

安装 Cocoapods, 并在项目根目录下创建 Podfile 文件, 在 Podfile 文件中添加如下内容:

```
pod 'MagicWindowSDK'  
//如果使用 bitcode, 使用 pod 'MagicWindowSDKBitcode'
```

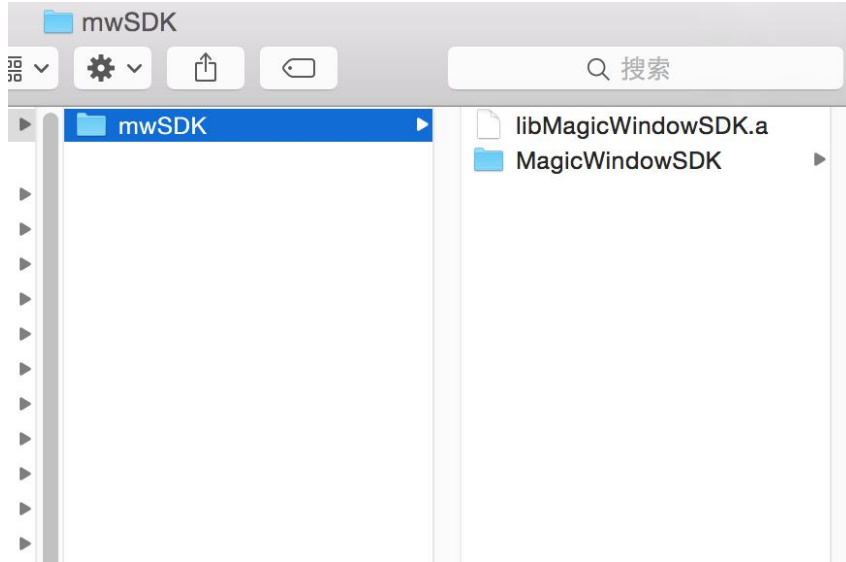
在 terminal 下运行命令如下:

```
pod install
```

