目錄

Introduction

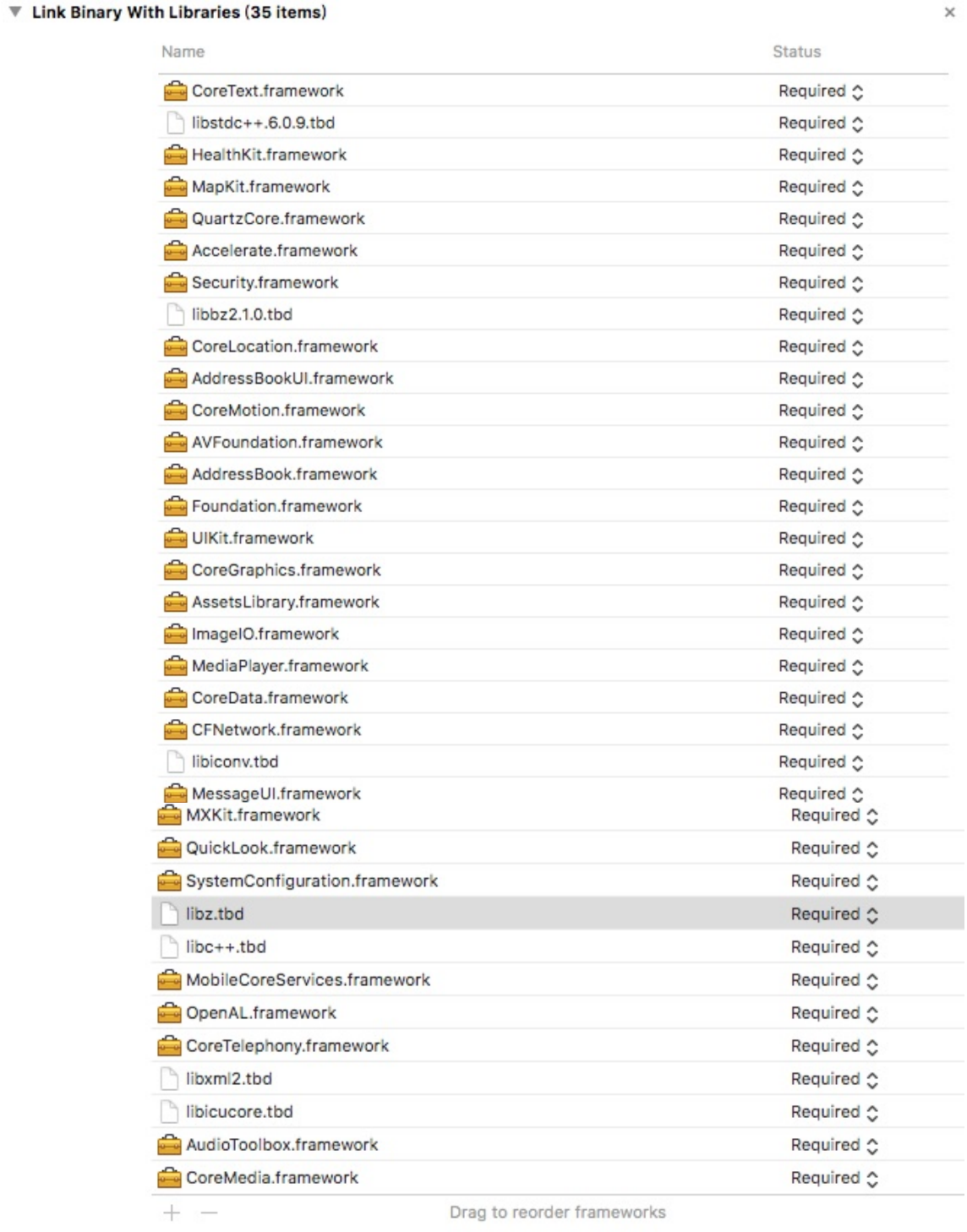
iOS快速集成指南

下载sdk

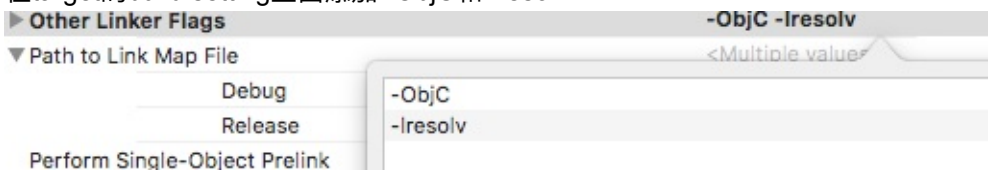
导入sdk以及工程设置

1. 拷贝MXKit.framework、MXKitResources.bundle和MXKit_Version.txt到相应得工程目录下，并且添加到工程里

2. 在工程文件的Build Phases 里面添加用到的Libraries:



3. 在target的build setting里面添加 -ObjC和-lresolv



编码实现

使用MXKit.framework需要按照以下流程实现：

1.引入头文件

```
#import <MXKit/MXKit.h>
```

2.初始化

```
//将宏定义设置为1
#define MX_DISABLE_SHARE_EXTENSION 1
MXKit *MXObj = [MXKit shareMXKit];
//将主程序的windows传到MXKit里面去，MXKit里面的一些UI操作会用到
[MXObj setWindowToMXKit:self.window];
#ifdef MX_DISABLE_SHARE_EXTENSION
[MXObj initDisableShareExtension:MX_DISABLE_SHARE_EXTENSION];
#endif
//MX_URL:敏行server的地址 e.g @"http://www.minxing365.com"
//MX_PORT:敏行server的端口 e.g @"80"
//MX_MQTT_URL: mqttserver的地址 e.g @"www.minxing365.com"
//MX_MQTT_PORT: mqtt server的端口 e.g @"1883"
[MXObj init:MX_URL withPort:MX_PORT withMqttUrl:MX_MQTT_URL withMqttPort:MX_MQTT_PORT];
//初始化umeng，这个是可选操作
[MXObj initUmeng:UMENG_KEY withChannel:UMENG_CHANNEL];
```

3.调用登陆接口

在执行任何操作之前需要先调用敏行的登陆接口获取到人员信息。
我们提供2个login的接口，一个是第一次登陆输入用户名和密码，另一个是已经登陆的用户下次再启动app直接登陆的接口
1.需要用户名密码：

```
MXKit *MXObj = [MXKit shareMXKit];
[MXObj login:name withPassword:password];
```

2.已经登陆过的用户直接登陆

```
MXKit *MXObj = [MXKit shareMXKit];
[MXObj login:^(id result, MXError *error){
    if(result && !error) {
        //登陆成功之后进行其他的操作
    }
}];
```

3.登录成功后回调

```
[MXObj registLoginCallback:^(id result, MXError *error){
      if(result && !error)
        {
          [self initapp];
        }
     }];
```

4.向敏行server注册设备

当app获取到deviceToken之后需要向敏行server注册

```
-(void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDevic
eToken:(NSData *)deviceToken {
    NSString *fullDeviceToken = [NSString stringWithFormat:@"%@",deviceToken];
    NSString *deviceTokenStr = [[fullDeviceToken substringWithRange:NSMakeRange(1, [fullDe
viceToken length]-2)]stringByReplacingOccurrencesOfString:@" " withString:@""];
    //这里用户可以把deviceToken存储起来，需要的时候向敏行server注册，也可以直接注册
    [[MXKit shareMXKit] registDeviceTokenToserver:deviceTokenStr];
}
//如果是模拟器，失败的情况下也需要想敏行server注册
- (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWi
thError:(NSError *)error {
    [[MXKit shareMXKit] registDeviceTokenToserver:nil];
}
```

5将主app的系统delegate传递给kit

```
//定义
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictiona
ry *)launchOptions;//应用程序启动后，要执行的委托调用
-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NS
String *)sourceApplication annotation:(id)annotation;//应用之间跳转
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDevi
ceToken:(NSData *)deviceToken;//向服务器注册token
- (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWi
thError:(NSError *)error;如果是模拟器，失败的情况下也需要想敏行server注册
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary
 *)userInfo;//推送
- (void)applicationWillResignActive:(UIApplication *)application;//应用程序将要由活动状态切换
到非活动状态时执行的委托调用，如按下home 按钮，返回主屏幕，或全屏之间切换应用程序等。
- (void)applicationDidEnterBackground:(UIApplication *)application;//在应用程序已进入后台程序
时，要执行的委托调用。所以要设置后台继续运行，则在这个函数里面设置即可。
- (void)applicationWillEnterForeground:(UIApplication *)application;//在应用程序将要进入前台
时(被激活)，要执行的委托调用，与applicationWillResignActive 方法相对应。
- (void)applicationDidBecomeActive:(UIApplication *)application;//在应用程序已被激活后，要执
行的委托调用，刚好与  applicationDidEnterBackground 方法相对应。
- (void)applicationWillTerminate:(UIApplication *)application;/在应用程序要完全退出的时候，要
```

```
执行的委托调用。
-(void)application:(UIApplication *)application performFetchWithCompletionHandler:(void (^
)(UIBackgroundFetchResult))completionHandler;
//示例
//在appdelegate.m里，各个api的使用方法相同，这里以进入后台的事件为例
- (void)applicationDidEnterBackground:(UIApplication *)application {
    MXKit *MXObj = [MXKit shareMXKit];
    [MXObj applicationDidEnterBackground:application];
}
```

至此用户就可以调用我们各个模块的api来进行集成了。

常见问题

1.如何从应用中心启动自己的native页面

扩展mxclient_ios中应用中心入口部门的代码添加的listener,并且可以根据appid（ocuid）来判断是启动的哪个公众号

IM集成指引

1.引入头文件

```
#import <MXKit/MXChat.h>
```

2.获取IM的viewController

```
//定义
-(UIViewController *)getChatViewController;
//示例
UIViewController *chat = [[MXChat sharedInstance] getChatViewController];
//你可以push这个viewController
//设置导航栏
UINavigationController *chatNav=[[UINavigationController alloc]initWithRootViewController:
chat];
```

3.IM模块的详细api使用

如果IM首页是在在UITabbarItem上需要设置tabbar的高度

```
//定义
-(void)setTabbarHeight:(int)tabbarHeight;
//示例
[[MXChat sharedInstance] setTabbarHeight:tabbarHeight];
```

发起聊天，自带通讯录选人界面

```
//定义
-(void)startChat;
//示例
[[MXChat sharedInstance] startChat];
```

发起聊天，不带通讯录选人界面

```
//定义
//userArray 是用户的登录名的数组，不包括自己
//VC 是将要push聊天view的viewController
-(void)chat:(NSArray *)userArray withViewController:(UIViewController *)VC withFailCallbac
k:(finishCallback)callback;
```

```
//示例
NSArray *userArr = [[NSArray alloc] initWithObjects:@"zhangsan",@"lisi",@"wangwu", nil];
[[MXChat sharedInstance] chat:userArr withViewController:vc withFailCallback:^(id object,
MXError *error) {
        NSLog(@"eror == %@", error.description);
    }];
```

注册消息未读数变化通知的接口

```
//定义
-(void)registUnreadMsgCountCallback:(finishCallback)callback;
//示例
//设置tabbaritem上面的消息未读数
[[MXChat sharedInstance] registUnreadMsgCountCallback:^(id result, MXError *error){
   NSLog(@"unread Message count = %@", result);
        if([result isEqualToString:@"0"])
        {
            [tabBarController.tabBar.items[0] setBadgeValue:nil];
        }
        else
        {
            [tabBarController.tabBar.items[0] setBadgeValue:[NSString stringWithFormat:@"%
@", result]];
        }
}];
```

通讯录集成指引

1.引入头文件

```
#import <MXKit/MXContacts.h>
```

2.获取通讯录的viewController

```
//定义
-(UIViewController *)getContactsViewController;
//示例
UIViewController *chat = [[MXContacts sharedInstance] getContactsViewController];
//你可以push这个viewController
```

3.通讯录模块的详细api使用

获取人员信息，并显示敏行内置的人员信息详情页

```
//定义
//loginName 是要获取的人员的登录名
//VC 是将要push聊天view的viewController
-(void)personInfo:(NSString *)loginName withIndicator:(BOOL)showIndicator withViewControll
er:(UIViewController *)VC;
//示例
[[MXContacts sharedInstance] personInfo:@"zhangsan" withIndicator:YES withViewController:v
c];
```

获取人员信息，结果放在callback中返回

```
//定义
//loginName 是要获取的人员的登录名
-(void)getPersonInfo:(NSString *)loginName withIndicator:(BOOL)showIndicator withCallback:
(finishCallback)callback;
//示例
[[MXContacts sharedInstance] getPersonInfo:@"zhangsan" withIndicator:YES withCallback:^(id
 result, MXError *error){
    //这里的result是一个NSDictionary;
   NSLog(@"zhangsan的个人信息 = %@", result);
}];
```

应用中心集成指引

1.引入头文件

```
#import <MXKit/MXAppCenter.h>
```

2.获取应用中心的viewController

```
//定义
-(UIViewController *)getChatViewController;
//示例
UIViewController *chat = [[MXAppCenter sharedInstance] getAppCenterViewController];
//你可以push这个viewController
```

3.应用中心模块的详细api使用

注册应用中心编辑开始的callback

```
//定义
//编辑模式启动的时候，用户可以对应用进行排序
-(void)registEditBeginCallback:(finishCallback)callback;
//示例
[[MXAppCenter sharedInstance] registEditBeginCallback:^(id result, MXError *error) {
    NSLog(@"应用中心编辑模式启动");
}];
```

注册应用中心编辑结束的callback

```
//定义
-(void)registEditEndCallback:(finishCallback)callback;
//示例
[[MXAppCenter sharedInstance] registEditEndCallback:^(id result, MXError *error) {
    NSLog(@"应用中心编辑模式结束");
}];
```

设置应用中心各个应用的消息未读数

```
//定义
//countDic 里面存的是appID和未读数的key value
-(void)showBadgeCount:(NSDictionary *)countDic;
```

```
//示例
NSDictionary *dicCount = [[NSDictionary alloc] initWithObjectsAndKeys:[NSNumber numberWith
Int:20],@"bde209e19cfebcfa2f719a52a4406441",[NSNumber numberWithInt:15],@"f3710a8b00b49240
57446b53609b5038", nil];
[[MXAppCenter sharedInstance] showBadgeCount:dicCount];
```

