# 游戏SDK（iOS）接入手册

[游戏海外发行游戏SDK（iOS）接入手册 1](#_Toc498003403)

[版本控制页 2](#_Toc498003404)

[一、SDK简介 3](#_Toc498003405)

[1、概述 3](#_Toc498003406)

2、SDK配置版本要求-------------------------------------------------------------------3

[3、SDK构成 3](#_Toc498003408)

二、SDK开发环境配置---------------------------------------------------------------------4

1、Xcode工程配置-----------------------------------------------------------------------4

[2、添加游戏SDK及资源库 4](#_Toc498003411)

[3、添加Framework 5](#_Toc498003412)

[4、配置相册，相机访问权限 5](#_Toc498003413)

[5、facebook相关的配置 6](#_Toc498003414)

[三、SDK接入示例 7](#_Toc498003415)

[1、在使用的位位置导入头文件 7](#_Toc498003416)

[2、添加SDK想着通知观察者（需要为下表所有的通知添加观察者，并实现处理通知的方法） 7](#_Toc498003417)

[3、移除SDK相关通知观察者 9](#_Toc498003418)

[4、接口和通知流程图（放大查看） 10](#_Toc498003419)

[5、初始化接口（添加通知观察者后，接入SDK最先调用的方法） 10](#_Toc498003420)

[6、登录接口（接收到初始化成功通知后调用） 11](#_Toc498003421)

[8、登出接口（切换账号） 13](#_Toc498003422)

[9、支付接口 13](#_Toc498003423)

[10、facebook分享 15](#_Toc498003424)

11、FACEBOOK好友获取----------------------------------------------------------------------------16

[12、进入游戏日志 17](#_Toc498003426)

[13、角色创建接口 18](#_Toc498003427)

[14、退出接口（选接） 19](#_Toc498003428)

[15、用户中心接口 19](#_Toc498003429)

[16、设置SDK语言 19](#_Toc498003430)

[17、删除未完成的测试订单接口（测试环境下可用） 20](#_Toc498003431)

[四、应用生命周期调用 20](#_Toc498003432)

版本控制页

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **发布时间** | **内容** | **作者** |
| 1.0.0 | 2020-04-07 | 初始化、登录、注册、支付、Facebook好友接口 | 刘康逸 |

# SDK简介

# 概述

# 本手册是研发IOS接入游戏海外发行用户体系，以及支付的使用指南（开发语言：Objective-C）。主要内容包括登录、注册、绑定、支付接口以及接口调用的方法和参数说明。

## 配置要求

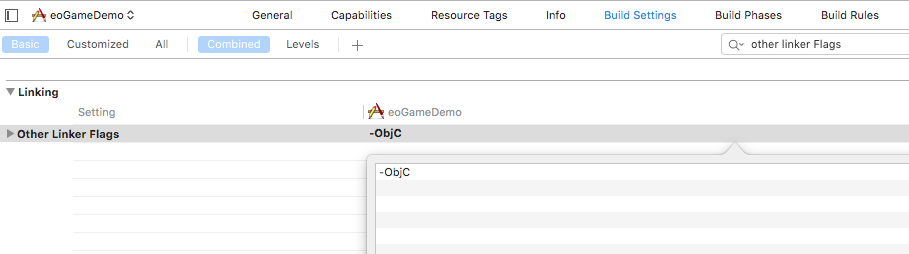
支持iOS8.0及以上版本。

## SDK构成

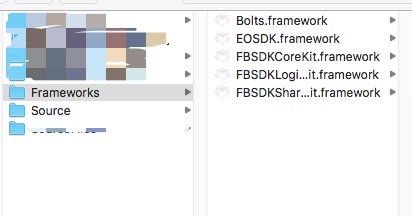
|  |  |
| --- | --- |
| SandClass.framework | 静态库文件 |
| SandClass.bundle | 资源文件 |
| Bolts.framework  FBSDKCoreKit.framework  FBSDKLoginKit.framework  FBSDKShareKit.framework | Facebook 相关的类库 |