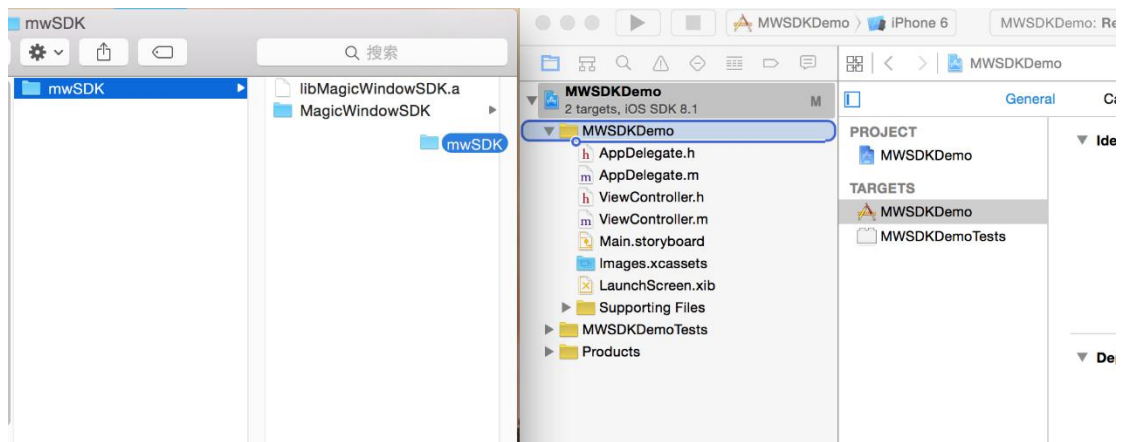
1.2.2 下载 SDK 并集成

(1) 下载 SDK 并集成

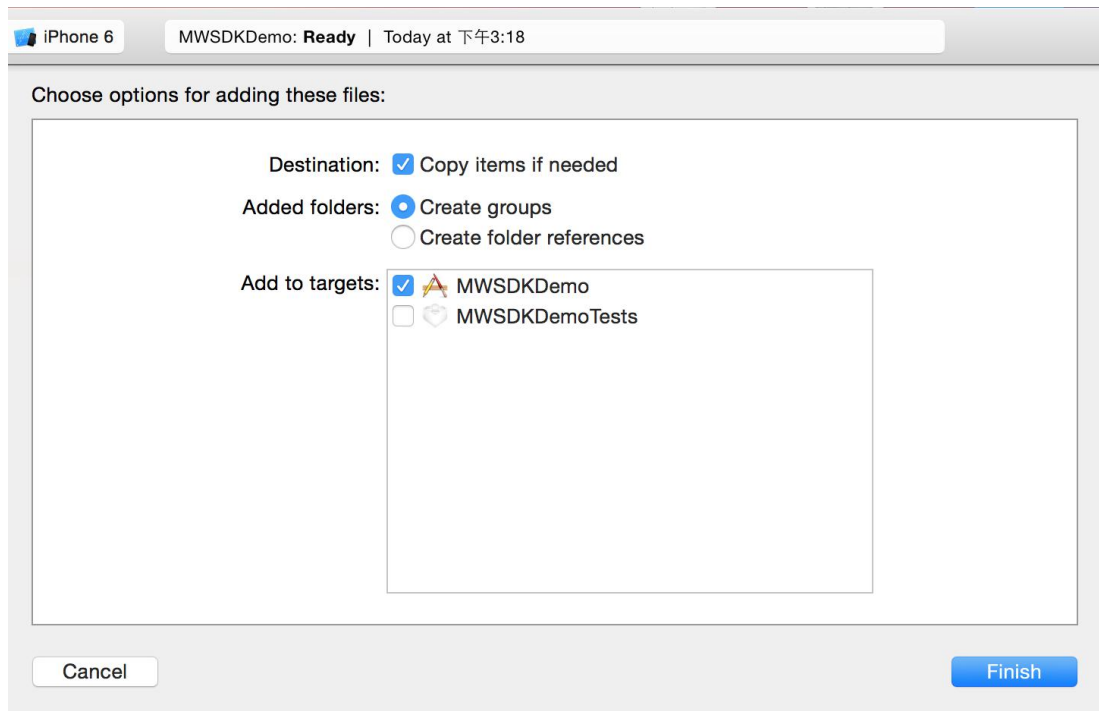
下载并解压最新版本 SDK 压缩包，解压后如下图：



将下载的 SDK 文件解压，拖动里面的 mwSDK 文件夹到工程中，如下图：



拖到工程中后，弹出以下对话框，勾选 **“Copy items into destination group's folder(if needed)”**，并点击 **“Finish”** 按钮，如图：



注意：请务必在上述步骤中选择 **“Create groups for any added folders”** 单选按钮组。如果您选择 **“Create folder references for any added folders”**，一个蓝色的文件夹引用将被添加到项目并且将无法找到它的资源。

(2) 添加依赖库

如果使用了 CocoaPods 集成的 SDK，可以忽略此步骤

AdSupport.framework

CoreTelephony.framework

CoreGraphics.framework

CoreFoundation.framework

SystemConfiguration.framework

CoreLocation.framework

CFNetwork.framework

Security.framework

WebKit.framework

ImageIO.framework

libz.tbd

libsqlite3.0.tbd

libc++.tbd

(3) 设置 Other Linker Flags

如果使用的是最新版本的微信分享 SDK , 需要在你的工程文件中选择 Build Setting , 在"Other Linker Flags"中加入"-Objc -all_load" , 详情见微信官方文档 :

https://open.weixin.qq.com/cgi-bin/showdocument?action=dir_list&t=resource/res_list&verify=1&id=open1419319164&token=&lang=zh_CN

1.3 初始化 SDK

在 AppDelegate 中 , 增加头文件的引用

```
#import "MWApi.h"
```

在- (BOOL)application: didFinishLaunchingWithOptions:方法中调用 registerApp 方法来初始化 SDK , 如下图所示 :

```
#import "AppDelegate.h"
#import "MWApi.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    //如果您创建应用时使用 storyboard 可以省略此步骤
    self.window = [[UIWindow alloc] initWithFrame:[UIScreen mainScreen]
bounds]];

    self.window.rootViewController = viewController;
    [self.window makeKeyAndVisible];

    //初始化 SDK , 必写
    [MWApi registerApp:AppKey];

    return YES;
}
```