设置应用中心某个应用隐藏

```
//定义
//appsArray里面存得是appID的数组
-(void)hiddenApps:(NSArray *)appsArray;
//示例
[[MXAppCenter sharedInstance] hiddenApps:[NSArray arrayWithObjects:@"f3710a8b00b4924057446
b53609b5038", nil]];
```

工作圈集成指引

1.引入头文件

```
#import <MXKit/MXCircle.h>
```

2.获取工作圈的viewController

```
//定义
-(UIViewController *)getCircleViewController;
//示例
UIViewController *chat = [[MXCircle sharedInstance] getCircleViewController];
//你可以push这个viewController
```

3.工作圈模块的详细api使用

如果工作圈首页是在在UITabbarItem上需要设置tabbar的高度

```
//定义
-(void)setTabbarHeight:(int)tabbarHeight;
//示例
[[MXCircle sharedInstance] setTabbarHeight:tabbarHeight];
```

显示工作圈列表

```
//定义
//这个api会把工作圈列表显示在你想要显示的地方，从button开始显示，显示在哪个view里
-(void)showGroupTableViewFromButton:(UIButton *)button InView:(UIView *)view;
//示例
//groupChange 当显示群组的button按下的时候触发的事件
-(void) groupChange:(id)sender {
    UIButton *btn = (UIButton*)sender;
    if (!btn.selected) {
        [[MXCircle sharedInstance] showGroupTableViewFromButton:btn InView:self];
    }
}
```

注册工作圈切换的callback

```
//定义
```

```
-(void)registGroupSelect:(finishCallback)callback;
//示例
[[MXCircle sharedInstance] registGroupSelect:^(id result, MXError *error){
   //result里面是title的名字
   //设置切换后title的名字
}];
```

新建工作圈消息

```
//定义
//messageType是工作圈的类型
/*
typedef enum {
    MESSAGE_TYPE_TEXT,//文本
    MESSAGE_TYPE_TASK,//待办
    MESSAGE_TYPE_ACTIVITY,//活动
    MESSAGE_TYPE_POLL,//投票
    MESSAGE_TYPE_ANNOUNCE,//公告
} MESSAGE_TYPE;
*/
//remoteUrl现在传入nil，将来做扩展用
//title是新建的工作圈消息的title
-(void) createNewMessageWithType:(int) messageType remote:(NSString*)remoteUrl title:(NSSt
ring*)titles;
//示例
//这个方法是说单击各个新建菜单触发的行为
-(void)selectedMenuItemAppVo:(MXAppsVO *)appVo
{
    MESSAGE_TYPE type;
    MXCircle *circle = [MXCircle sharedInstance];
    if([appVo.name isEqualToString: @"update"])
    {
        type = MESSAGE_TYPE_TEXT;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];

        return;
    }
    if([appVo.name isEqualToString:@"mini_task"])
    {
        type = MESSAGE_TYPE_TASK;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
    if([appVo.name isEqualToString:@"event"])
    {
        type = MESSAGE_TYPE_ACTIVITY;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
```

```
    if([appVo.name isEqualToString:@"poll"])
    {
        type = MESSAGE_TYPE_POLL;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
}
```

注册工作圈有新消息的callback通知

```
//定义
-(void)registGroupRedIcon:(finishCallback)callback;
//示例
[[MXCircle sharedInstance] registGroupRedIcon:^(id result, MXError *error) {
    //显示工作圈新消息提示，比如出现新消息的小红点儿
   NSLog(@"show circel new message alert === %d", [result boolValue]);
}];
```

发布消息到工作圈

```
//定义
//type 类型如下
/*
typedef enum {
    MESSAGE_TYPE_TEXT,//文本
    MESSAGE_TYPE_TASK,//待办
    MESSAGE_TYPE_ACTIVITY,//活动
    MESSAGE_TYPE_POLL,//投票
    MESSAGE_TYPE_ANNOUNCE,//公告
} MESSAGE_TYPE;
*/
//content是消息的内容
//VC是要push这个新建工作圈消息的view的viewController
-(void)sendToCircle:(int)type withContent:(NSString *)content withViewController:(UIViewCo
ntroller *)VC;
//示例
[[MXCircle sharedInstance] sendToCircle: MESSAGE_TYPE_TEXT withContent:@"测试新建文本消息" w
ithViewController:vc];
```

API详细说明文档

MXKit接口说明文档

1.引入头文件

```
#import <MXKit/MXKit.h>
```

2.初始化

2.1获取MXKit对象

```
定义：
+(MXKit*)shareMXKit;
示例代码：
MXKit *MXObj=[MXKit shareMXKit];
```

2.2获取敏行viewcontroller，调用此api是整体启动敏行

```
定义：
-(UIViewController *)getMainVC;
示例代码：
UIViewController *mainVC=[MXObj getMainVC];
```

2.3将app的windows对象传递给敏行，在Kit里对windows作相应操作

```
定义：
-(void)setWindowToMXKit:(UIWindow*)window;
示例代码：
[MXObj setWindowToMXKit:self.window];
```

2.4处理数据的加密升级

```
定义：
-(void)handleUpgrade:(finishCallback)callback;
示例代码：
[MXObj handleUpgrade:^(id result, MXError *error) {
  //一些初始化配置的操作...
}];
```

2.5将主app的系统delegate传递给kit；将敏行作为一个整体调用的时候才需要调用这些接口

```
定义：
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions;//应用程序启动之后，要执行的委托调用

-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation;

-(void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken;

-(void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error;

-(void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo;

-(void)applicationWillResignActive:(UIApplication *)application;//应用程序将要由活动状态切换到非活动状态时执行的委托调用，如按下home按钮，返回主屏幕，或全屏之间切换应用程序等

-(void)applicationDidEnterBackground:(UIApplication *)application;//在应用程序已进入后台程序时，要执行的委托调用，所以要设置后台继续运行，则在这个函数里面设置即可。
-(void)applicationWillEnterForeground:(UIApplication *)application;//在应用程序将要进入前台时（被激活），要执行的委托调用，与applicationWillResignActive方法相对应
-(void)applicationDidBecomeActive:(UIApplication *)application;////在应用程序已被激活后，要执行的委托调用，刚好与  applicationDidEnterBackground 方法相对应。
-(void)applicationWillTerminate:(UIApplication *)application;////在应用程序要完全退出的时候，要执行的委托调用。
-(void)application:(UIApplication *)application performFetchWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler;
//后台执行部分代码的方法
示例代码：
//在appdelegate.m里，各个api的使用方法相同，这里以进入后台的事件为例
- (void)applicationDidEnterBackground:(UIApplication *)application {
MXKit *MXObj = [MXKit shareMXKit];
[MXObj applicationDidEnterBackground:application];
}
```

3.设置deviceToken

注册设备到server

```
定义：
-(void)registDeviceTokenToserver:(NSString *)deviceToken;
```

示例代码：

```
[MXObj registDeviceTokenToserver:devicetoken];
```

4.服务器环境

4.1设置服务器环境

参数说明：
MX_URL：服务器地址
MX_PORT：服务器端口号
MX_MQTT_URL：服务器推送地址
MX_MQTT_PORT：服务器推送端口号
定义：

```
-(void)init:(NSString *)url withPort:(NSString *)minxingPort withMqttUrl:(NSString *)mqttU
rl withMqttPort:(NSString *)mqttPort;
```

示例代码：

```
[MXObj init:MX_URL withPort:MX_PORT withMqttUrl:MX_MQTT_URL withMqttPort:MX_MQTT_PORT];
```

4.2获取服务器参数

定义：

```
-(NSDictionary *)getMinxing;
```

示例代码：

```
NSDictionary *dic = [NSDictionary dictionaryWithDictionary:[MXObj getMinxing]];
```

4.3通过IP修改服务器参数进行保存

参数说明：
url：服务器地址
minxingPort:服务器端口号
mqttUrl:服务器推送地址
mqttPort:服务器推送端口号
定义：

```
-(void)save:(NSString *)url withPort:(NSString *)minxingPort withMqttUrl:(NSString *)mqttU
rl withMqttPort:(NSString *)mqttPort;
```

示例代码：

```
[MXObj save:serverField.text withPort:serverPortField.text withMqttUrl:mqttField.text with
MqttPort:mqttPortField.text];
```

5.登录与退出的相关接口

5.1第一次登录(需要用户名和密码)

参数说明：
loginName：登录名
password：密码
定义：
```
-(void)login:(NSString *)loginName withPassword:(NSString *)password;
```
示例代码：
```
[MXObj login:name withPassword:password];
```

5.2已经登录过的用户直接登录

定义：
```
-(void)login:(finishCallback)callback;
```
示例代码：
```
[MXObj login:^(id result, MXError *error){
if (result &&!error) {
//登录成功之后进行其他的操作
}
}];
```

5.3注册登录成功后回调

定义：
```
-(void)registLoginCallback:(finishCallback)callback;
```
示例代码：
```
[MXObj registLoginCallback:^(id result, MXError *error){
if(result && !error){
[self initapp];
}
}];
```

5.4用户被踢时的回调

定义：
```
-(void)registLogout:(finishCallback)callback;
```
示例代码：
```
[MXObj registLogout:^(id result, MXError *error){
if (result && !error) {
` ` ` ` ` `
}
}];
```

5.5获取短信验证码

参数说明：

```
phoneNumber: 手机号
定义:
-(void)getPhoneTextVerifyCode:(NSString*)phoneNumber withCallback:(finishCallback)callback
;
示例代码:
[[MXKit shareMXKit] getPhoneTextVerifyCode:phoneNumber withCallback:^(id result, MXError *
error){
if(error) {
[(MXLoginView *)self.view resetGetVerifyCodeButton];
}
}];
```

5.6修改密码

```
参数说明:
oldPassword:旧密码
newPassword:新密码
callback: 回调里执行修改成功后的相关操作
定义:
-(void)changPassword:(NSString *)oldPassword withNewPassword:(NSString *)newPassword WithC
allback:(finishCallback)callback;
示例代码:
[MXObj changPassword:currentstr withNewPassword:newPassword WithCallback:^(id result,MXErr
or *error){
if (result) {
//修改成功的操作
}
}];
```

5.7退出登录

```
定义:
-(void)logout;
示例代码:
[MXObj logout];
```

5.8主动注销，例如：切换其他账号登录时可调用

```
定义:
-(void)logoutWithoutUI:(BOOL)removeName;
示例代码:
[MXObj logoutWithoutUI:removeName];
```

6.个人信息相关接口

6.1获取侧拉栏数据(例如：姓名、头像url、已加入的社区等信息)

```
定义：
-(void)getUserDataWithCallback:(finishCallback)callback;
示例代码：
[MXObj getUserDataWithCallback:^(id result, MXError *error){
//result是一个字典类型
userName = [result objectForKey:@"name"];
avatar_url = [result objectForKey:@"imagePath"];
networkId = [[result objectForKey:@"joinedNetworks"] intValue];
joinedNetworksArray = [result objectForKey:@"networks"];
}];
```

6.2展现个人信息页面

```
定义：
-(void)showUserViewWithNav:(UINavigationController *)nav;
示例代码：
int index = [self.tabBarController selectedIndex];
UINavigationController *nav = [self.tabBarController.viewControllers objectAtIndex:index];
MXKit *MXObj = [MXKit shareMXKit];
[MXObj showUserViewWithNav:(UINavigationController *)nav];
```

6.3个人信息页面返回的回调

```
定义：
-(void)registUserInfoBackCallback:(finishCallback)callback;
示例代码：
[MXObj registUserInfoBackCallback:^(id result, MXError *error){
if (result && !error) {
NSLog(@"user info back!!!!!!!!!!!!!");
}
}];
```

6.4获取当前用户信息，返回值是字典类型，可从字典中取得用户名等信息

```
定义：
-(NSDictionary *)getCurrentUser;
示例代码：
NSDictionary *me = [[MXKit shareMXKit] getCurrentUser];
```

6.5获得账号ID

```
定义：
-(int)getAccountID;
示例代码：
int AccountID = [[MXKit shareMXKit] getAccountID];
```

6.6获取社区名

```
定义：
-(void)getNetWorkName;
示例代码：
[[MXKit shareMXKit] getNetWorkName];
```

6.7是否需要显示外部社区工作圈消息的红点儿

```
定义：
-(BOOL)needShowRedIcon;
示例代码：
BOOL showRedIcon = [[MXKit shareMXKit] needShowRedIcon];
```

6.8获取已加入社区未读数列表，通过字典获取未读数

```
定义：
-(NSDictionary *)getUnreadDict;
示例代码：
NSDictionary *unreadDict=[MXObj getUnreadDict];
```

6.9获取社区消息的红点

```
定义：
-(BOOL)getRedIcon:(NSString *)key;
示例代码：
NSString *key = [NSString stringWithFormat:@"%@_%d_%d", @"notification_red_icon", networkV
O.with_user_id, networkVO.ID];
MXKit *MXObj = [MXKit shareMXKit];
BOOL showRedIcon = [MXObj getRedIcon:key];
```

6.10更新未读数的回调

```
定义：
-(void)updateUnreadCount:(finishCallback)callback;
示例代码：
[MXObj updateUnreadCount:^(id result, MXError *error){
if (result && !error) {
```

```
[[NSNotificationCenter defaultCenter] postNotificationName:kNotificationNameChangeNetworkU
nreadCount object:result];
}
}];
```

6.11获取所有社区里的消息未读数

```
定义:
-(int)getCountAllUnreadMessageInAllNetwork;
示例代码:
int unreadCount = [MXObj getCountAllUnreadMessageInAllNetwork];
```

6.12切换社区

```
定义:
-(void)changeNetwork:(int)networkId;
示例代码:
[MXObj changeNetwork:networkVO.ID];
```

6.13用户保留上次登录的社区

```
参数说明:
networkId: 传入社区id
定义:
-(void)forceChangeNetwork2Last:(int)networkId;
示例代码:
[[MXKit shareMXKit] forceChangeNetwork2Last:54];
[self setNetWorkName:@"南区销售"];
```

