

微博 iOS 平台 SDK 文档

编号：WEIBO_IOS_SDK

版本：WEIBO_IOS_SDK V3.3.0

修订记录：

| 时间 | 文档版本 | 修订人 | 备注 |
|------------|-------|-----|------------------------------|
| 2012/07/19 | 1.0.0 | 陈行政 | 初稿 |
| 2013/01/30 | 1.0.1 | 陈行政 | 文档整合 |
| 2013/01/31 | 1.1.0 | 陈行政 | 文档更新 |
| 2013/03/12 | 2.0.0 | 洪涛 | SDK 升级到 2.0 版本 |
| 2013/04/16 | 2.1.0 | 唐庆杰 | SDK 升级到 2.1 版本 |
| 2013/09/09 | 2.3.0 | 唐庆杰 | 新增登陆登出按钮、好友邀请功能 |
| 2013/10/20 | 2.3.2 | 洪涛 | 重写文档，添加完整 SDK 指引 |
| 2014/11/17 | 3.0.0 | 邱文杰 | SDK 升级到 3.0 版本 |
| 2015/02/04 | 3.1.0 | 李靖宇 | 新增短信注册通道 |
| 2016/05/11 | 3.1.4 | 李靖宇 | 新增私信分享，支持 ipv6-only |
| 2017/4/19 | 3.2.0 | 李靖宇 | SDK 升级到 3.2 版本 |
| 2017/7/31 | 3.2.1 | 李靖宇 | SDK 新增多图，视频分享以及分享到微博 story |
| 2019/9/18 | 3.2.5 | 王美美 | 替换 UIWebView，并优化 SDK |
| 2019/12/17 | 3.2.6 | 王美美 | 修复 issue 中存在的问题，并且文档增加常见问题部分 |

| | | | |
|------------|-------|-----|--|
| 2020/3/18 | 3.2.7 | 王美美 | 替换旧的 API |
| 2020/10/10 | 3.3.0 | 吴锦润 | 支持 UniversalLink 分享跳转，修复 ios14 分享多图和视频丢数据问题，去掉分享使用相册功能 |
| 2021/06/29 | 3.3.1 | 吴锦润 | 支持分享超话 |
| 2021/10/12 | 3.3.2 | 吴锦润 | 支持分享 livephoto |
| 2022/05/06 | 3.3.3 | 吴锦润 | 下线 sdk 获取设备信息，如 idfa 等 |
| 2022/9/9 | 3.3.4 | 谢兴达 | 1.底层 app 跳转策略重构，防止跳转失败问题 2. 底层数据传输方式重构，用来兼容 iOS16 剪贴板问题。 3.解决当前已知 bug，建议所有用户尽快更新到最新版本。 |
| 2023/12/08 | 3.3.6 | 吴锦润 | 已知问题修复 |
| 2023/12/19 | 3.3.7 | 吴锦润 | 改善 SDK 稳定性，增强性能和安全性 |

目录

| | |
|--|----|
| Weibo SDK | 5 |
| 一..... | S |
| DK 接入设置 | 5 |
| 1. 注册成为开发者，创建移动应用 | 5 |
| 2. 设定授权回调页 | 6 |
| 3. 设定 Apple ID 和 Bundle ID | 6 |
| 4. 设置工程回调 URL Scheme | 7 |
| 5. 设置工程回调 universalLink | 8 |
| 6. 添加 SDK 文件到工程 | 9 |
| 7. 设置工程的 objC 编译选项 | 10 |
| 8. 添加 FrameWork 文件到工程 | 10 |
| 9. 定义应用 SSO 登录及 Oauth2.0 认证所需常量 | 11 |
| 10. 注册 appkey(clientid)和 universalLink | 11 |
| 11. 重写 AppDelegate 的 handleOpenURL 和 openURL 和 continueUserActivity 方法 | 11 |
| 二、应用场景代码示例 | 12 |
| 1. SSO 微博客户端授权认证 | 12 |
| 2. 第三方应用向微博发送请求，完成分享 | 13 |
| 3. 微博调起第三方, 向第三方请求数据 | 13 |
| 4. 用户取消对应用的授权 | 14 |
| 5. aid 相关 aid 是设备唯一的移动设备指纹 | 15 |
| 6. SDK 连接到微博 | 16 |
| 7. SDK 分享多图、视频到微博 | 16 |
| 8. SDK 提供分享单图、视频到 story | 17 |
| 三、常见问题 | 18 |
| 1..... | r |
| registerAppWithApp 使用 UIPastboard 导致主线程卡死问题 | 18 |
| 2..... | 微 |
| 博 SDK 需要放在 main bundle | 18 |
| 3..... | J |
| SONKit 报错问题 | 19 |

| | |
|--------------------------|----|
| 4..... | 微 |
| 博 SDK 授权后为什么没有返回应用 | 20 |