注意 : 必须设置 rootViewController , 否则会导致无法弹出活动界面。如果您创建应用时使用 storyboard 可以省略此步骤 , 系统会自动设置 rootViewController。

二 mLink 的设置

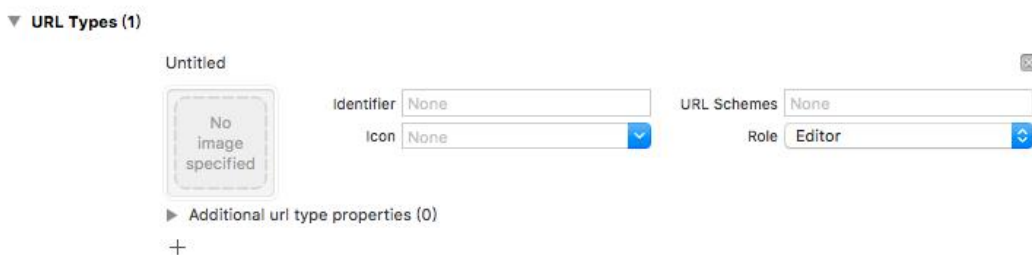
mLink，可以将 App 中具体的内容页面作为一个服务（比如滴滴的打车服务），以链接的形式分享出来，通过这个链接，可以从外部唤起 App 并自动进入 App 中的指定服务页。

2.1 mLink 基础配置

2.1.1 配置 App 的 URL Scheme

iOS 系统中 App 之间是相互隔离的，通过 URL Scheme，App 之间可以相互调用，并且可以传递参数。

选中 Target - Info - URL Types，配置 URL Scheme(比如: magicWindow)



在 Safari 中输入 URL Scheme :// （比如: magicWinodw://）如果可以唤起 App，说明该 URL Scheme 配置成功。

2.1.2 配置 Universal link

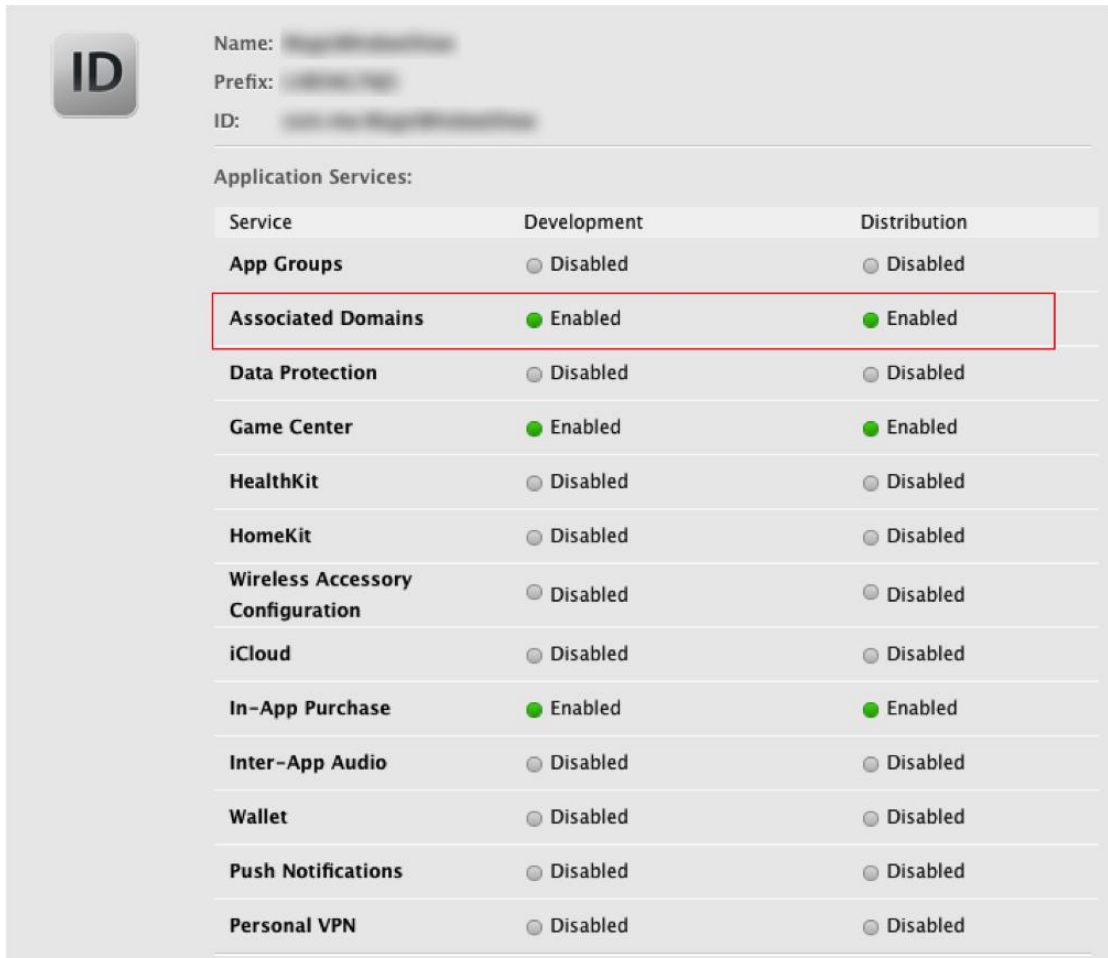
Universal link 是 iOS9 的一个新特性，通过 Universal link，App 可以无需打开 Safari，直接从微信中跳转到 App，真正的实现一键直达。如果使用 URL Scheme 的话，需要先打开 Safari，用户体验变得很差，如果 App 未安装，还会出现错误对话框：