6.14获得社区名称时的回调

```
定义:
-(void)getNetWorkNameWithCallback:(finishCallback)callback;
示例代码:
[MXObj getNetWorkNameWithCallback:^(id result, MXError *error) {
[self setNetWorkName:result];
}];
```

6.15展示外部社区

```
定义:
-(void)gotoExternalViewWithNav:(UINavigationController *)nav;
示例代码:
```

```
int index = [self.tabBarController selectedIndex];
UINavigationController *nav = [self.tabBarController.viewControllers objectAtIndex:index];
MXKit *MXObj = [MXKit shareMXKit];
[MXObj gotoExternalViewWithNav:(UINavigationController *)nav];
```

6.16展示用户头像

参数说明：
imageView：传入UIImageView控件
image：头像图片
定义：

```
-(void)showUserViewWithImageView:(UIImageView *)imageView andUserImage:(UIImage *)image;
```

示例代码：

```
[MXObj showUserViewWithImageView:herd.imageView andUserImage:herd.imageView.image];
```

6.17使用第三方开发的通讯录

定义：

```
-(void)registerContactsManagerClass:(Class)className;
```

示例代码：

```
[MXObj registerContactsManagerClass:[MXContactsUIManager class]];
```

7.可配配置(在MXClient/library/common/MXConfig.h配置文件中添加配置)

```
//初始化敏邮
[MXObj initMinyou:CLIENT_SHOW_MAIL];
//初始化umeng
[MXObj initUmeng:UMENG_KEY withChannel:UMENG_CHANNEL];
//设置个人信息的header页签(“个人信息”、“工作圈”、“文件”)
[MXObj initUserTabBar:CLIENT_SHOW_USER_TAB_BAR];
//加拨短号
[MXObj initExtendDailNumber:EXTEND_DAIL_NUM];
//是否隐藏工作圈的创建
[MXObj initHideCircleCreation:YES];
//隐藏公司通讯录
[MXObj initDisableCompanyContact:CLIENT_SHOW_CONTACT_COMPANY];
//设置控制程序内部浏览器右上角弹出菜单的背景颜色、分割线的颜色、选中颜色、字体颜色
[MXObj initBaseColor:navBarColor];
[MXObj initSepeartorColor:ColorWithRGB(88, 166, 255, 1)];
[MXObj initSelectedColor:[UIColor redColor]];
[MXObj initPopUpTextColor:[UIColor yellowColor]];
[MXObj initSureBtnTitleColor:[UIColor greenColor]];
//个人信息页配置toolView BtnColor(“加为联系人”、“发消息”、“特别关注”这些字体的颜色)和个人信息页面配置上部选中状态的底栏的颜色
```

```
[MXObj initInfoBtnColor:infoBtnAndBarColor];
[MXObj initInfoBarColor:infoBtnAndBarColor];
//设置电话号码是否需要加密(星号),比如13812345678，隐藏第4位到第7位，那么变为138xxxx5678。
[MXObj initEncryptCellphonePatten:@"4-7"];
//设置通讯录索引颜色
[MXObj initContactIndexColor:[UIColor greenColor]];
//设置自定义clientID
[MXObj initCustomClientID:MX_CUSTOM_CLIENT_ID];
//启用水印
[MXObj initWarterMask:MX_ENABLE_WATERMASK];
//隐藏公众号
[MXObj initHidePublicAccount:CLIENT_HIDDEN_PUBLIC_ACCOUNT];
//隐藏群聊
[MXObj initHideMultiChat:CLIENT_HIDDEN_MULTI_CHAT];
//隐藏特别关注
[MXObj initHideVipcontant:YES];
//启用邮件
[MXObj initEnableAddressEmail:MX_ENABLE_ADDRESS_MAIL];
//禁用应用中心划线
[MXObj initDisableDrawLine:YES];
//禁止下载
[MXObj initDisableDownload:MX_DISABLE_DOWNLOAD];
//短信验证登录
[MXObj initMobileLogin:MX_ENABLE_MOBILE_TEXT_VERIFY];
//禁用邮箱会话
[MXObj initDisableMXMail:MX_DISABLE_MXMAIL];
//禁用shareExtension
[MXObj initDisableShareExtension:MX_DISABLE_SHARE_EXTENSION];
//点击他人的邮箱地址禁止用敏邮启动
[MXObj initDisableMXMail:YES];
//设置状态栏颜色
[[MXKit shareMXKit]setStatusBarStyle:UIStatusBarStyleLightContent];
//设置title的字体和颜色,传入参数attr是一个字典
[MXObj setTitleBarAttribute:attr];
//设置item bar的字体和颜色，传入参数attr是一个字典
[MXObj setItemBarAttribute:attr];
//设置进度条加载的背景颜色
[[MXKit shareMXKit] setWebViewProgressColor:[UIColor mx_colorWithHexString:@"#1895ff"]];
//设置应用中心的划线
[MXObj initEnableDrawLine:MX_DISABLE_APPCENTERDRAWLINE];
//个人信息页面分组展示
[MXObj initEnableUserInfoGroup:YES];
//设置主题颜色，应用程序的主色调
[[MXKit shareMXKit] setThemeColor:themeColor];
//设置添加好友
[[MXKit shareMXKit] initEnableAddFriends:YES];
//启用公众号历史消息的分享
[[MXKit shareMXKit] endableHistoryMessageShare:YES];
```

8.其他API

注册第三方接收的推送的回调

```
定义：
-(void)registPushServiceCallback:(finishCallback)callback;
示例代码：
[MXObj registPushServiceCallback:^(id result, MXError *error) {
//这里的推送收到的时候是一个非主线程的推送,如果用户需要更新UI，请自行切换到主线程
NSLog(@"receiver push from server==%@", result);
}];
```

点击tabbar时kit内部处理tabbar红点更新等的一些操作

```
定义：
-(void)tabBarControllerDidSelect:(UINavigationController *)navigationController;
示例代码：
- (void)tabBarController:(UITabBarController *)tabBarController didSelectViewController:(U
IViewController *)viewController {
UINavigationController *navCtrl = [self.tabBarController.viewControllers objectAtIndex:self
.tabBarController.selectedIndex];
NSArray* vcArray = [navCtrl viewControllers];
if (vcArray.count == 1) {
MXKit *MXObj = [MXKit shareMXKit];
[MXObj tabBarControllerDidSelect:navCtrl];
}
}
```

切换页签时的回调(app2app)

```
定义：
-(void)registTabSelectCallback:(finishCallback)callback;
示例代码：
[MXObj registTabSelectCallback:^(id result, MXError *error){
if(!error) {
NSNumber *index = (NSNumber *)result;
[self.tabBarController setSelectedIndex:index.intValue];
}
}];
```

注册处理公众号里面第三方数据的回调

```
定义：
-(void)registNativeCallback:(handleNativeCallback)callback;
```

```
示例代码:
[[MXKit shareMXKit] registNativeCallback: ^(id result, MXError *error){
NSLog(@"handle native result===%@", result);
return NO;
}];
```

注册新版本的回调

```
定义:
-(void)registNewVersionCallback:(finishCallback)callback;
示例代码:
[[MXKit shareMXKit] registNewVersionCallback: ^(id result, MXError *error){
isNew = YES;
}];
```

注册页面显示的回调, 这里finishCallback是一个nsdictionary， 字段名是@"className"和@"title"

```
定义:
-(void)registPageAppearCallback:(finishCallback)callback;
示例代码:
[MXObj registPageAppearCallback:^(id object, MXError *error) {
NSLog(@"view will appear callback === %@", object);
}];
```

注册页面消失的回调

```
定义:
-(void)registPageDisapperCallback:(finishCallback)callback;
示例代码:
[MXObj registPageDisapperCallback:^(id object, MXError *error) {
NSLog(@"view will disappear callback === %@", object);
}];
```

注册导航栏底部隐藏的回调

```
定义:
-(void)registNavigationBottomBarHiddenCallback:(finishCallback)hiddenCallback;
示例代码:
[MXObj registNavigationBottomBarHiddenCallback:^(id object, MXError *error) {
` ` ` ` ` `
}];
```

关闭其他视图

```
定义：
-(void)closeFunctionView;
示例代码：
[MXObj closeFunctionView];
```

注册第二辑页面显示的回调

```
定义：
-(void)registSecondLevelPageAppearCallback:(finishCallback)callback;
示例代码：
[MXObj registSecondLevelPageAppearCallback:^(id object, MXError *error) {
NSLog(@"view will secondleverlPageappear callback === %@", object);
}];
```

在屏幕顶端显示error信息

```
定义：
-(void)showMXErrorTips:(NSString *)errorString;
示例代码：
[MXObj showMXErrorTips:errorstr];
```

提示添加名片成功

```
定义：
-(void)showShortMessage:(NSString *)text;
示例代码：
[[MXKit shareMXKit] showShortMessage:@"添加名片成功"];
```

扫一扫接口的使用

```
定义：
-(UIViewController *)getQRScanViewController;
-(void)registQRResultCallback:(handleNativeCallback)callback;
示例代码：
//获取扫一扫的页面
UIViewController *qrVC = [[MXKit shareMXKit] getQRScanViewController];
qrVC.hidesBottomBarWhenPushed = YES;
UIViewController *rootVC = [[MXChat sharedInstance] getChatViewController];
[rootVC.navigationController pushViewController:qrVC animated:YES];
//注册二维码扫描的结果回调
[[MXKit shareMXKit] registQRResultCallback:^(id result, MXError *error) {
NSLog(@"result===%@", result);
```

```
//这里的return表示是否需要mxkit继续后面的处理，如果return NO则表示需要mxkit继续处理，反之return YES
return NO;
}];
```

获得webview的url，通过给参数url拼接mxLayout来控制标题栏、工具栏、分享菜单

```
定义：
-(UIViewController *)getWebViewController:(NSString *)url;
示例代码：
UIViewController *vc = [[MXKit shareMXKit] getWebViewController:url];
```

urlpage作为页签

```
定义：
-(UIViewController *)getWebUrlViewController:(NSString *)url;
示例代码：
UIViewController *webUrl=[MXObj getWebUrlViewController:arr[i][@"value"]];
```

设置urlpage的headerview

```
参数说明：
headerView：传入自定义导航栏的UIView视图
定义：
-(void)setCustomHeaderView:(UIView *)headerView;
示例代码：
[MXObj setCustomHeaderView:urlHeader];
```

点击urlPage页签内部显示新的urlpage

```
参数说明：
url：页面网址
showStyle：页面的展示方式（push、present方式）
viewController：当前控制器
定义：
-(void)getUrlPageWithUrl:(NSString *)url withShowStyler:(NavigationBarShowStyle)showStyle withViewController:(UIViewController *)viewController withCallback:(finishCallback)callBack;
示例代码：
[MXObj getUrlPageWithUrl:url withShowStyler:NavigationBarShowStyle withViewController:vc withCallback:^(id object, MXError *error) {
//页面将要出现时的操作
}];
```

新闻页签

```
定义:
-(UIViewController *)getNewsViewController;//获取新闻controller
-(void)cleanNewsViewcontroller; //清除新闻
- (void)setNewsViewControllerCustomHeader:(UIView *)headerView; //设置新闻页签自定义header
示例代码:
UIViewController *news = [MXObj getNewsViewController];
[MXObj setNewsViewControllerCustomHeader:urlHeader];
```

展示第三方app调用敏行聊天-

```
定义:
-(void)showMineApp2AppChat;
示例代码:
[MXObj showMineApp2AppChat];
```

第三方app强制登录敏行的回调

```
定义:
-(void)registForceLoginCallback:(finishCallback)callback;
示例代码:
[MXObj registForceLoginCallback:^(id result, MXError *error) {
if(result && !error)
{
}
}];
```

展示我的收藏、我的消息、我上传的文件、我下载的文件的页面

```
定义:
-(void)showFavorteWithNav:(UINavigationController *)nav;
-(void)showMyMessage:(UINavigationController *)vc;
-(void)showMyUpload:(UINavigationController *)vc;
-(void)showMyDownload:(UINavigationController *)vc;
示例代码:
switch (index) {
case 0:
[[MXKit shareMXKit] showFavorteWithNav:[self.tabBarController.viewControllers objectAtInde
x:tabIndex]];
break;
case 1:
[[MXKit shareMXKit] showMyMessage:[self.tabBarController.viewControllers objectAtIndex:tab
Index]];
```

```
break;
case 2:
[[MXKit shareMXKit] showMyUpload:[self.tabBarController.viewControllers objectAtIndex:tabI
ndex]];
break;
case 3:
[[MXKit shareMXKit] showMyDownload:[self.tabBarController.viewControllers objectAtIndex:ta
bIndex]];
break;
default:
break;
}
```

plist的值的获取、存储、删除以及二进制的获取、存储

定义：

```
//获取MX plist的值
-(id)getMXUserDefaultValueForKey:(NSString *)key;
//存储MX plist的值
-(void)setMXUserDefaultValue:(id)value forKey:(NSString *)key;
//删除MX plist的值
-(void)removeMXUserDefaultValueForKey:(NSString *)key;
//获取二进制的值
-(id)getMXUserDefaultBinaryValueForKey:(NSString *)key;
//存储二进制的值
-(void)setMXUserDefaultBinaryValue:(id)value forKey:(NSString *)key;
```

示例代码：

```
NSString *userName = [[MXKit shareMXKit] getMXUserDefaultValueForKey:@"user_name"];
[[MXKit shareMXKit] setMXUserDefaultValue:name forKey:@"user_name"];
[[MXKit shareMXKit] removeMXUserDefaultValueForKey:userName];
NSData *data = [[MXKit shareMXKit] getMXUserDefaultBinaryValueForKey:@"gesturePassCode"];
[[MXKit shareMXKit] setMXUserDefaultBinaryValue:passCodeData forKey:@"gesturePassCode"];
```

敏邮的登录

定义：

```
//带用户名密码登陆
-(void)showMXEmailWithUserName:(NSString *)userName withPassword:(NSString *)password with
AppID:(NSString *)appID withViewController:(UIViewController*)parentVC;
//不带用户名密码
-(void)showMXEmailWithViewController:(UIViewController*)parentVC withAppID:(NSString *)app
ID withParams:(NSString *)params;
```