1. SDK开发环境配置
2. Xcode工程配置

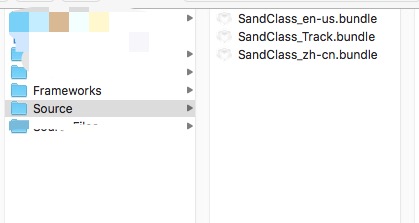
打开你的工程配置，Xcode ->TARGETS -> Build Settings，搜索other Linker Flags，添加：-ObjC



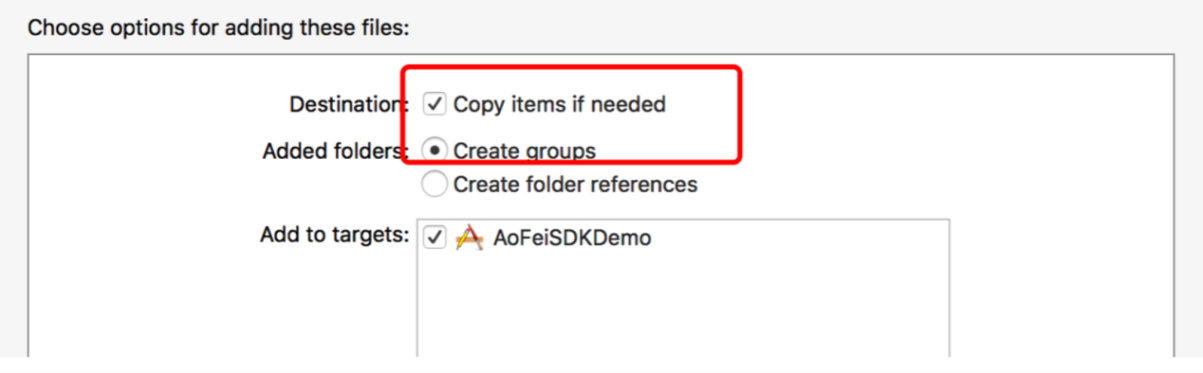
1. 添加游戏资源库
2. 将Frameworks文件夹拷贝到编译项目下



1. 将Source资源文件夹拖到项目中



（3）勾选“Copy items if needed”，选择“Create groups” 的添加方式

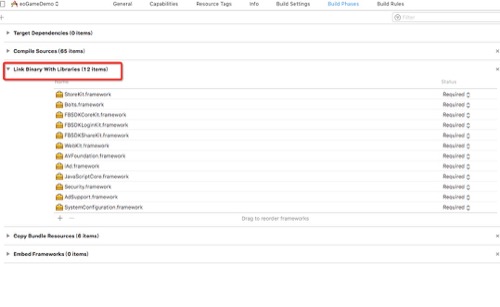


1. 添加Framework

Framework 添加方式：Target -> Build Phrases

|  |
| --- |
| **AdSupport.framework** |
| StoreKit.framework |
| Security.framework |
| SystemConfiguration.framework |
| JavaScriptCore.framework |

图示：

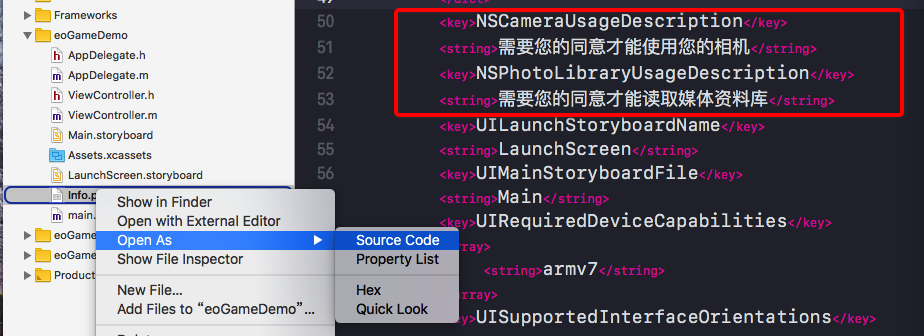


1. 配置相册，相机访问权限

添加方式：Target -> Info.plist 中添加如下字段

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **值（提示语，可自定义）** |
| Privacy - Photo Library Usage Description | String | 需要您的同意才能读取媒体资料库 |
| Privacy - Camera Usage Description | String | 需要您的同意才能使 您的相机 |

另一种添加方式如图：



1. Facebook相关配置

（1）在Info.plist中添加如下字段（添加在<dict>...</dict>中）

需要将{facebookAppid} 和 {facebookDisplayName} 替换成在facebook官网注册的Appid和facebookDisplayName.