所以，我们强烈推荐配置 Universal link，而通过魔窗来配置 Universal link 也变得非常的简便。

(1) 配置 developer.apple.com 的相关信息

登录 <https://developer.apple.com> , 选择 Certificate, Identifiers & Profiles , 选择相应的 AppID , 开启 Associated Domains



The screenshot shows the 'Application Services' configuration page for an App ID. The 'Associated Domains' row is highlighted with a red box, indicating it is enabled for both Development and Distribution.

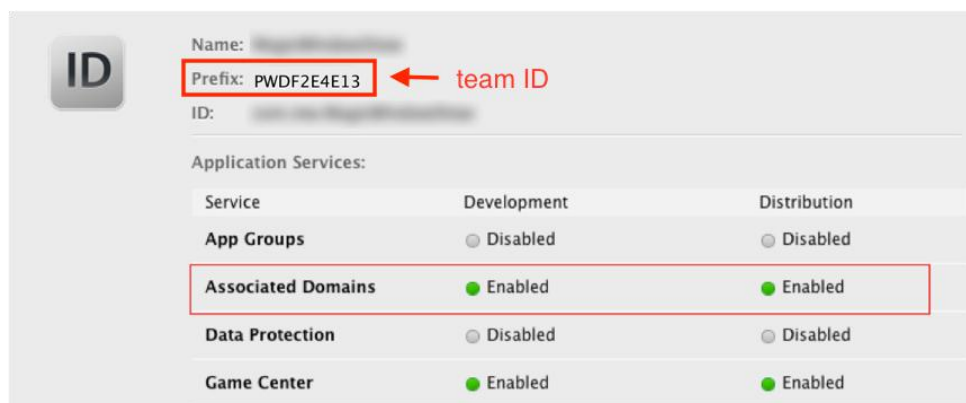
Service	Development	Distribution
App Groups	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Associated Domains	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Data Protection	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Game Center	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
HealthKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
HomeKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wireless Accessory Configuration	<input type="radio"/> Disabled	<input type="radio"/> Disabled
iCloud	<input type="radio"/> Disabled	<input type="radio"/> Disabled
In-App Purchase	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Inter-App Audio	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wallet	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Push Notifications	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Personal VPN	<input type="radio"/> Disabled	<input type="radio"/> Disabled