示例代码：

```
[[MXKit shareMXKit] showMXEmailWithUserName: @"test@dehuinet.com" withPassword:@"Test123"
withAppID:self.appID withViewController:self.parentViewController];
[[MXKit shareMXKit] showMXEmailWithViewController:self.parentViewController withAppID:self
.appID withParams:self.params];
```

设置字体大小完成时的回调

参数说明:
num: 字体大小(0-3依次为小、标准、大、特大)
aBlock: 回调里执行设置字体大小完成时的操作
定义:

```
-(void)setFontNum:(CGFloat)num andFinishedBlock:(finishCallback)aBlock;
```

示例代码:

```
[[MXKit shareMXKit] setFontNum:_flag andFinishedBlock:^(id object, MXError *error) {
[weakHUD hide:YES];
[weakSelf.navigationController popViewControllerAnimated:YES];
}];
```

获取字体大小

定义:

```
-(CGFloat)getFontNum;
```

示例代码:

```
CGFloat _flag = [[MXKit shareMXKit] getFontNum];
```

设置分享的Keys

定义:

```
-(void)setShareExtensionGroupID:(NSString *)groupID andWorkCircleKey:(NSString *)key1 andC
hatKey:(NSString *)key2 andheadersKey:(NSString *)key3 andTokenKey:(NSString *)key4 chatRe
tArrayKey:(NSString *)key5;
```

示例代码:

```
[[MXKit shareMXKit] setShareExtensionGroupID:MX_SHARE_GORUPID andWorkCircleKey:MX_SHARE_CI
RCLE andChatKey:MX_SHARE_CHAT andheadersKey:MX_SHARE_REQUEST_HEADER andTokenKey:MX_SHART_T
OKEN chatRetArrayKey:MX_SHARE_CAHTARRAY];
```

获取邮箱未读消息

定义:

```
-(int)getUnreadEmailCount;
```

示例代码:

```
[[MXKit shareMXKit] getUnreadEmailCount];
```

敏邮升级

定义:

```
-(void)mxMailUpgrade;
```

示例代码:

```
[[MXKit shareMXKit] mxMailUpgrade];
```

设置邮件前台收取间隔，单位是秒

```
参数说明：
interval：收取间隔(秒数)
定义：
-(void)initEmailInterval:(int)interval;
示例代码：
[MXObj initEmailInterval:MX_GET_EMAIL_INTERVAL];
```

发送邮件给指定的人

```
参数说明：
name：名字(可为空)
email：邮箱地址
parentViewController：传入当前控制器，用于跳转到写邮箱界面
定义：
-(void)sendEmailTo:(NSString *)name withEmail:(NSString *)email withViewControler:(UIViewC
ontroller *)parentViewController;
示例代码：
//参数name可以为nil，其他参数不可以为nil
[[MXKit shareMXKit] sendEmailTo:nil withEmail:@"liyang@dehuinet.com" withViewControler:self
];
```

获取设备UUID

```
定义：
-(NSString *)getDeviceUUID;
示例代码：
NSString *str =[[MXKit shareMXKit] getDeviceUUID];
```

开启敏邮日志

```
定义：
-(void)enableMailLog;
示例代码：
[[MXKit shareMXKit]enableMailLog];
```

删除敏邮信息

```
定义：
-(void)removeMXMail;
```

```
示例代码：
[[MXKit shareMXKit] removeMXMail];
```

测试会话

```
定义：
-(void)testConversation;
示例代码：
[[MXKit shareMXKit]testConversation];
```

同步未读数到server

```
参数说明；
unreadCount：未读数
定义：
-(void)syncUnreadCount2Server:(NSInteger )unreadCount;
示例代码：
[[MXKit shareMXKit]syncUnreadCount2Server:unredcount];
```

显示健步走排行榜列表

```
参数说明：
parentVC:父控制器，用来push到排行榜页面
AppId:应用id
ocuId:公众号Id
定义：
-(void)showWalkingRankingListWithViewController:(UIViewController *)parentVC AppId:(NSStri
ng *)appId OcuId:(int)ocuId;
示例代码：
[[MXKit shareMXKit]showWalkingRankingListWithViewController:self.parentViewController AppI
d:@"running_man" OCuId:1];
```

显示健步走配置页面

```
参数说明：
parentVC:父控制器，用来push到排行榜页面
AppId:应用id
ocuId:公众号Id
定义：
-(void)showWalkingInfoWithViewController:(UIViewController *)parentVC withApppID:(NSString
 *)appId OcuId:(int)ocuId;
示例代码：
[[MXKit shareMXKit]showWalkingInfoWithViewController:self.parentViewController AppId:@"run
ning_man" OCuId:1];
```

获取当前环境的服务器地址，例如：test.dehuinet.com

定义：

```
-(NSString *)getHostname;
```

示例代码：

```
[MXObj getHostname];
```

设置非wifi环境下 上传下载文件大小限制 超过限制弹出提示

定义：

```
- (void)initNoWifiFileSizeLimit:(NSInteger)size;
```

示例代码：

```
[MXObj initNoWifiFileSizeLimit:MX_NOWIFI_FILESIZE_LIMIT];
```

公众号入口，传入参数OCuID

定义：

```
- (void)startOcuChat:(NSString *)ocuID;
```

示例代码：

```
[[MXKit shareMXKit] startOcuChat:@"ccb74893d3c14028524e9751a3770381"];
```

ios附件下载问题-在删除缓存的时候调用接口把下载文件时存到数据库里的文件相关也删掉

定义：

```
- (void)cleanLocalFiles;
```

示例代码：

```
[[MXKit shareMXKit]cleanLocalFiles];
```

获得用户敏行信息,例如：MinxingMessenger/4.0.0.036(iPhone 5s; iOS/10.1.1)

定义：

```
-(NSString *)getMXUserAgent;
```

示例代码：

```
NSString *userAgent = [[MXKit shareMXKit] getMXUserAgent];
```

启用安全隧道

参数说明

address：IP地址

定义：

```
-(void)startMXTunnel:(NSString *)address;
示例代码:
[[MXKit shareMXKit] startMXTunnel:@"dev3.dehuinet.com:2990"];
```

设置安全隧道白名单端口号

```
定义:
-(void)setMXTunnelWhiteListPort:(NSString *)port;
示例代码:
[[MXKit shareMXKit] setMXTunnelWhiteListPort:@"3021"];
```

注册安全隧道连接

```
定义:
-(void)registMXTunnelConnected:(finishCallback)callback;
示例代码:
[[MXKit shareMXKit] registMXTunnelConnected:^(id object, MXError *error){
 if(!error) {
 //成功后的操作
  [self handleLoginCallback];
 } else {
   //show error
  [[MXKit shareMXKit] showMXErrorTips:error.description];
   }
}];
```

获取安全隧道登录名

```
定义:
-(NSString *)getTunnelUser;
示例代码:
NSString *tunnelUser = [[MXKit shareMXKit] getTunnelUser];
```

设置移动设备管理

```
定义:
- (void)setMDMEnable:(BOOL)isMDM;
示例代码:
[MXObj setMDMEnable:MX_MDM_ENABLE];
```

接收自签证书challenge

```
定义:
```

```
-(void)mxDidReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge;
示例代码:
[[MXKit shareMXKit] mxDidReceiveAuthenticationChallenge:challenge];
```

注册分析日志，用来分析相关问题

```
定义:
-(void)registAnalayseEvent:(functionCallback)callback;
示例代码:
//UM Event Handler
[[MXKit shareMXKit] registAnalayseEvent:^(NSDictionary *eventDic) {
if(eventDic && [eventDic isKindOfClass:[NSDictionary class]]) {
NSString *eventID = [eventDic objectForKey:@"eventID"];
NSDictionary *attriDic = [eventDic objectForKey:@"attributes"];
NSNumber *duration = [eventDic objectForKey:@"durations"];
if([duration integerValue] > 0) {
if(attriDic) {
[MobClick event:eventID attributes:attriDic counter:[duration intValue]];
} else {
[MobClick event:eventID durations:[duration intValue]];
}
} else {
if(attriDic) {
[MobClick event:eventID attributes:attriDic];
} else {
[MobClick event:eventID];
}
}
}
}];
```

注册更新GPS的回调

```
定义:
-(void)registGPSProvider:(functionCallback)callback;
示例代码:
[MXObj registGPSProvider:^(handleGpsUpdateCallback gpsCallback){
    MXLocation *locationpProvider = [[MXLocation alloc] init];
    [locationpProvider getLocationWithCallback:gpsCallback];
}];
```

自动上传GPS

```
参数说明:
url: 接收gps,自动上传的url
interval: 自动上传gps的时间间隔（单位是秒）
```

```
定义：
-(void)registGPSUploadServerAddress:(NSString *)url withInterval:(CGFloat)interval;
示例代码：
[MXObj registGPSUploadServerAddress:MX_GPS_UPLOADING_URL withInterval:MX_GPS_UPLOADING_INT
ERVAL];
```

检查是否启用手势密码或者指纹验证，如果成功或者不需要验证则执行callback

```
定义：
 -(void)checkSecurtyAuthStatusWithCallback:(fuctionNoParamCallback)callback;
示例代码：
[[MXKit shareMXKit] checkSecurtyAuthStatusWithCallback:^{
//有token进入主页面
[weakSelf showMain];                    dispatch_async(dispatch_get_global_queue(DISPATCH_
QUEUE_PRIORITY_DEFAULT, 0), ^{
  [weakSelf startVersionCheck:YES];
});
}];
```

注册工作圈自定义action的点击事件

```
定义：
//同MXCircle的方法(-(void)setAction:(NSArray *)action)配合使用
- (void)registCustomActionCallback:(finishCallback)callback;
示例代码：
[MXObj registCustomActionCallback:^(id object, MXError *error) {
if ([object isKindOfClass:[NSDictionary class]]) {
  if ([[object valueForKey:@"key"] isEqualToString:@"mx_baidu"]) {
     CDVViewController * cdv = [[CDVViewController alloc] init];
     UINavigationController * nav = [[UINavigationController alloc] initWithRootViewContro
ller:cdv];
     cdv.startPage = @"https://www.baidu.com";
    [self.window.rootViewController presentViewController:nav animated:YES completion:nil]
;
  }
}
}];
###  发送用户验证码的回调
参数说明：
phoneNum：要发送验证码的手机号
state:  状态(1:注册，2:忘记密码，3:修改手机号)
定义：
-(void)sendAuthCode:(NSString *)phoneNum state:(int)state  callback:(finishCallback)callba
ck;
示例代码：
[MXObj sendAuthCode:self.phoneNum state:self.state callback:^(id object, MXError *error) {
```

```
//执行成功或错误的操作
}];
### 验证用户验证码
```objc
参数说明：
phoneNum：手机号
authCode：验证码
state：状态(1:注册，2:忘记密码，3:修改手机号)
定义：
-(void)validateAuthCode:(NSString *)phoneNum authCode:(NSString *)authCode state:(int)stat
e callback:(finishCallback)callback;
示例代码：
[MXObj validateAuthCode:self.phoneNum authCode:code state:self.state callback:^(id object,
 MXError *error) {
 if (!error.description) {
//执行验证成功时的操作
} else {
//执行验证失败时的操作
}
 }];
```

验证两次输入的密码是否一致

```
参数说明：
phoneNum：手机号
passcode：密码
confirm：第二次密码
deptID：部门ID
state：状态(1:注册，2:忘记密码)
定义：
-(void)regist:(NSString *)phoneNum passcode:(NSString *)passcode confirm:(NSString *)confi
rm deptID:(NSInteger)deptID state:(int)state params:(NSDictionary *)params callback:(finis
hCallback)callback;
示例代码：
[MXObj regist:self.phoneNum passcode:passCodeTextField.text confirm:confirmTextField.text
deptID:-1 state:self.state params:nil callback:^(id object, MXError *error) {
if (!error.description) {
 [SVProgressHUD show];
[MXObj login:self.phoneNum withPassword:passCodeTextField.text];
 } else {
[self showAlertWithTitle:nil message:error.description type:UIAlertControllerStyleAlert ac
tionTitle:GetLocalResStr(@"mx_system_sure")  actionHandler:nil];
 }
}];
```

判断输入的手机账号是否已经注册
```

参数说明：
phoneNum：输入的手机号
定义：
```
-(void)mobileAlreadyExist:(NSString *)phoneNum callback:(finishCallback)callback;
```
示例代码：
```
[MXObj mobileAlreadyExist:emailTF.text callback:^(id object, MXError *error) {
if (!error.description) {
 NSDictionary *dic = (NSDictionary *)object;
 BOOL exist = [dic objectForKey:@"exists"];
 if (exist) {
//手机号已注册
 } else {
//没注册
 }
}
}];
```

修改手机号

参数说明：
newPhoneNum：新的手机号
定义：
```
- (void)changePhoneNum:(NSString *)newPhoneNum callback:(finishCallback)callback;
```
示例代码：
```
[MXObj changePhoneNum:self.phoneNum callback:^(id object, MXError *error) {
if (!error.description) {
//修改成功
} else {
//修改失败
}
}];
```

校验密码强度

参数说明：
password：输入的密码
定义：
```
- (void)checkPasswordStrength:(NSString *)password callback:(finishCallback)callback;
```
示例代码：
```
[MXObj checkPasswordStrength:newPasswordTF.text callback:^(id object, MXError *error) {
if (!error.description) {
} else {
}
}];
```

拿到view的stack里面最上面的viewController

定义：
```
-(UIViewController *)getTopViewController;
```
示例代码：
```
[[MXKit shareMXKit] getTopViewController];
```

获取注册时需要填写的用户属性

定义：
```
- (void)getPersonInfoData:(finishCallback)callback;
```
示例代码：
```
[[MXKit shareMXKit] getPersonInfoData:^(id object, MXError *error) {
if (!error.description) {
}
}];
```

获取社群名或域名(大服务)

定义：
```
-(NSString*)getCurrentDomainName;
```
示例代码：
```
[[MXKit shareMXKit]getCurrentDomainName];
```