Weibo SDK

一. SDK 接入设置



1. 注册成为开发者，创建移动应用

如果你还不是一名开发者，请登录微博开放平台注册成为开发者。成为开发者之后，方可集成微博 sdk，使用微博 sdk 提供的丰富的功能，详细的集成过程可以参考新手指南：<http://open.weibo.com/wiki/%E6%96%B0%E6%89%8B%E6%8C%87%E5%8D%97>。

在创建应用时，开发者需要谨慎选择应用对应平台，不同的平台建议使用不同 APPKEY 开发。

应用平台：☐ iPhone ☐ Android ☐ BlackBerry
☐ Windows Phone ☐ Symbian ☐ WebOS
☐ Other

本文档读者请选择 iPhone

2. 设定授权回调页

请在“我的应用 - 应用信息 - 高级信息”中填写您的应用回调页，这样才能使 OAuth2.0 授权正常进行。如果您的 APPSECRET 发生泄露，您也可以通过该页面中的重置按钮对其重置，如下图所示：

OAuth2.0 授权设置 编辑

授权回调页: <https://api.weibo.com/oauth2/default.html>

取消授权回调页: <https://api.weibo.com/oauth2/default.html>

安全设置 编辑

应用的服务IP地址: 未填写

重置App Secret

cf95889f127786279e4a7ad286258949 重置

如果App Secret泄露，可以通过重置更换，原App Secret将作废

注意： iOS 应用推荐使用默认授权回调页！地址为：

<https://api.weibo.com/oauth2/default.html>

3. 设定 Apple ID 和 Bundle ID

请在“我的应用 - 应用信息 - 基本信息”中填写您个人或公司的 Apple ID 和应用的 Bundle ID，填写信息是保证您的应用可以正常使用微博 iOS SDK 提供的授权和回调。（更改设置有延时，建议退出账号重新登录后再测试）



应用基本信息

应用类型： 普通应用 - 客户端

应用名称： 客户端sdk测试使用 该名称也用于来源显示，不超过10个汉字或20个字母

应用平台： 手机 [查看移动客户端接入指南](#)

☒ iPhone ☐ Android ☐ BlackBerry ☐ Windows Phone ☐ Symbian ☐ WebOS ☐ Other

* Apple ID: [如何查找Apple ID ?](#)

* Bundle ID:

注：Apple ID 如果没有的话，先随意填写，当获取了合法的 Apple ID 之后请马上到这个页面修改为正式版本。而 Bundle ID 需要和工程设置保证一致，在 XCODE11 下 Bundle 的截图如下：

4. 设置工程回调 URL Scheme

需修改应用工程中 info.plist 文件 中的 URL types 项，设置该项是保证微博授权成功后能够打开您的应用，该项中 URL Schemes 为您应用被授权后 sso 的回调地址，格式为“WB[你的应用程序的 Appkey]”，例如:wb204543436852

▼ URL Types (1)

com.weibo

No image specified

Identifier: com.weibo

Icon: None

URL Schemes: wb2045436852

Role: Editor

▼ Additional url type properties (0)

| Key | Type | Value |
|--|------|-------|
| Click here to add additional url type properties | | |

添加微博 scheme URL 到白名单 <sinaweibo、weibosdk、weibosdk2.5、

| ▼ LSAApplicationQueriesSchemes | Array | (5 items) |
|--------------------------------|--------|-------------|
| Item 0 | String | sinaweibohd |
| Item 1 | String | sinaweibo |
| Item 2 | String | weibosdk |
| Item 3 | String | weibosdk2.5 |
| Item 4 | String | weibosdk3.3 |

weibosdk3.3>

更新 3.3.0 版本后或大于 3.3.0 版本必须添加 weibosdk3.3 到工程白名单里，不然 ios14 的适配 bug 还会存在

5.设置工程回调 universalLink

首先需要配置 universalLink，建议大家先看看苹果官方文档 [Support Universal Links](#)，配置完成后自己可以通过把 link 地址放入手机的备忘录里让后点击是否可以呼起自己的 app，也可以通过 3.3.0 版本 SDK 的代码进行校验是否配置成功。

```
+(void)checkUniversalLink:(void (^)(WBULCheckStep step, NSError *error))checkBlock
```