注意：当 AppID 重新编辑过之后，需要更新相应的 provisioning Profiles。

(2) 在后台填写相关 App 的 mLink 配置信息

产品信息	App信息	魔窗位信息
iOS 应用		
<p>* 应用名称: <input type="text" value="请输入应用名称"/></p> <p>* ? Bundle ID: <input type="text" value="请输入Bundle ID"/></p> <p>以下配置用于mLink, 当您使用mLink时, 务必填下。 mLink是什么?</p> <div><p>? URI Scheme: <input type="text" value="请输入URI Scheme"/></p><p>下载地址: <input type="text" value="请输入http://格式的下载地址"/></p><p>以下配置适用于从微信一键唤起app</p><p>? 是否配置Universal Link: <input checked="" type="radio"/> 是 <input type="radio"/> 否</p><p>? * Team ID: <input type="text" value=""/> 我已有Universal Link</p><p>! 分配域名: s.mlinks.cc 备注: 您必须在Xcode的capabilities里添加魔窗的域名 (applinks:s.mlinks.cc), 详情请参考iOS SDK集成文档。请注意每次新注册的App分配的魔窗域名可能不一样。</p></div>		

team ID 可以从你的苹果开发账号页面获取，如图：



(3) 配置 Xcode

在 Xcode 中选择相应的 target，点击 Capabilities tab，开启 Associated Domains，在里面添加魔窗的域名（applinks:xxxx），域名的值在后台获取，请注意每次新注册的 App 分配的魔窗域名可能不一样。如图：

以下配置适用于从微信一键唤起app

是否配置Universal Link: ☒ 是 ☐ 否

Team ID:

[我已有Universal Link](#)

分配域名: **s.mlinks.cc**
备注: 您必须在Xcode的capabilities里添加魔窗的域名 (**applinks:s.mlinks.cc**), 详情请参考iOS SDK集成文档。请注意每次新注册的App分配的魔窗域名可能不一样。

Associated Domains ☒

Domains: **applinks:s.mlinks.cc**

Steps: ☒ Add the Associated Domains entitlement to your entitlements file
☒ Add the Associated Domains feature to your App ID.

2.2 配置相关代码

2.2.1 通过 URL Scheme 和 Universal link 唤起 app

(1) 在 AppDelegate 中的 openURL 方法中调用, 用来处理 URL Scheme

```
+ (void)routeMLink:(nonnull NSURL *)url;
```

示例如下:

```
//iOS9 以下
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    //必写
    [MWApi routeMLink:url];
    return YES;
}

//iOS9+
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(nonnull
NSDictionary<NSString *,id> *)options
{
    //必写
    [MWApi routeMLink:url];
}
```

```
return YES;
}
```

注意：当需要在 `openURL` 这个方法中处理第三方回调的时候（比如支付宝回调，微信回调等），请注意区分，比如

```
if(支付宝回调) return 支付宝回调处理逻辑;
else if(微信回调) return 微信回调处理逻辑;
else if(其他第三方回调) return 其他第三方回调处理逻辑;
else return [MWApi routeMLink:url];
```

（2）在 AppDelegate 中的 `continueUserActivity` 方法中调用，用来处理 Universal link

```
+ (BOOL)continueUserActivity:(nonnull NSUserActivity *)userActivity;
```

示例如下：

```
- (BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity restorationHandler:(void (^)(NSArray * _Nullable))restorationHandler
{
    //如果使用了 Universal link ，此方法必写
    return [MWApi continueUserActivity:userActivity];
}
```

2.2.2 配置深度链接，一键直达具体页

本模块实现的功能是通过深度链接跳转到 APP 内的详情页面，若想要使用如下功能，请务必将“mlink 基本配置”部分全部实现。

mLink服务配置

产品名称: 魔窗

* mLink服务名称:

* mLink服务key:

mLink key是mLink服务的唯一标识

① 页面key只能包含英文字母,数字或者下划线开头,且长度不得大于30

② iOS URI:

scheme://

② Android URI:

scheme://

在 AppDelegate 中的 didFinishLaunchingWithOptions 方法中调用

```
/**
 * 注册一个 mLink handler，当接收到 URL 的时候，会根据 mLink key 进行匹配，当匹配成功会调用相应的 handler，在相应的 handler 中自行处理跳转逻辑
 * @param key : mLink key 是 mLink 服务的唯一标识，在魔窗后台设置，如上图：
 */
+ (void)registerMLinkHandlerWithKey:(NSString *)key handler:(CallBackMLink)handler;
```