展示社群页面

定义：
```
-(void)showDomainViewWithNav:(UINavigationController *)nav;
```
示例代码：
```
int tabIndex =  [self.tabBarController selectedIndex];
[[MXKit shareMXKit] showDomainViewWithNav:[self.tabBarController.viewControllers objectAtI
ndex:tabIndex]];
```

MXChat接口说明文档

1.引入头文件

```
#import <MXKit/MXChat.h>
```

2.初始化MXChat

2.1获取MXChat对象

```
定义:
+(MXChat*)sharedInstance;
示例代码:
[MXChat sharedInstance];
```

2.2获取聊天的ViewController

```
定义:
-(UIViewController *)getChatViewController;
示例代码:
UIViewController *chat = [[MXChat sharedInstance] getChatViewController];
```

2.3自定义聊天导航栏视图

```
定义:
-(void)setCustomHeaderView:(UIView *)headerView;
示例代码:
MXChatViewHeader *chatHeader = [[MXChatViewHeader alloc] initWithFrame:headerFrame];
//如果聊天view带tabbar,  需要设置tabbarheight
[[MXChat sharedInstance] setTabbarHeight:tabbarHeight];
[[MXChat sharedInstance] setCustomHeaderView:chatHeader];
```

2.4释放聊天的viewcontroller

```
定义:
-(void)cleanChatViewController;
示例代码:
[[MXChat sharedInstance] cleanChatViewController];
```

3.MXChat模块详细API使用说明

右滑chat界面时的回调

```
定义:
-(void)startPullDownWithCallback:(finishCallback)callback;
示例代码:
[[MXChat sharedInstance] startPullDownWithCallback:^(id result, MXError *error) {
  if (result) {
  [chatHeader startPullDown];
    }
}];
```

注册通讯录名片插件

```
定义:
-(void)registbussnissCard;
示例代码:
[[MXChat sharedInstance] registbussnissCard];
```

注册红包插件

```
定义:
-(void)registRedPacket;
示例代码:
[[MXChat sharedInstance] registRedPacket];
```

注册文件共享

```
定义:
- (void)registDocView;
示例代码:
[[MXChat sharedInstance] registDocView];
```

注册健步走插件

```
定义:
-(void)registWalking;
示例代码:
[[MXChat sharedInstance] registWalking];
```

发起聊天，自带通讯录选人界面

```
定义：
-(void)startChat;
示例代码：
[[MXChat sharedInstance] startChat];
```

发起聊天，不带通讯录选人界面

```
参数说明：
userArray：是用户的登录名的数组，不包括自己
VC：是将要push聊天view的viewController
定义：
-(void)chat:(NSArray *)userArray withViewController:(UIViewController *)VC withFailCallbac
k:(finishCallback)callback;
示例代码：
NSArray *userArr = [[NSArray alloc] initWithObjects:@"zhangsan",@"lisi",@"wangwu", nil];
[[MXChat sharedInstance] chat:userArr withViewController:vc withFailCallback:^(id object,
MXError *error) {
  NSLog(@"eror == %@", error.description);
}];
```

分享到聊天接口

```
参数说明：
title：标题
description：详细描述
url：链接地址
nativeURL：appurl，可为空
thumbnailURL：缩略图地址
VC：传入控制器参数，用于present到本地群聊页面，可为空
定义：
-(void)shareTitle:(NSString *)title withDescription:(NSString *)description withURL:(NSStr
ing *)url withNativeURL:(NSString *)nativeURL withThumbnailURL:(NSString *)thumbnailURL wi
thViewController:(UIViewController *)VC;
示例代码：
[[MXChat sharedInstance] shareTitle:@"112233" withDescription:@"ee33" withURL:@"https://ww
w.baidu.com" withNativeURL:nil withThumbnailURL:@"https://ss3.baidu.com/-fo3dSag_xI4khGko9
WTAnF6hhy/super/whfpf%3D425%2C260%2C50/sign=08ba595e4fed2e73fcbcd56ce13c95b9/d000baa1cd117
28b79d03af8cdfcc3cec2fd2c6e.jpg" withViewController:self];
```

收藏message

```
参数说明：
messageID：消息Id(message_id)
title：标题
url：  网址
```

```
appUrl：app地址
定义：
-(void)addFavoriteToServer:(int)messageID withTitle:(NSString *)title withUrl:(NSString *)
url withAppUrl:(NSString *)appUrl;
示例代码：
[[MXChat sharedInstance] addFavoriteToServer:messageId withTitle:nil withUrl:nil withAppUr
l:nil];
```

插件消息分享到聊天的接口

```
参数说明：
dataString：  消息内容
VC：  传入控制器参数，用于present到选人页面
定义：
-(void)sharePluginMessage:(NSString *)dataString withViewController:(UIViewController *)VC
;
示例代码：
[[MXChat sharedInstance] sharePluginMessage:_bodyString withViewController:self];
```

进入详情资料

```
定义：
-(void)goToDetail;
示例代码：
[[MXChat sharedInstance] goToDetail];
```

进入赞我的朋友界面

```
定义：
-(void)goToLike;
示例代码：
[[MXChat sharedInstance] goToLike];
```

添加联系人

```
定义：
-(void)addContact;
示例代码：
[[MXChat sharedInstance] addContact];
```

发布工作圈的消息

```
定义：
```

```
-(void)shareJob;
示例代码：
[[MXChat sharedInstance] shareJob];
```

注册未读消息数变化的回调，用来更新聊天消息数

```
定义：
-(void)registUnreadMsgCountCallback:(finishCallback)callback;
示例代码：
[[MXChat sharedInstance] registUnreadMsgCountCallback:^(id result, MXError *error){
 NSLog(@"unread Message == %@", result);
 int messageIndex = [self getMessageIndex];
 if([result isEqualToString:@"0"])
 {
   [tabBarController.tabBar.items[messageIndex] setBadgeValue:nil];
 }else{
    [tabBarController.tabBar.items[messageIndex] setBadgeValue:[NSString stringWithFormat:
@"%@", result]];
    }
}];
```

主动获取未读消息数

```
定义：
-(int)getUnreadMessageCount;
示例代码：
int uncount=[MXChat sharedInstance] getUnreadMessageCount];
```

注册收到邀请通知后点击跳转到指定群组的回调-

```
定义：
-(void)registSwitchGroupCallback:(finishCallback)callback;
示例代码：
[[MXChat sharedInstance] registSwitchGroupCallback: ^(id result, MXError *error){
  NSLog(@"switch group");
 [tabBarController setSelectedIndex:[self getCircleIndex]];
 }];
```

注册点击聊天cell的回调

```
定义：
-(void)registChatDidSelectRowCallback:(finishCallback)callback;
示例代码：
[[MXChat sharedInstance] registChatDidSelectRowCallback:^(id result, MXError *error) {
```

```
  if (result) {
   //点击聊天的cellback 写隐藏tabbar的方法
  }
}];
```

注册聊天列表上方显示收取中的回调

```
定义:
-(void)registChatReceivingIndicatorCallback:(finishCallback)callback;
示例代码:
[[MXChat sharedInstance] registChatReceivingIndicatorCallback:^(id result, MXError *error)
 {
    BOOL hidden = [result boolValue];
    [chatHeader setNavBarTitleHidden:!hidden];
    [chatHeader setNavBarTitleIndicatorHidden:hidden];
}];
```

注册chatViewWillAppear的回调

```
定义:
-(void)registViewWillAppearCallback:(finishCallback)callback;
示例代码:
[[MXChat sharedInstance] registViewWillAppearCallback:^(id result, MXError *error) {
  NSLog(@"list view will appear");
}];
```

注册聊天会话点back之后的行为，如果不注册这个callback，那么就是默认行为，直接返回到上一个view

```
定义:
-(void)registChatBackAction:(finishCallback)callback;
示例代码:
[[MXChat sharedInstance] registChatBackAction:^(id result, MXError *error) {
 NSLog(@"chat back action callback called");
[self.tabBarController setSelectedIndex:[self getMessageIndex]];
//如果是通讯里里面直接点人发起的聊天，通讯录应该回到最顶层，用户可以自定义这个行为
 [navAddressBook popToRootViewControllerAnimated:YES];
 [navWorkCircle popToRootViewControllerAnimated:YES];
 [navAppCenter popToRootViewControllerAnimated:YES];
 [navChat popToRootViewControllerAnimated:YES];
}];
```

注册用Api调用聊天页面会话页面点back的行为，不注册默认回到上一个view

```
定义：
-(void)registApiChatBackAction:(finishCallback)callback;
示例代码：
[[MXChat sharedInstance] registApiChatBackAction:^(id result, MXError *error) {

}];
```

删除指定ID的message

```
定义：
-(BOOL)deleteMsg:(int)messageID;
示例代码：
[[MXChat sharedInstance]deleteMsg:messageID];
```

自定义会话conversation

```
参数说明：
customKey：自定义会话的key
title：会话标题
content：会话内容
avatarUrl：头像地址
updateTime：server发出消息的时间，"2015-11-19T12:08:50+08:00"
count：标记数
displayOrder：显示顺序
alert：显示标记数，YES是显示标记数，NO是显示红点
定义：
-(void)updateCustomConversation:(NSString *)customKey
                      withTitle:(NSString *)title
                    withContent:(NSString *)content
                     withAvatar:(NSString *)avatarUrl
                 withUpdateTime:(NSString *)updateTime
                withBadegeCount:(int)count
               withDisplayOrder:(int)displayOrder
                      withAlert:(BOOL)alert;
示例代码：
[[MXChat sharedInstance] updateCustomConversation:@"test_custom_key" withTitle:@"testTitle"
 withContent:@"testContent" withAvatar:@"/photos/56dcd11e0de1756f0eb4ef3d4644307a" withUpd
ateTime:@"2015-11-19T12:08:50+08:00" withBadegeCount:10 withDisplayOrder:0 withAlert:YES];
```

注册自定义会话点击事件的回调

```
定义：
-(void)registCustomConversationClickCallback:(finishCallback)callback;
示例代码：
```

```
[[MXChat sharedInstance] registCustomConversationClickCallback:^(id result, MXError *error
) {
        if(!error) {
           NSDictionary *resultDic = (NSDictionary *)result;
           NSString *custom_key = [resultDic objectForKey:@"custom_key"];
         if(custom_key) {
           NSLog(@"custom_key == %@", custom_key);
           //test push view
          MXSettingViewController *vc = [[MXSettingViewController alloc] init];
           vc.hidesBottomBarWhenPushed = YES;
           [[[MXChat sharedInstance] getChatViewController].navigationController pushViewC
ontroller:vc animated:YES];
            }
        }
}];
```

通过appID打开会话，例如通过意见反馈

```
定义:
-(void)openConversationByOpenID:(NSString *)openID withNavigationVC:(UINavigationController
 *)navVC;
示例代码:
int index =  [self.tabBarController selectedIndex];
UINavigationController *nav = [self.tabBarController.viewControllers objectAtIndex:index];
[[MXChat sharedInstance] openConversationByOpenID:MX_FEEDBACK_APP_ID withNavigationVC:nav]
;
```

通过会话ID删除所选会话

```
参数说明:
conversationID: 会话ID
定义:
-(void)deleteSelectConversation:(long int)conversationID;
示例代码:
[[MXChat sharedInstance]deleteSelectConversation:1];
```

通过会话ID获取会话信息

```
定义:
-(NSDictionary *)getConversationByID:(long int)conversationID;
示例代码:
NSDictionary *dic=[[MXChat sharedInstance]getConversationByID:1];
```

通过消息ID获取消息信息

```
定义：
-(NSDictionary *)getMssagesByID:(int)messageID;
示例代码：
NSDictionary *dic=[[MXChat sharedInstance]getMssagesByID:1];
```

配置聊天详情的设置按钮

```
定义：
-(void)disableChatSetting:(BOOL)disable;
示例代码：
[[MXChat sharedInstance] disableChatSetting:YES];
```

发送自定义视频消息

```
参数说明：
conversationVO：会话信息字典
key：自定义视频插件的key
定义：
- (void)sendCustomVideoChatMessageWithConversation:(NSDictionary *)conversationVO key:(NSS
tring *)key;
示例代码：
NSDictionary *conversationDic = [NSMutableDictionary new];[conversationDic setValue:@(self
.conversationID) forKey:@"conversation_id"];
[conversationDic setValue:@(self.theConferenceMode) forKey:@"multi_user"];
[conversationDic setValue:@[@([self.online_persons firstObject].ID)] forKey:@"user_ids"];
。。。。。。
[[MXChat sharedInstance] sendCustomVideoChatMessageWithConversation:conversationDic key:@"
chat_video_status"];
```

聊天列表页自动获取下一个未读数消息(双击tabbar)

```
定义：
-(void) autoScroll2UnreadMessage;
示例代码：
[[MXChat sharedInstance] autoScroll2UnreadMessage];
```

敏信列表下拉刷新触发后会回调这个callback

```
定义：
-(void)registChatRefreshCallback:(functionCallback)callback;
示例代码：
[[MXChat sharedInstance]registChatRefreshCallback:^(id object) {
// 执行敏信列表下拉刷新后的操作
}];
```

敏信列表刷新完毕需要调用这个方法来更新下拉的状态

```
定义：
-(void)chatRefreshFinished;
示例代码：
[[MXChat sharedInstance]chatRefreshFinished];
```

注册聊天顶部banner的对象

```
定义：
//聊天窗口顶部预留扩展位，方便添加宣传内容或接入广告
-(void)registBannerObj:(id)bannerObj;
示例代码：
testConversationBannerObject *bannerObj = [[testConversationBannerObject alloc] init];
[[MXChat sharedInstance] registBannerObj:bannerObj];
```

移除聊天顶部banner的对象

```
定义：
-(void)removeBannerView;
示例代码：
[[MXChat sharedInstance] removeBannerView];
```

注册自定义聊天插件(聊天附件栏显示的插件应用)

```
定义：
-(void)registCutsomPlugIn:(id)plugIn;
示例代码：
//添加视频会议
MXVideoPluginObject *mvc=[[MXVideoPluginObject alloc] init];
[[MXChat sharedInstance] registCutsomPlugIn:mvc];
```