step 分为 8 步，WBULCheckStepFinal 状态是校验成功，其余都是 link 校验不成功

然后将配置好的 link 注册到微博开放平台自己的 app 下，让后初始化分享 sdk

时传入即可，注册的格式：<https://> 开头 / 结尾, 如：<https://applink.com/>

Associated Domains



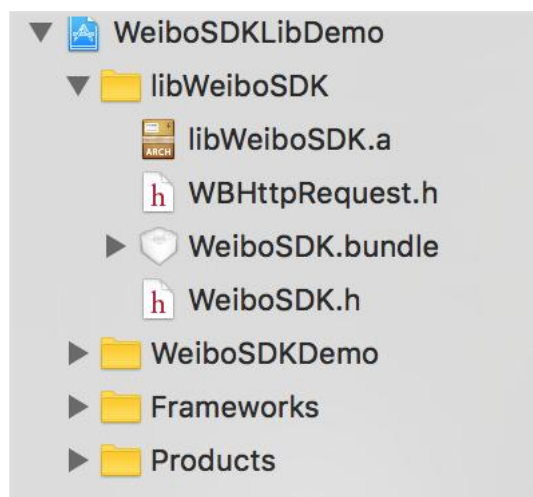
添加 universalLink 代理回调

```
- (BOOL)application:(UIApplication *)application
continueUserActivity:(NSUserActivity *)userActivity
restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> * __nullable
restorableObjects))restorationHandler{

    return [WeiboSDK handleOpenUniversalLink:userActivity delegate:self];
}
```

6. 添加 SDK 文件到工程

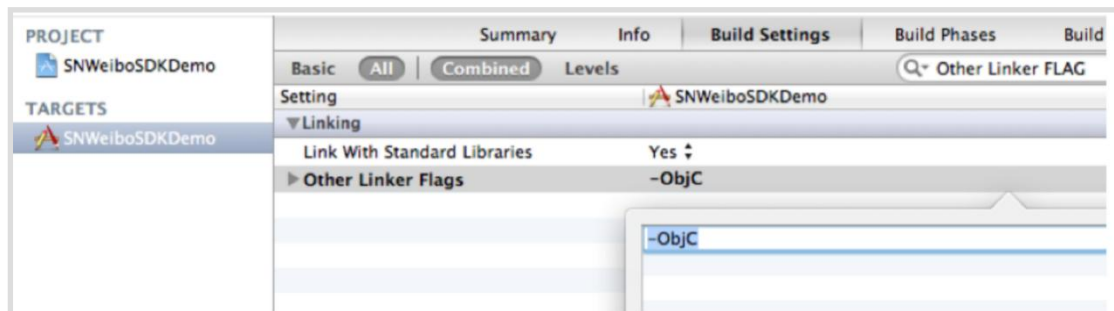
将从 GitHub 上下载的 libWeiboSDK 文件夹添加至工程，其中包含 WeiboSDK.h、WBHttpRequest.h 这两个 .h 文件以及 libWeiboSDK.a 和 WeiboSDK.bundle，统共 4 个文件。



7. 设置工程的 **objc** 编译选项

在工程中引入静态库之后，为避免静态库中类加载不全造成程序崩溃，需设置 Other Linker Flags，设置步骤如下：










程序 Target->Build Settings->Linking 下 Other Linker Flags 项添加 -ObjC。



8. 添加 **Framework** 文件到工程

在工程中修改 Other Linker Flags 后，需要修改编译步骤的链接库设置，添加微博 SDK 需要的链接库，避免链接阶段由于库的设置错误导致程序崩溃，添加链接库步骤如下：

程序 Target->Build Phases->Link Binary With Libraries 下添加以下 Framework 至工程中。集成微博 SDK 后需要添加的 Frameworks 为：
QuartzCore.framework、ImageIO.framework、
SystemConfiguration.framework、Security.framework、
CoreTelephony.framework、CoreText.framework、UIKit.framework、
Foundation.framework、CoreGraphics.framework、libz.dylib、
libsqlite3.dylib、WebKit.framework。

| Name | Embed |
|---|-------------------|
|  CoreGraphics.framework | Do Not Embed ⬆️⬆️ |
|  CoreTelephony.framework | Do Not Embed ⬆️⬆️ |
|  CoreText.framework | Do Not Embed ⬆️⬆️ |
|  Foundation.framework | Do Not Embed ⬆️⬆️ |
|  ImageIO.framework | Do Not Embed ⬆️⬆️ |
|  libsqlite3.dylib | Do Not Embed ⬆️⬆️ |
|  libWeiboSDK.a | |
|  libz.dylib | Do Not Embed ⬆️⬆️ |
|  Photos.framework | Do Not Embed ⬆️⬆️ |