示例如下

```
[MWApi registerMLinkHandlerWithKey:@"mLinkKey" handler:^(NSURL *url, NSDictionary *params) {
    //自行处理跳转逻辑，示例如下：
    ViewController *rootVC = (ViewController *)self.window.rootViewController;

    CategoryDetailViewController *detailVC = [tabVC.storyboard instantiateViewControllerWithIdentifier:@"CategoryDetailView"];
    detailVC.name1 = params[@"name1"];
    detailVC.name2 = params[@"name2"];
    detailVC.name3 = params[@"name3"];
    [rootVC pushViewController:detailVC animated:YES];
}];
```

2.2.3 场景还原

场景还原：用户下载后无需任何操作，直达指定内页，大幅提升转化率

mLink 基础功能中包含了场景还原，实现了“一键唤起具体页”功能即可实现场景还原，仅需在处理跳转逻辑的时候特殊处理下启动页或者登录等事件。

三 高级设置

3.1 ABA 跳转

通过魔窗位连接 App，实现 App 之间场景式的跳转及返回。

3.1.1 ABA 的实现原理和具体实现

详情见：<http://www.magicwindow.cn/doc/mw-AB.html>

3.1.2 ABA 返回浮层按钮

用户可通过“返回浮层”随时从跳转方 App 返回来源方 App。

(1) 初始化返回浮层

在 AppDelegate 中，增加头文件的引用

```
#import "MWFloatView.h"
```

初始化

```
/**
 * 初始化浮层 buttonView
 * @param y buttonView 的 y 轴的起始位置
 * @return void
 */
- (id)initWithYPoint:(float)y;
```

(2) 设置返回浮层滑动区域

```
/**
 * 是否允许 buttonView 上下滑动
 * @param enable YES 允许滑动，NO 不允许滑动，默认允许滑动
 * @return void
 */
- (void)dragEnable:(BOOL)enable;
```

```
/**
 * 设置 buttonView 的上下滑动的区域范围，默认是 0 ~ deviceHeight
 * @param minY : buttonView 的允许滑动 y 轴的最小值，maxY : buttonView 的允许滑动 y 轴的最大值
 * @return void
 */
- (void)setMinY:(float)minY andMaxY:(float)maxY;
```

(3) 展示返回浮层

```
/**
 * 将 buttonView 添加到 view 上
 * @return void
 */
- (void)show;
```

(4) 隐藏返回浮层

```
/**
 * 将 buttonView 隐藏掉
 * @return void
 */
- (void) hide;
```

(5) 设置返回应用 A 时所需参数

```
/**
 * 设置从当前 App 返回应用 A 的时候需要的参数
 * @param params 返回 A 时需要传入的参数
```

```
* @return void
*/
- (void)setReturnOriginAppParams:(nonnull NSDictionary *)params;
```

(6) 返回浮层的 debug 模式

```
/**
 * 是否开启 debug 模式，一旦开始 debug 模式，在不需要返回的时候也会展示 button
View，供开发者参考 buttonView 的展示样式
 * @param enable YES 开启，NO 不开启，默认不开启
 * @return void
 */
- (void)debugEnable:(BOOL)enable;
```

(7) 返回浮层的示例

```
MWFloatView *view = [[MWFloatView alloc] initWithYPoint:300];
//[view debugEnable:YES];
[view show];
```

3.2 无码邀请

老用户分享 App 内容给新用户，新用户通过老用户分享的 H5 页完成安装 App 注册转化，老用户即可积累拉新成就，全程无需邀请码。奖励操作由 App 自行完成。

注意：若想要使用无码邀请的功能，请务必将 mlink 一键直达的功能全部实现。

```
/**
 * 获取无码邀请中传回来的相关值
 * @param paramKey，比如:u_id
 * @return id 返回相应的值
 */
+ (nullable id)getMLinkParam:(nonnull NSString *)paramKey;
```


四 基础指标统计

4.1 页面统计

统计某个页面的访问情况：

标记一次页面访问的开始

+ (void)pageviewStartWithName:(NSString *)name;