自定义消息，可以在群聊附件栏启动应用，发送链接，发送本地通讯录的名片等。需实现协议方法如下：

必须实现的协议方法

```
定义：
//获取聊天应用图标
-(UIImage *)getIconImage;
示例代码：
-(UIImage *)getIconImage {
```

```
    return [UIImage imageNamed:@"Icon-72"];
}
```

定义：

```
//获取聊天应用名字
-(NSString *)getName;
```

示例代码：

```
-(NSString *)getName {
    return @"test";
}
```

定义：

```
//获取聊天应用的key
-(NSString *)getPlugInKey;
```

示例代码：

```
-(NSString *)getPlugInKey {
    return @"test_plugin";
}
```

定义：

```
//聊天应用发起聊天
-(void)startAction:(NSString *)conversationJsonData withReCallback:(plugInReCallback)recal
lback;
```

示例代码：

```
-(void)startAction:(NSString *)conversationJsonData withReCallback:(plugInReCallback)recal
lback
{
    NSString *testData = @"testData";
    recallback(testData);
}
```

定义：

```
//获取插件视图
-(UIView *)getPlugInViewByData:(NSString *)data;
```

示例代码：

```
-(UIView *)getPlugInViewByData:(NSString *)data
{
UIView *bgView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 200, 80)];
......
return bgView;
}
```

定义：

```
//聊天页里点击插件消息
-(void)clickAction:(NSString *)data withView:(UIView *)view andPopViewController:(UIViewCo
ntroller *)viewC;
```

示例代码：

```
-(void)clickAction:(NSString *)data withView:(UIView *)view andPopViewController:(UIViewCo
ntroller *)viewC{

}
```

定义：

```
//接受消息的背景图片
-(UIImage *)getIncomingImage;
```

示例代码：

```
-(UIImage *)getIncomingImage{
```

```
    UIImage *image = [UIImage imageNamed:@"mx_bg_share_others_chat_phone"];
    return image;
}
定义:
//发消息的背景图片
-(UIImage *)getOutGoingImage;
示例代码:
-(UIImage *)getOutGoingImage{
    UIImage *image = [UIImage imageNamed:@"mx_bg_share_my_chat_phone"];
    return image;
}
定义:
//是否在附件栏显示
- (BOOL)showInAttachUtilView;
示例代码:
- (BOOL)showInAttachUtilView{
return YES;
}
```

可选择实现的协议方法

```
定义:
- (NSString *)getPluginMessgeType;
示例代码:
-(NSString *)getPluginMessgeType {
    return @"test";
}
定义:
- (BOOL)showInMiddle;
示例代码:
- (BOOL)showInMiddle{
return YES;
}
定义:
- (CGSize)getPluginViewSizeByData:(NSString *)data;
示例代码:
- (CGSize)getPluginViewSizeByData:(NSString *)data{
CGSize size=......;
return size;
}
定义:
- (BOOL)hiddenFromLastSeenData;
示例代码:
- (BOOL)hiddenFromLastSeenData{
return YES;
}
定义:
- (NSDictionary *)getSenderIDByBody:(NSString *)body;
示例代码:
- (NSDictionary *)getSenderIDByBody:(NSString *)body{
```

```
return dic;
}
```
定义：
```
//红包插件专用的方法，现在用于扩展选中、取消背景色的方法，字典里含有type（incomingcell、outgoingcell）、bubble
- (void)HandlePluginByDic:(NSDictionary *)dic;
```
示例代码：
```
- (void)HandlePluginByDic:(NSDictionary *)dic{
}
```
定义：
```
- (BOOL)isScreenShotMessage;
```
示例代码：
```
- (BOOL)isScreenShotMessage{
return NO;
}
```
定义：
```
//用于长按操作菜单显示的指定，返回对应菜单类型的int数组
- (NSArray *)itemsOfLongPressPopupMenuWithParameters:(NSDictionary *)dic;
```
示例代码：
```
- (NSArray *)itemsOfLongPressPopupMenuWithParameters:(NSDictionary *)dic{
}
```

我的收藏相关协议方法

定义：
```
//收藏中 展示的高度，建议在getPlugInViewInFavByData中复用该方法
- (CGSize)getPluginViewSizeInFavByData:(NSString *)data;
```
示例代码：
```
- (CGSize)getPluginViewSizeInFavByData:(NSString *)data{
    return CGSizeMake(appWidth-40, 60);
}
```
定义：
```
//插件消息在收藏页显示的视图
- (UIView *)getPlugInViewInFavByData:(NSString *)data;
```
示例代码：
```
- (UIView *)getPlugInViewInFavByData:(NSString *)data{
 //自定义视图
 return view;
}
```
定义：
```
//在收藏中点击
-(void)clickActionInFav:(NSString *)data withView:(UIView *)view andPopViewController:(UIViewController *)viewC;
```
示例代码：
```
-(void)clickActionInFav:(NSString *)dataStr withView:(UIView *)view andPopViewController:(UIViewController *)viewC{
 NSData *binData = [dataStr dataUsingEncoding:NSUTF8StringEncoding];
 NSDictionary *dicData = [NSJSONSerialization JSONObjectWithData:binData options:kNilOptions error:nil];
```

```
 NSDictionary * data = [dicData objectForKey:@"data"];
 MXLocationBrowseController * vc = [[MXLocationBrowseController alloc] init];
 vc.bodyString = dataStr;
MXLocationPOIVO * location = [[MXLocationPOIVO alloc] initWithDictionary:data];
vc.location = location;
viewC.searchDisplayController.active = NO;
[viewC.navigationController pushViewController:vc animated:YES];
}
```

MXContacts接口说明文档

1.引入头文件

```
#import <MXKit/MXContacts.h>
```

2.初始化MXContacts

2.1获取MXContacts对象

```
定义：
+(MXContacts *)sharedInstance;
示例代码：
[MXContacts sharedInstance];
```

2.2获取通讯录的ViewController

```
定义：
-(UIViewController *)getContactsViewController;
示例代码;
UIViewController *contacts = [[MXContacts sharedInstance] getContactsViewController];
```

2.3自定义通讯录的header

```
定义：
-(void)setCustomHeaderView:(UIView *)headerView;
示例代码：
[[MXContacts sharedInstance] setCustomHeaderView:contactsHeader];
```

2.4释放通讯录的viewController

```
定义：
-(void)cleanContactsViewController;
示例代码：
[[MXContacts sharedInstance] cleanContactsViewController];
```

3.MXContacts模块详细API使用说明

添加联系人

```
定义：
-(void)addContact;
示例代码：
[[MXContacts sharedInstance] addContact];
```

右滑contact界面的时候回调

```
定义：
-(void)startPullDownWithCallback:(finishCallback)callback;
示例代码：
[[MXContacts sharedInstance] startPullDownWithCallback:^(id result, MXError *error) {
if (result) {
   [contactsHeader startPullDown];
   }
}];
```

获取人员信息，并显示人员信息详情页

```
参数说明：
loginName：  是要获取的人员的登录名
VC：  是将要push聊天view的viewController
定义：
-(void)personInfo:(NSString *)loginName withIndicator:(BOOL)showIndicator withViewControll
er:(UIViewController *)VC;
示例代码：
[[MXContacts sharedInstance] personInfo:@"1065" withIndicator:YES withViewController:self]
;
```

获取人员信息，结果放在callback中返回

```
参数说明：
loginName 是要获取的人员的登录名
定义：
-(void)getPersonInfo:(NSString *)loginName withIndicator:(BOOL)showIndicator withCallback:
(finishCallback)callback;
示例代码：
[[MXContacts sharedInstance] getPersonInfo:@"1065" withIndicator:YES withCallback:^(id obj
ect, MXError *error) {
NSData* jsonData = [NSJSONSerialization dataWithJSONObject:(NSDictionary *)object options:
NSJSONWritingPrettyPrinted error:nil];

NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncodi
ng];
```

```
UIAlertView *alter = [[UIAlertView alloc] initWithTitle:@"信息" message:jsonString delegate:
nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
[alter show];
}];
```

选择人员，isMult = YES 是多选

```
参数说明:
isMult:  是否是多选模式
title:  标题
isAllDept:是否是所有部门
VC:传入当前控制器，用于跳转
定义:
-(void)selectUser:(BOOL)isMult withTitle:(NSString *)title withAllDept:(BOOL)isAllDept wit
hViewController:(UIViewController *)VC withCallback:(finishCallback)callback;
示例代码:
[[MXContacts sharedInstance] selectUser:YES withTitle:@"text" withAllDept:YES withViewCont
roller:self withCallback:^(id object, MXError *error) {
NSData* jsonData = [NSJSONSerialization dataWithJSONObject:(NSDictionary *)object
                                                   options:NSJSONWritingPrettyPrinted error:nil]
;
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncodi
ng];
UIAlertView *alter = [[UIAlertView alloc] initWithTitle:@"信息" message:jsonString delegate:
nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
[alter show];
}];
```

选择人员，从应用中心进入视频会议，isMult = YES 是多选（modal模式）

```
参数说明:
isMult: 是否是多选
title:标题
isAllDept:  是否是所有部门
VC:传入当前控制器，用于跳转
selectContacts:选择的人员
定义:
-(void)selectAVConferenceContact:(BOOL)isMult withTitle:(NSString *)title withAllDept:(BOOL
)isAllDept withViewController:(UIViewController *)VC selectContacts:(NSMutableArray *)sele
ctContacts  withCallback:(finishCallback)callback;
示例代码:
[[MXContacts sharedInstance]selectAVConferenceContact:YES withTitle:@"text" withAllDept:YES
 withViewController:self selectContacts:selectContactsArr withCallback:^(id object, MXErro
r *error) {
。。。。。。
```

```
}];
```

视频会议时用到的选人界面

```
定义：
-(void)selectAVConferenceContactWithTitle:(NSString *)title topViewController:(UIViewContr
oller *)vc usersIDS:(NSArray *)usersIDS selectContacts:(NSMutableArray *)selectContacts ca
llback:(finishCallback)callback;
示例代码：
[[MXContacts sharedInstance]selectAVConferenceContactWithTitle:@"text" topViewController:v
c usersIDS:usersIDSArr selectContacts:selectContactsArr callback:^(id object, MXError *err
or) {
。。。。。。
}];
```

获得来电显示设置界面

```
定义：
-(UIViewController *)getContactSettingViewController;
示例代码：
UIViewController *contactSetting=[[MXContacts sharedInstance] getContactSettingViewControl
ler];
```

清除来电显示设置

```
定义：
-(void)clearContactSettingViewController;
示例代码：
[[MXContacts sharedInstance] clearContactSettingViewController];
```

聊天界面点击名片，显示可选择的通讯录（手机通讯录或企业通讯录）

```
定义：
-(void)showSelectContacterView:(dataCallback)callback;
示例代码：
[[MXContacts sharedInstance]showSelectContacterView:^(id data) {
  NSLog(@"名片result===%@", data);
}];
```

配置通讯录发短信，敏信，打电话功能

```
定义：
```

```
//加在初始化敏行参数的位置
-(void)setPhoneMessageShow:(BOOL)phoneMessageShow withMessageShow:(BOOL)messageShow withPh
oneShow:(BOOL)phoneShow;
示例代码:
[[MXContacts sharedInstance] setPhoneMessageShow:YES withMessageShow:YES withPhoneShow:YES
];
```

设置群聊，公众号，公司通讯录以及特别关注的显示顺序，displayOrder越大越靠前

```
定义:
-(void)setMultiChatDisplayOrder:(int)displayOrder;
-(void)setPublicAccountDisplayOrder:(int)displayOrder;
-(void)setVipDisplayOrder:(int)displayOrder;
-(void)setContactDisplayOrder:(int)displayOrder;
示例代码:
[[MXContacts sharedInstance] setMultiChatDisplayOrder:0];
[[MXContacts sharedInstance] setPublicAccountDisplayOrder:1];
[[MXContacts sharedInstance] setVipDisplayOrder:2];
[[MXContacts sharedInstance] setContactDisplayOrder:3];
```

通讯录界面添加自定义cell(与群聊、公众号、特别关注、通讯录并齐)

```
参数说明:
key: 自定义通讯录的key
name: 显示的名字
avatar: 头像地址
displayOrder: 显示顺序
定义:
-(void)setCustomContactKey:(NSString *)key withName:(NSString *)name withAvatarUrl:(NSStri
ng *)avatar withDisplayOrder:(int)displayOrder;
示例代码:
[[MXContacts sharedInstance] setCustomContactKey:@"testCustomKey" withName:@"生活服务" with
AvatarUrl:@"" withDisplayOrder:2];
```

注册通讯录界面自定义cell点击的回调

```
定义:
-(void)registCustomContactClickCallback:(finishCallback)callback;
示例代码:
[[MXContacts sharedInstance] registCustomContactClickCallback:^(id key, MXError *error) {
   NSString *customKey = (NSString *)key;
   NSLog(@"custom contact click key ==== %@", customKey);
}];
```

显示公众号列表，VC是parent viewcontroller

定义：

```
-(void)showPublicAccount:(UIViewController *)VC;
```

示例代码：

```
[[MXContacts sharedInstance] showPublicAccount:[[MXContacts sharedInstance] getContactsVie
wController]];
```

显示保存的群聊列表，VC是parent viewcontroller

定义：

```
-(void)showMultiChat:(UIViewController *)VC;
```

示例代码：

```
[[MXContacts sharedInstance] showMultiChat:self];
```

启动隐藏手机号

定义：

```
-(void)initDisableShowPhoneNumber:(BOOL)disableshowPhoneNumber;
```

示例代码：

```
[[MXContacts sharedInstance] initDisableShowPhoneNumber:MX_DISABLE_SHOWPHONENUMBER];
```

启动隐藏添加联系人功能

定义：

```
-(void)initDisableAddContacts:(BOOL)disableAddContacts;
```

示例代码：

```
[[MXContacts sharedInstance] initDisableAddContacts:MX_DISABLE_ADDCONTACTS];
```

是否隐藏在线状态

定义：

```
-(void)initEnableOnlineStatus:(BOOL)enableOnlieStatus;
```

示例代码：

```
[[MXContacts sharedInstance] initEnableOnlineStatus:MX_ENABLE_ONLINE_STATUS];
```