9. 定义应用 SSO 登录及 OAuth2.0 认证所需常量

(1) AppKey:第三方应用申请的 appkey, 用来鉴证第三方应用的身份、显示来源等;

(2) AppRedirectURL:应用回调页,在进行 OAuth2.0 登录认证时所用。

对于 Mobile 客户端应用来说,是不存在 Server 的,故此处的应用回调页地址只要与新浪微博开放平台->我的应用->应用信息->高级应用->授权设置->应用回调页中的 url 地址保持一致就可以了,如图所示:

```
#define kAppKey          @"2045436852"  
#define kRedirectURI     @"http://www.sina.com"
```

10. 注册 appkey(clientid)和 universalLink

程序启动时,在代码中向微博终端注册你的 Appkey 和 universalLink,如果首次集成微博 SDK,建议打开调试选项以便输出调试信息。使用实例如图所示:

```
- (BOOL)application:(UIApplication *)application  
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{  
    [WeiboSDK enableDebugMode:YES];  
    [WeiboSDK registerApp:kAppKey  
        universalLink:@"https://applink.com/"];  
    return YES;  
}
```

11. 重写 AppDelegate 的 handleOpenURL 和 openURL 和 continueUserActivity 方法

handleOpenURL 和 openURL 和 continueUserActivity 处理微博客户端通过 URL 启动您的应用,通过这三个方法微博客户端可以传

递消息给您的应用。使用实例如图所示：

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    return [WeiboSDK handleOpenURL:url delegate:self];
}

- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url
{
    return [WeiboSDK handleOpenURL:url delegate:self];
}

- (BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity
    restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> * __nullable
    restorableObjects))restorationHandler{
    return [WeiboSDK handleOpenUniversalLink:userActivity delegate:self];
}
```

二、应用场景代码示例

1. SSO 微博客户端授权认证

使用微博 SDK 提供的 SSO 授权，微博会授权给第三方应用程序当前微博客户端已登录账户的相关信息，第三方应用程序可以使用微博账号信息处理自己相关逻辑，授权给用户登录状态。具体使用实例如图所示：

```
{
    WBAuthorizeRequest *request = [WBAuthorizeRequest request];
    request.redirectURI = kRedirectURI;
    request.scope = @"all";
    request.userInfo = @{@"SSO_From": @"SendMessageToWeiboViewController",
                        @"Other_Info_1": [NSNumber numberWithInt:123],
                        @"Other_Info_2": @{@"obj1", @"obj2"},
                        @"Other_Info_3": @{@"key1": @"obj1", @"key2": @"obj2"}};
    [WeiboSDK sendRequest:request];
}
```

图中 UserInfo 内容为用户自定义的内容，可不填写，微博回调 Response 中会通过 requestUserInfo 包含原 request.userInfo 中的所有数据，用于第三方自定义操作或 request 区分。调用 SendRequest 的方法后，会跳转到微博客户端，进行 SSO 授权。如果当前微博客户端没有账号登录，则跳转到微博后将进入微博提供的登录界面，用户登录成功后将跳转到您的应用，并传递用户的相关信息给您的应用；

如果当前微博客户端已经有账户，则进入账户管理界面，选择要向第三方授权的账户。

当授权完成后均会回调给第三方应用程序，第三方应用程序需要实现 WeiboSDKDelegate 的 didReceiveWeiboResponse 方式监听此次请求的 response。

2. 第三方应用向微博发送请求，完成分享

第三方向微博发送请求，利用 WBSendMessageToWeiboRequest 类完成该功能。

(1) message 对象，承载第三方应用发送给微博的消息。

(2) +requestWithMessage :

+requestWithMessage:authInfo:access_token:方法，

上述两个方法均返回一个 WBSendMessageToWeiboRequest 对象，生成 WBSendMessageToWeiboRequest 对象被 [WeiboSDK sendRequest:] 会唤起微博客户端的发布器进行分享，如果未安装微博客户端或客户端版本太低，会根据 shouldOpenWeiboAppInstallPageIfNotInstalled 属性判断是否弹出安装/更新微博的对话框。使用实例如图所示：

请求的 UserInfo 内容也为用户自定义内容（可已不填写），微博回调 Response 中