参数：name 页面名

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [MWApi pageviewStartWithName:@"homePage"];
}
```

标记一次页面访问的结束

+ (void)pageviewEndWithName:(NSString *)name;

参数：name 页面名

```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
    [MWApi pageviewEndWithName:@"homePage"];
}
```

(注意：pageviewStartWithName 和 pageviewEndWithName 要成对匹配使用才能正常统计页面情况)

4.2 计数统计

统计指定行为被触发的次数

```
[MWApi setCustomEvent:@"eventId" attributes:@{@"type":@"pan",@"color":
@"white",@"price":@"3.0"}];
```

(注意：eventId 需要先在魔窗后台管理上注册，才能参与正常的数据统计)

4.3 设置用户信息

配置用户的基本信息

登录的时候，设置用户信息

```
+ (void)setUserPhone:(NSString *)userPhone;
```

```
+ (void)setUserProfile:(MWUserProfile *)user;
```

退出登录的时候，取消当前的用户信息

```
+ (void)cancelUserProfile;
```

(注意：设置 MWUserProfile 的时候，用户唯一标识 mwUserId 不能为空)

五 使用多渠道分析

如果您要对 APP 不同的发布渠道进行统计，不需要在魔窗后台创建多个 APP，只需要设置不同的渠道即可。如果您只有 App Store 一个分发渠道，则不再需要做设定，我们会默认标记为 App Store。

例如您在 91 助手发布，需要统计 91 渠道：

```
[MWApi setChannelId:@"channelId"];
```

(注意：channelId 需要先在魔窗后台管理上注册，才能参与正常的数据统计)

六 注意事项

6.1 如何防止 app 因获取 IDFA 被 App Store 拒绝

如果您的 app 使用魔窗 SDK 而未集成任何广告服务 ,但需要跟踪广告带来的激活行为 ,
请按照下图填写 App Store 中的 IDFA 选项 (勾选 2 , 3 , 4):

Advertising Identifier

Does this app use the Advertising Identifier (IDFA)?

The **Advertising Identifier (IDFA)** is a unique ID for each iOS device and is the only way to offer targeted ads. Users can choose to limit ad targeting on their iOS device.

If your app is using the Advertising Identifier, check your code—including any third-party code—before you submit it to make sure that your app uses the Advertising Identifier only for the purposes listed below and respects the Limit Ad Tracking setting. If you include third-party code in your app, you are responsible for the behavior of such code, so be sure to check with your third-party provider to confirm compliance with the usage limitations of the Advertising Identifier and the Limit Ad Tracking setting.

This app uses the Advertising Identifier to (select all that apply):

- ☐ Serve advertisements within the app
- ☒ Attribute this app installation to a previously served advertisement
- ☒ Attribute an action taken within this app to a previously served advertisement

If you think you have another acceptable use for the Advertising Identifier, [contact us](#).

Limit Ad Tracking setting in iOS

- ☒ I, W.Feng Xiao, confirm that this app, and any third party that interfaces with this app, uses the Advertising Identifier checks and honors a user's Limit Ad Tracking setting in iOS and, when it is enabled by a user, this app does not use Advertising Identifier, and any information obtained through the use of the Advertising Identifier, in any way other than for "Limited Advertising Purposes" as defined in the [iOS Developer Program License Agreement](#).

- (1) Serve advertisements within the app
服务 app 中的广告。如果你的 app 中集成了广告 , 你需要勾选这一项。
- (2) Attribute this app installation to a previously served advertisement
跟踪广告带来的安装。
- (3) Attribute an action taken within this app to a previously served advertisement
跟踪广告带来的用户的后续行为。
- (4) Limit Ad Tracking setting in iOS
这一项下的内容其实就是对你的 app 使用 idfa 的目的做下确认 , 只要你选择了采集 idfa , 那么这一项都是需要勾选的。

6.2 支持 ATS

苹果：2017 年 1 月 1 日后所有 iOS 应用必须启用 ATS。

SDK 已启用 ATS。

6.3 如何验证 SDK 已经对接成功

如何验证 SDK 已经对接成功，请参看

<http://documentation.magicwindow.cn/?mlink-integration-validation>

七 FAQ

FAQ : <https://github.com/magicwindow/mw-sdk-faq>