MXAppCenter接口说明文档

1.引入头文件

```
#import <MXKit/MXAppCenter.h>
```

2.初始化MXAppCenter

2.1获取MXAppCenter对象

```
定义:
+(MXAppCenter *)sharedInstance;
示例代码:
[MXAppCenter sharedInstance];
```

2.2获取应用中心的ViewController

```
定义:
-(UIViewController *)getAppCenterViewController;
示例代码:
UIViewController *appCenter = [[MXAppCenter sharedInstance] getAppCenterViewController];
```

2.3自定义应用中心的headerView

```
定义:
-(void)setCustomHeaderView:(UIView *)headerView;
示例代码:
[[MXAppCenter sharedInstance] setCustomHeaderView:appCenterHeader];
```

2.4释放应用中心的viewController

```
定义:
-(void)cleanAppCenterViewController;
示例代码:
[[MXAppCenter sharedInstance] cleanAppCenterViewController];
```

3.MXAppCenter模块详细API使用说明

删掉自定义应用中心上部的view

```
定义：
-(void)removeCustomAppHeaderView;
示例代码：
[[MXAppCenter sharedInstance]removeCustomAppHeaderView];
```

右滑appCenter界面的时候回调

```
定义：
-(void)startPullDownWithCallback:(finishCallback)callback;
示例代码：
[[MXAppCenter sharedInstance] startPullDownWithCallback:^(id result, MXError *error) {
   if (result) {
    [appCenterHeader startPullDown];
   }
}];
```

设置某个应用中心上部某个特定的应用或者轮播图的view，如果设置开启轮播图，这个view会出现在轮播图上方。

```
定义：
-(void)setCustomAppHeaderView:(UIView *)appHeaderView;
示例代码：
[[MXAppCenter sharedInstance] setCustomAppHeaderView:testAppHeaderview];
```

设置应用中心轮播图下方的自定义View

```
定义：
-(void)setCustomAppHeaderViewBelowBanner:(UIView *)appHeaderView;
示例代码：
[[MXAppCenter sharedInstance] setCustomAppHeaderViewBelowBanner:testAppHeaderview];
```

设置自定义应用中心上部刷新view的背景颜色

```
定义：
-(void)setCustomAppRefreshViewBackgroundColor:(UIColor *)bgColor;
示例代码：
[[MXAppCenter sharedInstance] setCustomAppRefreshViewBackgroundColor:[UIColor mx_colorWith
HexString:@"#ff5300"]];
```

设置应用中心刷新view的字体颜色

```
定义：
-(void)setCustomAppRefreshFontColor:(UIColor *)fontColor;
示例代码：
[[MXAppCenter sharedInstance] setCustomAppRefreshFontColor:[UIColor whiteColor]];
```

设置应用中心刷新View的菊花颜色

```
定义：
- (void)setCusomAppRefreshActivityColor:(UIColor *)color;
示例代码：
[[MXAppCenter sharedInstance]setCusomAppRefreshActivityColor:[UIColor yellowColor]];
```

设置应用中心应用的自动下载的大小的最大值,默认100k以下的应用会自动安装

```
定义：
-(void)setAutoDownloadMaxSize:(NSInteger)maxSize;
示例代码：
[[MXAppCenter sharedInstance] setAutoDownloadMaxSize:MX_APP_AUTO_DOWNLOAD_LIMIT_SIZE];
```

跳转到添加app页面

```
定义：
-(void)addApp;
示例代码：
[[MXAppCenter sharedInstance] addApp];
```

自定义的应用中心的viewcontroller跳转到添加app页面

parentvc：当前控制器

```
定义：
-(void)addApp:(UIViewController *)parentVC;
示例代码：
[[MXAppCenter sharedInstance] addApp:parentvc];
```

结束编辑模式

```
定义：
-(void)endEdit;
示例代码：
[[MXAppCenter sharedInstance] endEdit];
```

注册应用中心编辑开始的callback，编辑开始的时候会回调callback里面得代码

```
定义：
//编辑模式启动的时候，用户可以对应用进行排序
-(void)registEditBeginCallback:(finishCallback)callback;
示例代码：
[[MXAppCenter sharedInstance] registEditBeginCallback:^(id result, MXError *error) {
[navBarView setRightBtnImage:@"mx_finish_EN_phone"];
if ([[[NSLocale preferredLanguages] firstObject] hasPrefix:@"zh-"]){
  [navBarView setRightBtnImage:@"mx_finish_phone"];
 }
  editing = YES;
}];
```

注册应用中心编辑结束的callback，编辑结束的时候会回调callback里面得代码

```
定义：
-(void)registEditEndCallback:(finishCallback)callback;
示例代码：
[[MXAppCenter sharedInstance] registEditEndCallback:^(id result, MXError *error) {
editing = NO;
[navBarView setRightBtnImage:@"mx_btn_add_phone"];
 }];
```

应用中心的app加载完成之后回调callback里面得代码,第一次登录时应用中心加载完后会调用

```
定义：
-(void)registGetAppFinishCallback:(finishCallback)callback;
示例代码：
[[MXAppCenter sharedInstance] registGetAppFinishCallback:^(id result, MXError *error) {
  [navBarView setRightBtnUserInteractionEnabled:YES];
}];
```

注册应用中心的应用点击的事件，当用户点击应用中心的应用之后，会回调callback里面得方法，用户可以自由处理，处理完成之后可以调用appItemClick:(NSString*)appID来调用敏行的逻辑

```
定义：
-(void)registAppClickedCallback:(finishCallback)callback;
示例代码：
[[MXAppCenter sharedInstance] registAppClickedCallback:^(id result, MXError *error) {
```

```
 NSString *appID = [result copy];
 NSLog(@"appID===%@", appID);
 [[MXAppCenter sharedInstance] appItemClick:appID];
}];
```

点击未安装或未升级的应用时，返回这个应用的url及大小

```
定义:
- (void)registAppInstallOrUpgradeCallback:(finishCallback)callback;
示例代码:
[[MXAppCenter sharedInstance] registAppInstallOrUpgradeCallback:^(id result, MXError *erro
r) {
NSLog(@"%@",result);
 }];
```

点击应用公众号消息跳转应用时的回调接口信息,触发时传入应用公众号对应的应用id，可以自己在回调中根据appid判断调整跳转逻辑

需要在初始化聊天控制器（demo里在showmain里）时注册该回调

```
定义:
-(void)registPublicAccountClickedCallback:(handleNativeCallback)callback;
示例代码:
[[MXAppCenter sharedInstance] registPublicAccountClickedCallback:^(id result, MXError *err
or) {
 "app_url" = "";
 url = "http://test.dehuinet.com:8030/dehuinet/apps/sso_redirect?ocu_id=ccb74893d3c1402852
4e9751a3770381&url=/mxpp/articles/29717?source_id=235898";
NSLog(@"result===%@", result);
 return NO;//返回NO表示第三方使用者自己已经处理，跳过敏行逻辑；YES表示继续进入敏行处理逻辑。
}];
```

外部处理完成之后再进入app

```
定义:
-(void)appItemClick:(NSString *)appID;
//extParam：传入参数形式a=1&b=2
-(void)appItemClick:(NSString *)appID withExtParame:(id)extParam;
示例代码:
[[MXAppCenter sharedInstance] appItemClick:appID];
[[MXAppCenter sharedInstance] appItemClick:appID withExtParame:@"a=1&b=2"];
```

设置应用中心各个应用的消息未读数

参数说明：
countDic参数里面存的是appID和未读数的key value
定义：
```
-(void)showBadgeCount:(NSDictionary *)countDic;
```
示例代码：
```
NSDictionary *dicCount = [[NSDictionary alloc] initWithObjectsAndKeys:[NSNumber numberWithInt:20],@"bde209e19cfebcfa2f719a52a4406441",[NSNumber numberWithInt:15],@"f3710a8b00b4924057446b53609b5038", nil];
[[MXAppCenter sharedInstance] showBadgeCount:dicCount];
```

设置应用中心某个应用隐藏

参数说明：
appsArray里面存得是appID的数组
定义：
```
-(void)hiddenApps:(NSArray *)appsArray;
```
示例代码：
```
[[MXAppCenter sharedInstance] hiddenApps:[NSArray arrayWithObjects:@"bde209e19cfebcfa2f719a52a4406441", nil]];
```

用户自定义appview的frame

定义：
```
//用户自定义frame，这样appview就可以放到指定的任意位置
-(void)setCustomViewFrame:(CGRect)frame;
//设置appCenter的ParentController，这个api需要跟上面的setCustomViewFrame配套使用。
-(void)setParentController:(UIViewController *)controller;
```
示例代码：
```
[[MXAppCenter sharedInstance] setCustomViewFrame:CGRectMake(0, 100, 320, 100)];
[[MXAppCenter sharedInstance] setParentController:self];
```

是不是只有一个应用中心， YES表示只有一个应用中心

定义：
```
-(void)setOnlyOneAppCenter:(BOOL) onlyOneAppCenter;
```
示例代码：
```
[[MXAppCenter sharedInstance]setOnlyOneAppCenter:YES];
```

手动调api来获取应用中心的应用信息, callback 里面得object 是一个包含NSDictionary的NSArray，每一个NSDictionary里面是每个应用的信息

定义：

```
-(void)getAppsMessageWithFinshCallback:(finishCallback)callback;
示例代码:
[[MXAppCenter sharedInstance] getAppsMessageWithFinshCallback:^(id object, MXError *error)
 {
//object 里面是NSDictionary的NSArray,  NSDictionary的结构是key值为{avatar_url, name, ocuID, id, operationUrl}
 NSLog(@"apps message result == %@", object);
}];
```

点击某个应用的事件

```
定义:
-(void)appClick:(NSString *)appID WithExtParam:(NSString *)extParam withViewController:(UIViewController*)VC;
示例代码:
[[MXAppCenter sharedInstance] appClick:appID WithExtParam:nil withViewController:self];
```

用接口点击没有安装的应用的时候的回调,在非应用中心页面用接口点击没有安装的应用，弹出alert，点击确定后会回调

```
定义:
-(void)registClickAppFailCallback:(finishCallback)callback;
示例代码:
[[MXAppCenter sharedInstance] registClickAppFailCallback:^(id result, MXError *error) {
NSString *appID = (NSString *)result;
NSLog(@"app not intall or need upgrade!!!!!!!, appID == %@", appID);
//跳转到应用中心
[tabBarController setSelectedIndex:[self getAppCenterIndex]];
}];
```

切换内网

```
定义:
-(void)setSwitchNetworkInAppCenter:(BOOL)flag;
示例代码:
[[MXAppCenter sharedInstance] setSwitchNetworkInAppCenter:MX_ENABLE_NETWORK_CHANGE];
```

更新应用中心应用的角标数字

```
定义:
-(void)updateApp:(NSString *)app_id withBadgeCount:(int)badgeCount;
示例代码:
[[MXAppCenter sharedInstance]updateApp:appID withBadgeCount:0];
```

更新应用会话的角标数字

定义：

```
-(void)updateAppConversationFromAPI:(NSString *)app_id withBadgeCount:(int)badgeCount;
```

示例代码：

```
[[MXAppCenter sharedInstance]updateAppConversationFromAPI:appID withBadgeCount:0];
```

应用中心的设置

```
//应用中心禁止移动
[[MXAppCenter sharedInstance]initDiableAppItemMove:NO];
//设置应用中心增加右下角添加按钮
[[MXAppCenter sharedInstance]initEnableAppCenterAddButton:YES];
//是否显示应用中心的轮播图
[[MXAppCenter sharedInstance]initEnableAppCenterBanner:YES];
//设置一行显示几个app应用
[[MXAppCenter sharedInstance]initShowLineAppCount: 4];
//设置应用中心首页每行的item的数目
[[MXAppCenter sharedInstance] initShowMainAppCount:4];
//设置ipa竖屏时一行显示几个app应用
[[MXAppCenter sharedInstance]initShowLineAppCountIpadVertical:5];
//设置ipa横屏时一行显示几个app应用
[[MXAppCenter sharedInstance]initShowLineAppCountIpadHorizontal: 8];
//应用中心是否是分类展示
[[MXAppCenter sharedInstance] initShowApPItemByCategory:MX_SHOW_APP_CATEGORY];
// 禁用应用中心划线
[[MXAppCenter sharedInstance]initDisableDrawLine:NO];
```

注册滚动偏移量的回调，callback的返回的数据是一个NSDictionary，分别代表滚动的方向和滚动停止的y轴坐标值

定义：

```
-(void)registScrollOffsetCallback:(functionCallback)callback;
```

示例代码：

```
[[MXAppCenter sharedInstance] registScrollOffsetCallback:^(NSDictionary *dataDic) {
if(dataDic) {
int scrollDiection = [[dataDic objectForKey:@"scrollDirection"] intValue];
CGFloat scrollYOffset = [[dataDic objectForKey:@"scrollYOffset"] floatValue];
UIScrollView *appScrollView = [[MXAppCenter sharedInstance] getAppScrollView];
if(scrollDiection == 1) {
//down
if(scrollYOffset > -appCustomHeaderViewHeight && scrollYOffset < -5) {
    [UIView animateWithDuration:0.5 animations:^{
     testAppHeaderview.alpha = 1.0f;
     appScrollView.contentOffset = CGPointMake(0, -appCustomHeaderViewHeight);
   [[[MXAppCenter sharedInstance] getAppCenterViewController].view layoutIfNeeded];
```

```
      }];
   }
} else if(scrollDiection == 2) {
//up
if(scrollYOffset > -appCustomHeaderViewHeight + 5 && scrollYOffset < 0) {
    [UIView animateWithDuration:0.5 animations:^{
    testAppHeaderview.alpha = 1.0f;
    appScrollView.contentOffset = CGPointMake(0, 0);
   [[[MXAppCenter sharedInstance] getAppCenterViewController].view layoutIfNeeded];
      }];
    }
  } else {
      testAppHeaderview.alpha = 1.0f;
       }
    }
}];
```

应用中心下拉刷新触发后会回调这个callback

```
定义:
-(void)registAppCenterRefreshCallback:(functionCallback)callback;
示例代码:
[[MXAppCenter sharedInstance] registAppCenterRefreshCallback:^(id obj) {
 //do some refresh
 [[MXAppCenter sharedInstance] thirdAppRefreshFinished];
}];
```