```
WBSendMessageToWeiboRequest *request = [WBSendMessageToWeiboRequest
    requestWithMessage:_messageObject authInfo:authRequest
    access_token:myDelegate.wbtoken];
request.userInfo = @{@"ShareMessageFrom": @"SendMessageToWeiboViewController",
    @"Other_Info_1": [NSNumber numberWithInt:123],
    @"Other_Info_2": @[@"obj1", @"obj2"],
    @"Other_Info_3": @{@"key1": @"obj1", @"key2": @"obj2"}};
[WeiboSDK sendRequest:request];
```

会通过 requestUserInfo 包含原 request.userInfo 中的所有数据，用于第三方自定义操作或 request 区分。

3. 微博调起第三方,向第三方请求数据

第三方应用需实现 WeiboSDKDelegate 的

didReceiveWeiboRequest 方法，当向微博提供数据时，需将数据封装成 WBMessageObject，需要使用 WBProvideMessageForWeiboResponse 对象完成该功能，使用实例如图所示：

```
- (WBMessageObject *)messageToShare
{
    WBMessageObject *message = [WBMessageObject message];
    message.text = @"测试通过WeiboSDK发送文字到微博!";
    return message;
}
```

利用微博 SDK，构造出 WBProvideMessageForWeiboResponse 对象，通过 sendResponse 方法回传数据给微博客户端，完成此次任务。使用实例如图所示：

```
WBProvideMessageForWeiboResponse *response =
    [WBProvideMessageForWeiboResponse responseWithMessage:[self
    messageToShare]];

if ([WeiboSDK sendResponse:response])
{
    [self dismissModalViewControllerAnimated:YES];
}
```

4. 用户取消对应用的授权

微博 SDK 提供用户取消应用授权功能，该功能对应的 api 为

```
+(void)logOutWithToken:(NSString *)token
delegate:(id<WBHttpRequestDelegate>)delegate
withTag:(NSString*)tag;
```

```
AppDelegate *myDelegate =(AppDelegate*)[[UIApplication sharedApplication] delegate];
[WeiboSDK logOutWithToken:myDelegate.wbtoken delegate:self withTag:@"user1"];
```

使用实例如图所示：

调用此接口后，微博 SDK 会发起网络请求使 token 失效。应用可实现 WBHttpRequestDelegate 中的

```
-(void)request:(WBHttpRequest*)request
```

```

didReceiveResponse:(NSURLResponse *)response;

- (void)request:(WBHttpRequest *)request didFailWithError:(NSError
*)error;

- (void)request:(WBHttpRequest *)request
didFinishLoadingWithResult:(NSString *)result;

```

方法用于监听此次 Http 请求，如：

```

- (void)request:(WBHttpRequest *)request didFinishLoadingWithResult:(NSString *)result
{
    NSString *title = nil;
    UIAlertView *alert = nil;

    title = @"收到网络回调";
    alert = [[UIAlertView alloc] initWithTitle:title
                                       message:[NSString stringWithFormat:@"%s",result]
                                       delegate:nil
                                       cancelButtonTitle:@"确定"
                                       otherButtonTitles:nil];

    [alert show];
    [alert release];
}

- (void)request:(WBHttpRequest *)request didFailWithError:(NSError *)error;
{
    NSString *title = nil;
    UIAlertView *alert = nil;

    title = @"请求异常";
    alert = [[UIAlertView alloc] initWithTitle:title
                                       message:[NSString stringWithFormat:@"%s",error]
                                       delegate:nil
                                       cancelButtonTitle:@"确定"
                                       otherButtonTitles:nil];

    [alert show];
    [alert release];
}

```

5. aid 相关 aid 是设备唯一的移动设备指纹

微博 SDK 提供 aid 作为设备的唯一标识，如果您的应用需要使用一个设备唯一标识符来区分设备，可以使用 aid 值。

使用方法见以下方法：

```
+ (NSString*)getWeiboAid;
```

方法声明在 WeiboSDK.h 中,说明如下：

- 1)该方法返回当前设备的 aid 值。返回的 aid 值可能为 nil,当值为 nil 时会尝试向服务器获取 aid 值。
- 2)当获取成功时(aid 值变为有效值)时,SDK 会发出名为 WeiboSDKGetAidSucessNotification 的通知,通知中带有 aid 值。
- 3)当获取失败时,SDK 会发出名为 WeiboSDKGetAidFailNotification

的通知,通知中带有 NSError 对象。

6.SDK 连接到微博

在 SDK3.2 版本之后,微博 SDK 添加了跳转到微博不同页面的功能,即呼起微博客户端并跳转到对应的页面或打开微博 H5 页面。具体逻辑及使用实例如下。

微博 SDK 会自动检测是否安装微博客户端,当调用 SDK 相关方法时:页面,用户登录后跳转到对应的页面。

