

# iOS 平台 SDK 文档

编号: WJOAUTH\_IOS\_SDK

版本: WJOAUTH \_IOS\_SDK V1.0.0

修订记录:

时间	文档版本	修订人	备注
2018/05/14	1.0.0	周美丽	初稿

# 目录

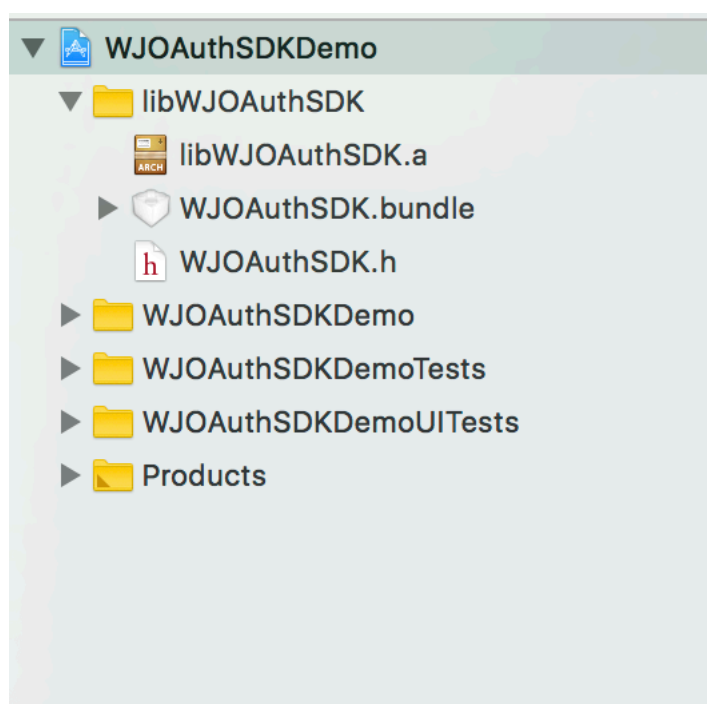
一、SDK 接入设置 .....	3
1、添加 SDK 文件到工程 .....	3
2、在工程中引入静态库之后，需要在编译时添加-objc 编译选项 .....	3
3、添加 Framework 文件到工程 .....	4
4、定义应用 OAuth2.0 认证所需的几个常量 .....	4
5、注册 appkey(clientid) .....	4
二、应用场景代码示例 .....	4
1、oauth2.0 授权认证 .....	4
2、从第三方应用向 WJOAuthSDK 发送请求 .....	5
3、第三方应用收到 WJOAuthSDK 回调 .....	5

# WJOAuth SDK

## 一、SDK 接入设置

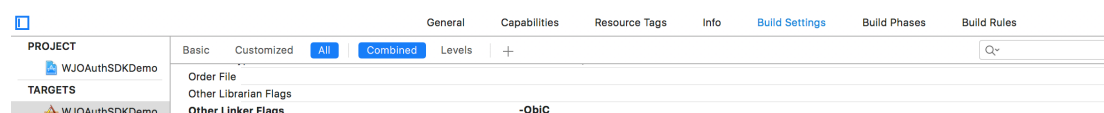
### 1、添加 SDK 文件到工程

将从 GitHub 上下载的 libWJOAuthSDK 文件夹添加至工程，其中包含 WJOAuthSDK.h 文件以及 libWJOAuthSDK.a 和 WJOAuthSDK.bundle，统共 3 个文件。



### 2、在工程中引入静态库之后，需要在编译时添加-objc 编译选项

避免静态库中类加载不全造成程序崩溃。方法：程序 Target->Build Settings->Linking 下 Other Linker Flags 项添加-ObjC。



### 3、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，避免链接阶段由于库的设置错误导致程序崩溃。方法：程序 Target->Build Phases->Link Binary With Libraries 下添加以下 Framework 至工程中。需要添加的 Frameworks 为：SystemConfiguration.framework。

### 4、定义应用 OAuth2.0 认证所需的几个常量

AppKey:第三方应用申请的 appkey, 用来身份鉴证、显示来源等; 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应用回调页中的 url 地址保持一致就可以了, 如图所示:

```
#define kAppKey          @"3573952935"
#define kRedirectURI     @"http://your_callback_uri"
```

### 5、注册 appkey(clientid)

程序启动时, 在代码中向 WJOAuthSDK 注册你的 Appkey, 可设置授权界面语言, 默认跟随系统。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    [WJOAuthSDK setLanguageType:WJLanguageTypeChinese];

    [WJOAuthSDK registerApp:kAppKey];

    return YES;
}
```

## 二、应用场景代码示例

### 1、oauth2.0 授权认证

调用[WJOAuthSDK sendRequest:delegate:]的方法后会跳转到 WJOAuthSDK, 进入登录界面进行授权, 当授权完成后会回调给第三方应用程序, 第三方实现 WJOAuthSDKDelegate 的相关方法监听此次请求的 response. 此中 state 内容为 用户自定义(可不填写), WJOAuthSDK 回调 Response 中会通过 requestState 包含原 request.state 中的所有数据, 用于数据传输过程中校验相关的上下文环境数据。

## 2、从第三方应用向 WJOAuthSDK 发送请求

代码示例如：

```
WJOAuthRequest *request = [WJOAuthRequest request];
request.state = @"Verification";
request.redirectURI = kRedirectURI;
[WJOAuthSDK sendRequest:request delegate:self];
```

同上此中 state 内容为用户自定义(可不填写)，WJOAuthSDK 回调 Response 中会通过 requestState 包含原 request.state 中的所有数据，用于数据传输过程中校验相关的上下文环境数据。

## 3、第三方应用收到 WJOAuthSDK 回调

代码示例如：

```
- (void)didReceiveWJOAuthSucceededResponse:(WJBaseResponse *)response {
    WJOAuthSucceededResponse *authorizeResponse = (WJOAuthSucceededResponse *)response;
}
```

WJOAuthSDK 回调 Response 中包含 accessToken、requestState。accessToken 为访问相关信息的认证口令； requestState 包含原 request.state 中的所有数据，用于数据传输过程中校验相关的上下文环境数据。

```
- (void)didReceiveWJOAuthFailedResponse:(WJBaseResponse *)response {
    WJOAuthFailedResponse *authorizeResponse = (WJOAuthFailedResponse *)response;
}
```

WJOAuthSDK 回调 Response 中包含 errorCode、errorCodeDescription。授权失败时 WJOAuthSDK 会通过 errorCode 返回相关响应状态码，通过 errorCodeDescription 返回状态码描述；具体状态码说明，请访问开发文档 -> OAuth2.0 -> 开发相关资源 -> 返回状态码说明。

```
- (void)didReceiveWJOAuthClickedCancelResponse {
}
```

当用户点击关闭授权页面时，将收到 WJOAuthSDK 回调

具体调用示例参见demo