第三方应用刷新完毕需要调用这个方法来更新应用中心下拉的状态

```
定义:
-(void)thirdAppRefreshFinished;
示例代码:
[[MXAppCenter sharedInstance] thirdAppRefreshFinished];
```

获得应用的ScrollView

```
定义:
-(UIScrollView *)getAppScrollView;
示例代码:
[[MXAppCenter sharedInstance]getAppScrollView];
```

设置app图标大小

```
定义:
```

```
-(void)setAppIconSize:(CGFloat)size;
示例代码:
[[MXAppCenter sharedInstance]setAppIconSize:size];
```

刷新nativeapp的状态，版本检测,在应用中心viewwillappear中调用,防止安装完第三方app之后安装状态不对

```
定义:
-(void)refreshNativeAppStatus;
示例代码:
[[MXAppCenter sharedInstance]refreshNativeAppStatus];
```

设置请求超时时间间隔 单位是秒

```
定义:
-(void)setRequsetTimeoutInterval:(NSTimeInterval)timeoutInterval;
示例代码:
[[MXAppCenter sharedInstance]setRequsetTimeoutInterval:30s];
```

刷新应用中心view

```
定义:
-(void)refreshAppsView;
示例代码:
[[MXAppCenter sharedInstance]refreshAppsView];
```

获取应用中心应用下载进度

```
定义:
-(void)registAppDownloadingStatus:(functionCallback)callback;
示例代码:
[[MXAppCenter sharedInstance] registAppDownloadingStatus:^(id object) {
  NSLog(@"appdownloadStatus======%@", object);
}];
```

检测license是否可用

```
参数说明:
licenseCode：license编码
定义:
//yes:可用  NO：不可用
-(BOOL)checkLicense:(NSString *)licenseCode;
```

```
示例代码：
BOOL enabled = [[MXAppCenter sharedInstance] checkLicense:@"video_chat"];
```

检测appItem是否需要密码认证

```
appDic：app信息
定义：
- (BOOL)checkAppItemAuthenticationWithDic:(NSMutableDictionary *)appDic;
示例代码：
BOOL enabled = [[MXAppCenter sharedInstance] checkAppItemAuthenticationWithDic:dic];
```

清理应用中心缓存

```
参数说明：
loginName：  登录名
定义：
-(void)cleanAppItemsCache:(NSString *)loginName;
示例代码：
[[MXAppCenter sharedInstance] cleanAppItemsCache:@"小明"];
```

处理插件点击事件

```
定义：
-(void)handleAppClickWithSchemeUrl:(NSString *)schemeUrl withExtParm:(NSString *)extParams
 withParentVC:(UIViewController*)parentVC withActionCallback:(functionCallback)actionCallb
ack;
示例代码：
NSString *schemeURL=@"video://";
[[MXAppCenter sharedInstance] handleAppClickWithSchemeUrl:schemeURL withExtParm:nil withPa
rentVC:self.viewController withActionCallback:^(id object) {

}
```

获取应用config配置

```
参数说明：
app_id：应用id
定义：
-(NSString *)loadNativeConfig:(NSString *)app_id;
示例代码：
NSString *pluginConfig=[appcenter loadNativeConfig:appid];
NSDictionary *configDic=[pluginConfig JSONValue];
```

获取应用名字

参数说明:
appid: 应用id
定义:

```
- (NSString *)getAppNameByAppID:(NSString *)appid;
```

示例代码:

```
NSString *appName=[appcenter getAppNameByAppID:appid];
```

MXCircle接口说明文档

1.引入头文件

```
#import <MXKit/MXCircle.h>
```

2.初始化MXCircle

2.1获取MXCircle对象

```
定义:
+(MXCircle *)sharedInstance;
示例代码:
[MXCircle sharedInstance];
```

2.2获取工作圈的ViewController

```
定义:
-(UIViewController *)getCircleViewController;
示例代码:
UIViewController *workCircle = [[MXCircle sharedInstance] getCircleViewController];
```

2.3自定义工作圈的headerView

```
定义:
-(void)setCustomHeaderView:(UIView *)headerView;
示例代码:
//如果是用tabbar加载circle那么就需要设置tabbar的高度
[[MXCircle sharedInstance] setTabbarHeight:tabbarHeight];
[[MXCircle sharedInstance] setCustomHeaderView:circleHeader];
```

释放工作圈的viewController

```
定义:
-(void)cleanCircleViewController;
示例代码:
[[MXCircle sharedInstance] cleanCircleViewController];
```

3.MXCircle模块详细API使用说明

获取当前工作圈信息

定义：

```
-(NSDictionary *)getCurrentGroupDic;
```

示例代码：

```
NSDictionary *groupDic = [[MXCircle sharedInstance] getCurrentGroupDic];
```

右滑circle界面时的回调

定义：

```
-(void)startPullDownWithCallback:(finishCallback)callback;
```

示例代码：

```
[[MXCircle sharedInstance] startPullDownWithCallback:^(id result, MXError *error) {
    if (result) {
    [circleHeader startPullDown];
  }
}];
```

注册工作圈群组切换的回调

定义：

```
-(void)registGroupSelect:(finishCallback)callback;
```

示例代码：

```
[[MXCircle sharedInstance] registGroupSelect:^(id result, MXError *error){
  //result里面是title的名字
 [circleHeader updateGroupTitle:[(NSDictionary *)result objectForKey:@"groupName"]];
 [circleHeader popupGroupSwitcherDismissed];
}];
```

显示工作圈群组列表，这个api会把工作圈列表显示在你想要显示的地方，从button开始显示，显示在哪个view里

参数说明：

button：当前点击的按钮

view：当前视图

定义：

```
-(void)showGroupTableViewFromButton:(UIButton *)button InView:(UIView *)view;
```

示例代码：

```
//groupChange 当显示群组的button按下的时候触发的事件
-(void) groupChange:(id)sender {
  UIButton *btn = (UIButton*)sender;
  if (!btn.selected) {
[[MXCircle sharedInstance] showGroupTableViewFromButton:btn InView:self];
  }
}
```

工作圈群组列表消失

```
定义:
-(void)dissmissGroupTableView;
示例代码:
[[MXCircle sharedInstance]dissmissGroupTableView];
```

注册工作圈有新消息的callback通知

```
定义:
-(void)registGroupRedIcon:(finishCallback)callback;
示例代码:
[[MXCircle sharedInstance] registGroupRedIcon:^(id result, MXError *error) {
  //显示工作圈新消息提示，比如出现新消息的小红点儿
  NSLog(@"show circel === %d", [result boolValue]);
  unreadMessageView.hidden = ![result boolValue];
```

初始化的时候去获取是否应该显示工作圈的红点儿

```
定义:
-(BOOL)getCircleRedIcon;
示例代码:
BOOL showRedIcon = [[MXCircle sharedInstance] getCircleRedIcon];
```

新建工作圈消息

```
参数说明:
messageType是工作圈的类型
typedef enum {
    MESSAGE_TYPE_TEXT,//文本
    MESSAGE_TYPE_TASK,//待办
    MESSAGE_TYPE_ACTIVITY,//活动
    MESSAGE_TYPE_POLL,//投票
    MESSAGE_TYPE_ANNOUNCE,//公告
} MESSAGE_TYPE;
remoteUrl:  现在传入nil，将来做扩展用
title:  新建的工作圈消息的title
定义:
-(void) createNewMessageWithType:(int) messageType remote:(NSString *)remoteUrl title:(NSS
tring *)titles;
示例代码:
//这个方法是说单击各个新建菜单触发的行为
-(void)selectedMenuItemAppVo:(MXAppsVO *)appVo{
    MESSAGE_TYPE type;
```

```
    MXCircle *circle = [MXCircle sharedInstance];
    if([appVo.name isEqualToString: @"update"]){
        type = MESSAGE_TYPE_TEXT;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
    if([appVo.name isEqualToString:@"mini_task"]){
        type = MESSAGE_TYPE_TASK;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
    if([appVo.name isEqualToString:@"event"]){
        type = MESSAGE_TYPE_ACTIVITY;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
    if([appVo.name isEqualToString:@"poll"]){
        type = MESSAGE_TYPE_POLL;
        [circle createNewMessageWithType:type remote:nil title:appVo.text];
        return;
    }
}
```

发布消息到工作圈

参数说明:
type 类型如下

```
typedef enum {
    MESSAGE_TYPE_TEXT,//文本
    MESSAGE_TYPE_TASK,//待办
    MESSAGE_TYPE_ACTIVITY,//活动
    MESSAGE_TYPE_POLL,//投票
    MESSAGE_TYPE_ANNOUNCE,//公告
} MESSAGE_TYPE;
```

content：是消息的内容
VC：是要push这个新建工作圈消息的view的viewController
定义:

```
-(void)sendToCircle:(int)type withContent:(NSString *)content withViewController:(UIViewCo
ntroller *)VC;
```

示例代码:

```
[[MXCircle sharedInstance] sendToCircle: MESSAGE_TYPE_TEXT withContent:@"测试新建文本消息" w
ithViewController:vc];
```

分享到工作圈接口

参数说明:
title:标题

```
description：详细描述
url：http链接地址
nativeURL：native链接地址
thumbnailURL：缩列图链接地址
VC：分享成功push到的页面
定义：
-(void)shareTitle:(NSString *)title withDescription:(NSString *)description withURL:(NSString *)url withNativeURL:(NSString *)nativeURL withThumbnailURL:(NSString *)thumbnailURL withViewController:(UIViewController *)VC;
示例代码：
[[MXCircle sharedInstance] shareTitle:@"test" withDescription:@"test" withURL:@"https://www.baidu.com" withNativeURL:nil withThumbnailURL:@"http://img1.gtimg.com/13/1301/130137/13013760_980x1200_0.jpg" withViewController:self];
```

分享到工作圈接口，不带UI, 分享的结果在callback里回调，成功返回@"success"， 失败，返回errorstring

```
参数说明：
title:标题
description:详细描述
url：http链接地址
nativeURL：native链接地址
thumbnailURL：缩列图链接地址
groupID:工作圈ID
callback:分享后的回调
定义：
-(void)shareTitle:(NSString \*)title withDescription:(NSString \*)description withURL:(NSString \*)url withNativeURL:(NSString \*)nativeURL withThumbnailURL:(NSString \*)thumbnailURL withGroupID:(int)groupID withCallback:(finishCallback)callback;
示例代码：
[[MXCircle sharedInstance] shareTitle:@"test" withDescription:@"test" withURL:@"www.baidu.com" withNativeURL:nil withThumbnailURL:@"http://img1.gtimg.com/13/1301/130137/13013760_980x1200_0.jpg" withGroupID:163 withCallback:^(id object, MXError *error) {
  if (object) {
  UIAlertView *alter = [[UIAlertView alloc] initWithTitle:@"信息" message:object delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
    [alter show];
   }
}];
```

设置显示指定的工作圈

```
定义：
//获取已加入的工作圈列表,遍历数组，获得字典，每一个字典里包含groupID和groupName的信息
-(NSArray *)getJoinedGroupArray;
//通过传入groupID参数切换工作圈
-(void)setCurrentGroup:(int)groupID;
```

```
//强制刷新工作圈，按照已经设置的currentgroupid来刷新
-(void)froceRefrushCircle;
示例代码:
NSArray * array = [[MXCircle sharedInstance] getJoinedGroupArray];
NSDictionary * dic=array[0];
int groupID=array[0]@["groupID"];
[[MXCircle sharedInstance]setCurrentGroup:groupID];
[[MXCircle sharedInstance]froceRefrushCircle];
```

获得话题列表

```
定义:
-(UIViewController *)getTopicViewController;
示例代码:
UIViewController * vc = [[MXCircle sharedInstance] getTopicViewController];
```

通过groupID设置当前群组

```
定义:
-(void)setCurrentGroup:(int)groupID;
示例代码:
[[MXCircle sharedInstance]setCurrentGroup:45];
```

强制刷新工作圈，按照已经设置的currentgroupid来刷新

```
定义:
-(void)froceRefrushCircle;
示例代码:
[[MXCircle sharedInstance]froceRefrushCircle];
```

启动不限制点赞人数显示

```
定义:
-(void)initDisableLimitShowPrise:(BOOL)disableLimitShowPrise;
示例代码:
//启动不限制点赞人数显示.  YES为完整显示  NO为不完整显示
[[MXCircle sharedInstance] initDisableLimitShowPrise:MX_DISABLE_LIMITSHOWPRAISE];
```

设置分享接口

```
定义:
-(void)setShareToXXX:(NSString *)share2XXXUrl;
示例代码:
```

```
[[MXCircle sharedInstance] setShareToXXX:url];
```

可配置工作圈弹出的横条(赞、回复、分享等)

```
定义:
-(void)setAction:(NSArray *)action;
示例代码:
#ifdef MX_Circle_Action
NSData *data = [MX_Circle_Action dataUsingEncoding:NSUTF8StringEncoding];
NSArray *arr = nil;
if (data) {
arr = [NSJSONSerialization JSONObjectWithData:data options:kNilOptions error:nil];
NSLog(@"dic ============   %@",arr);
}
[[MXCircle sharedInstance] setAction:arr];
[MXObj registCustomActionCallback:^(id object, MXError *error) {

}];
#endif
```

MXNetModel接口说明文档

1.引入头文件

```
#import <MXKit/MXNetModel.h>
```

2.敏行API通道

```
参数说明:
method:为HTTP得方法有这么几种 "GET", "POST", "DELETE", "PUT"
url:为http请求的api方法，可以是相对路径，也可以是绝对路径
params:请求的参数
headers:请求头
showIndiator:为是否显示等待提示框
invoke:为response处理block
定义:
-(void)startRequset:(NSString *)method withURL:(NSString *)url withParams:(NSDictionary *)
params withHeaders:(NSDictionary *)headers withIndicator:(BOOL)showIndiator withCallback:(
requsetCallback) invoke;
示例代码:
[[MXNetModel shareNetModel] startRequset:@"GET" withURL:"" withParams:nil withHeaders:nil
withIndicator:YES withCallback:^(id object, MXError *error) {
 //handle callback
 if(object && !error) {
   //处理response
    } else {
    //出错处理
   }
 }];
```

常见问题

1