```
linkToUserButton = [self setupButtonWithTitle:@"连接到指定用户的微博个人主页" andSelector:
    @selector(linkToUser:)];
linkToSingleBlogButton = [self setupButtonWithTitle:@"连接到指定的单条微博详情页" andSelector:
    @selector(linkToSingleBlog:)];
linkToArticleButton = [self setupButtonWithTitle:@"连接到指定的微博头条文章页" andSelector:
    @selector(linkToArticle:)];
shareToWeiboButton = [self setupButtonWithTitle:@"分享到微博" andSelector:@selector
    (shareToWeibo:)];
commentToWeiboButton = [self setupButtonWithTitle:@"评论指定的微博" andSelector:@selector
    (commentToWeibo:)];
linkToSearchButton = [self setupButtonWithTitle:@"连接到微博搜索内容流" andSelector:@selector
    (linkToSearch:)];
linkToTimelineButton = [self setupButtonWithTitle:@"连接到我的微博消息流" andSelector:
    @selector(linkToTimeline:)];
linkToProfileButton = [self setupButtonWithTitle:@"连接到我的微博个人主页" andSelector:
    @selector(linkToProfile:)];
```

注:以上场景均可在 Demo 源码中找到相应代码片段

7. SDK 分享多图、视频到微博

WBMessageObject 对象的 imageObject 属性完成分享多图到微博。将分享到微博的图片使用 -imageObject 的 -addImages: 方法。(ps:旧的图片数据还留着,两种都有的话,新版微博优先识别新的数据)。使用实例如图所示:

WBMessageObject 对象的 videoObject 属性完成分享视频到微博。

```
UIImage *image = [UIImage imageNamed:@"image_1.jpg"];
UIImage *image1 = [UIImage imageNamed:@"image_2.jpg"];
NSArray *imageArray = [NSArray arrayWithObjects:image,image1, nil];
WBImageObject *imageObject = [WBImageObject object];
if (self.storySwitch.on) {
    imageObject.isShareToStory = YES;
    imageArray = [NSArray arrayWithObject:image];
}
imageObject.delegate = self;

[imageObject addImages:imageArray];
message.imageObject = imageObject;
```


使用 -addVideo 的 -addImages: 方法将视频地址。使用实例如图所示：

```
WBNewVideoObject *videoObject = [WBNewVideoObject object];
if (self.storySwitch.on) {
    videoObject.isShareToStory = YES;
}
NSURL *videoUrl = [NSURL URLWithString:[NSBundle mainBundle] pathForResource:@"apm"
ofType:@"mov"]];
videoObject.delegate = self;
[videoObject addVideo:videoUrl];
message.videoObject = videoObject;
```

该功能实现使用相册作为载体，存在异步通信，故第三方开发者需要实现 WBMediaTransferProtocol 这个协议，并且在 SDK 写入图片或视频时给用户等待的提示（加载界面），协议目前只有数据准备成功和失败两种回调：

```
typedef NSInteger WBSDKMediaTransferErrorCode
{
    WBSDKMediaTransferAlbumPermissionError = 0, //相册权限
    WBSDKMediaTransferAlbumWriteError = 1, //相册写入错误
    WBSDKMediaTransferAlbumAssetTypeError = 2, //资源类型错误
};

/**
 图片视频分享协议
 */
@protocol WBMediaTransferProtocol <NSObject>

/**
 数据准备成功回调
 */
-(void)wbsdk_TransferDidReceiveObject:(id)object;

/**
 数据准备失败回调
 */
-(void)wbsdk_TransferDidFailWithErrorCode:(WBSDKMediaTransferErrorCode)errorCode
andError:(NSError*)error;
```

8. SDK 提供分享单图、视频到 story

WBNewVideoObject 和 WBImageObject 都有一个 isShareToStory 的属性，设为 YES 后表示分享到 story。具体使用实例可参考本文档第 7 部分，SDK 分享多图、视频到微博。

图片，多媒体，视频对象两两不能共存，文字和多媒体对象无

法分享到 story，如果和图片或视频分享到 story 时会被丢弃，没有图片或者视频，默认会呼起发布者分享。

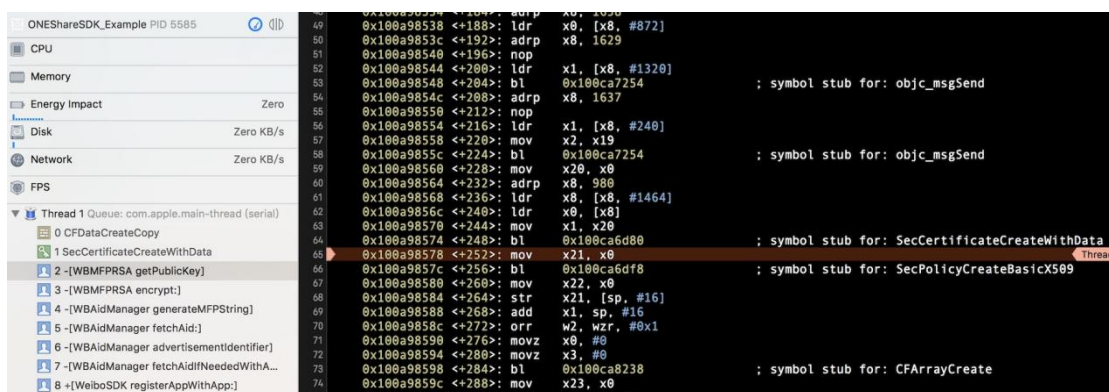
三、常见问题

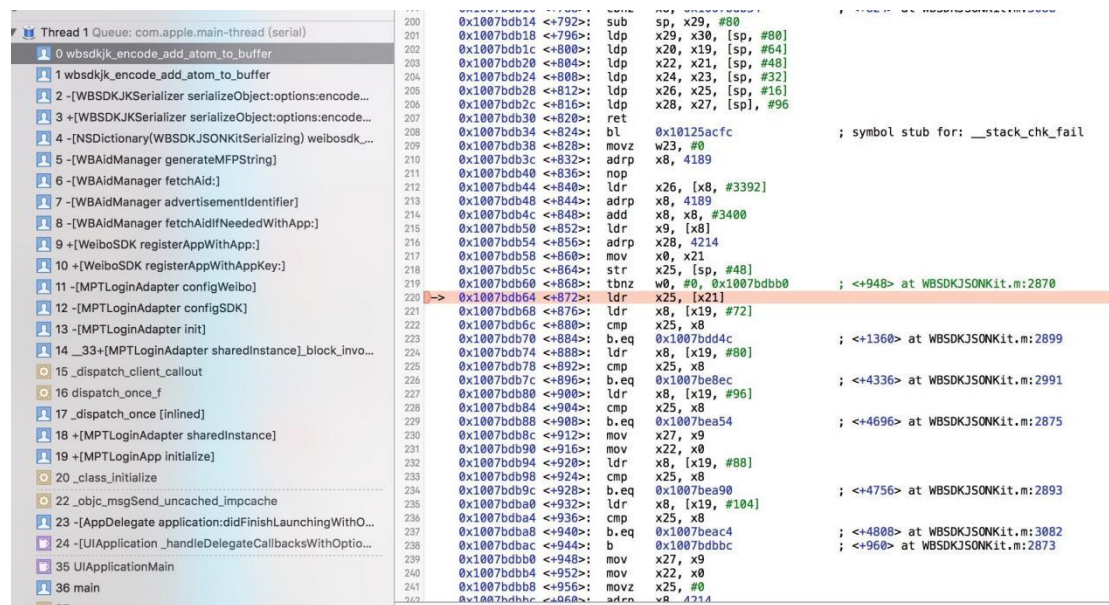
1. registerAppWithApp 使用 UIPastboard 导致主线程卡死问题

- 1) 原因：应该是苹果的坑
- 2) 解决方法：可以考虑将该方法放在子线程中执行
- 3) 参考 issue：[#412](#)，[#371](#)

2. 微博 SDK 需要放在 main bundle

- 1) 原因：由于微博 SDK 需要获取第三方 app 中 main bundle 里面的信息，保证微博 sdk 在 main bundle 中。
- 2) 可能造成问题：bundle 位置不对可能会造成崩溃，崩溃的位置都在[WBMFPRSA getPublicKey]、[WBMFPRSA encrypt:]、[WBMFPRSA generateMFPString]，如下所示



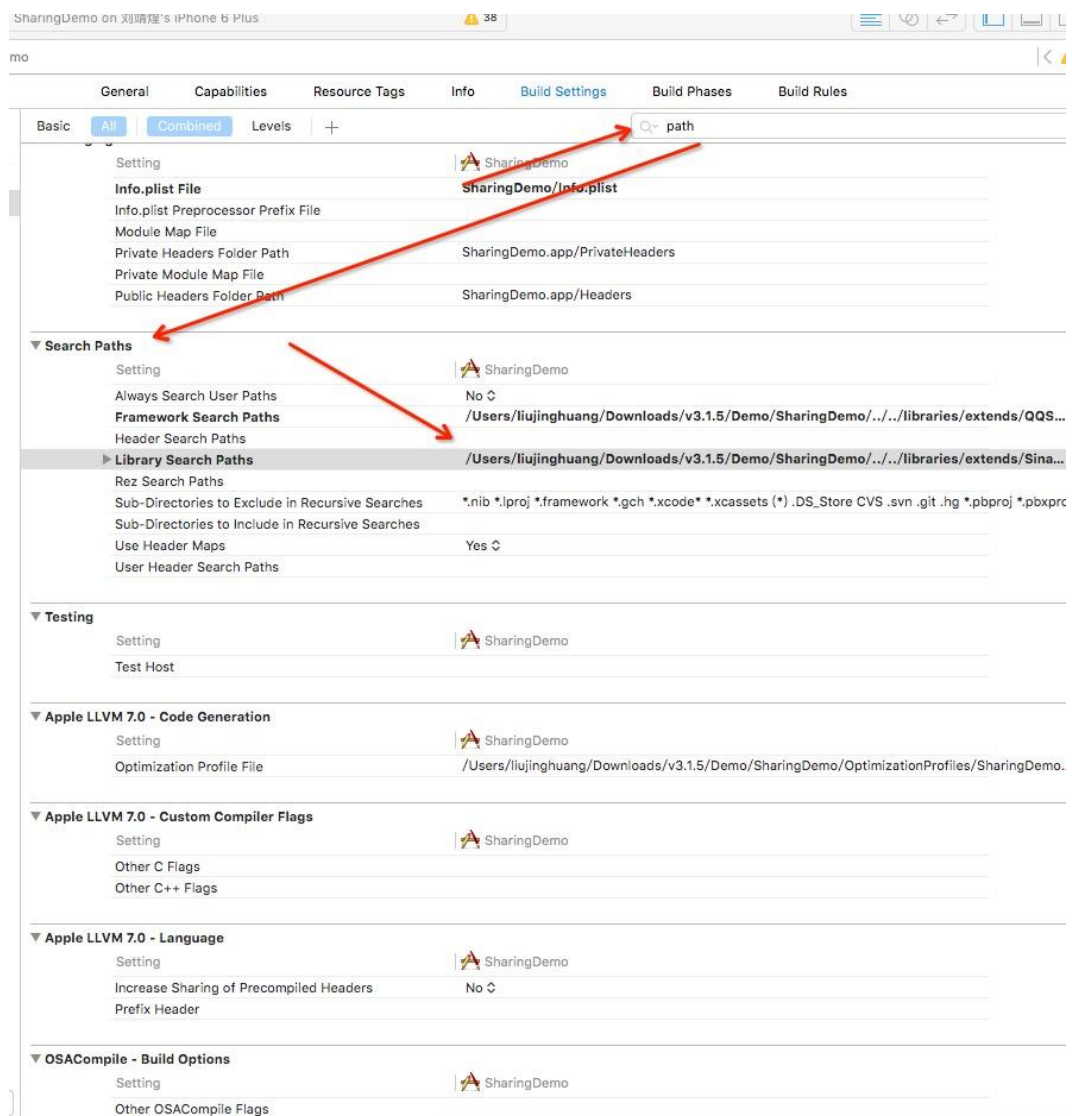


3) 解决方法：微博 SDK 在 main bundle 里面，微博中的 resource bundle 也应添加到 main resources 中，不应封装为动态库

4) 参考 issue：[#74](#), [#117](#), [#129](#), [#133](#)

3. JSONKit 报错问题

1) 原因：path 问题



2) 解决方法：如下所示

4. 微博 SDK 授权后为什么没有返回应用

1) 原因：可能是配置问题

2) 解决方法：检查配置,info.plist 里设置的 Scheme 是否正确，需要是 wb+appkey 的形式