<key>CFBundleURLTypes</key>

<array>

<dict> <key>CFBundleURLSchemes</key>

<array> <string> fb{facebookAppId} </string> </array>

</dict>

</array>

<key>FacebookAppID</key> <string> {facebookAppId} </string>

<key>FacebookDisplayName</key>

<string> {facebookDisplayName} </string>

图示：



1. SDK接入示例
2. 在使用的位置导入头文件

#import "SandClass.h"

1. 添加SDK相关的通知观察者

调用SDK的接口后，SDK会发送一些通知返回SDK接口调用的结果，接入方需在合适的地方，先添加SDK相关通知观察者并实现处理对应的通知的方法。

|  |  |  |  |
| --- | --- | --- | --- |
| 通知名称 | 触发时间 | **返回数据类型** | 详细字段 |
| kEOUnInitNotification | 初始化完成后 | 无 | 无 |
| kEOUnInitNotification | 当未初始化却调用了”登录”方法时 | 无 | 无 |
| kEOLoginSuccessNotification | 当登录成功 | NSDictionanry类型 | tk //token |
| kEOLoginErrorNotification | 当登录发生错误 | NSString类型 | 登录失败原因 |
| kEOLogoutSuccessNotification | 调用”登出接口”，或者是用户中心中的切换帐号会触发，必须处理该通知 | 无 | 无 |
| kEOUnLoginNotification | 当未登录却调用了”支付”接口 | 无 | 无 |
| kEOOrderErrorNotification | 当支付失败 | NSString类型 | 支付错误原因 |
| kEOOrderSuccessNotification | 当支付成功 | SGPayInfo类型 | 支付信息 |
| kEOFacebookShareFailNotification | 分享失败 | 无 | 无 |
| kEOFacebookShareSuccessNotification | 分享成功 | 无 | 无 |
| kEOFacebookFriendsFailNotification | 获取好友失败 | NSString | 错误信息 |
| kEOFacebookFriendsSuccessNotification | 获取好友成功 | NSAray | 获取的好友数组 |

示例：

#pragma mark -- 添加观察者

- (void)addObserver

{

    //初始化完成相关通知

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOInitCompletion) name:kEOInitCompletionNotification object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOUnInit) name:kEOUnInitNotification object:nil];

    //登录相关通知

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOLoginFail:) name:kEOLoginErrorNotification object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOLoginSuccess:) name:kEOLoginSuccessNotification object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOUnLogin) name:kEOUnLoginNotification object:nil];

    //

    //    //登出相关通知

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOLogOutSuccess) name:kEOLogoutSuccessNotification object:nil];

    //

    //支付相关通知

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOOrderFail:) name:kEOOrderErrorNotification object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOOrderCompletion:) name:kEOOrderSuccessNotification object:nil];

    //facebook分享相关通知

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOFacebookShareSuccess) name:kEOFacebookShareSuccessNotification object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOFacebookShareFail) name:kEOFacebookShareFailNotification object:nil];

    //facebook好友相关通知

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOFacebookFriendSuccess:) name:kEOGetFacebookFriendsSuccessNotification object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(EOFacebookFriendFail:) name:kEOGetFacebookFriendsFailNotification object:nil];

}

1. 移除SDK相关通知观察者

#pragma mark -- 移除观察者

- (void)dealloc {

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOInitCompletionNotification  object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOUnInitNotification object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOLoginSuccessNotification object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOLoginErrorNotification   object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOUnLoginNotification object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOLogoutSuccessNotification   object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOOrderSuccessNotification   object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOOrderErrorNotification     object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOFacebookShareFailNotification   object:self];

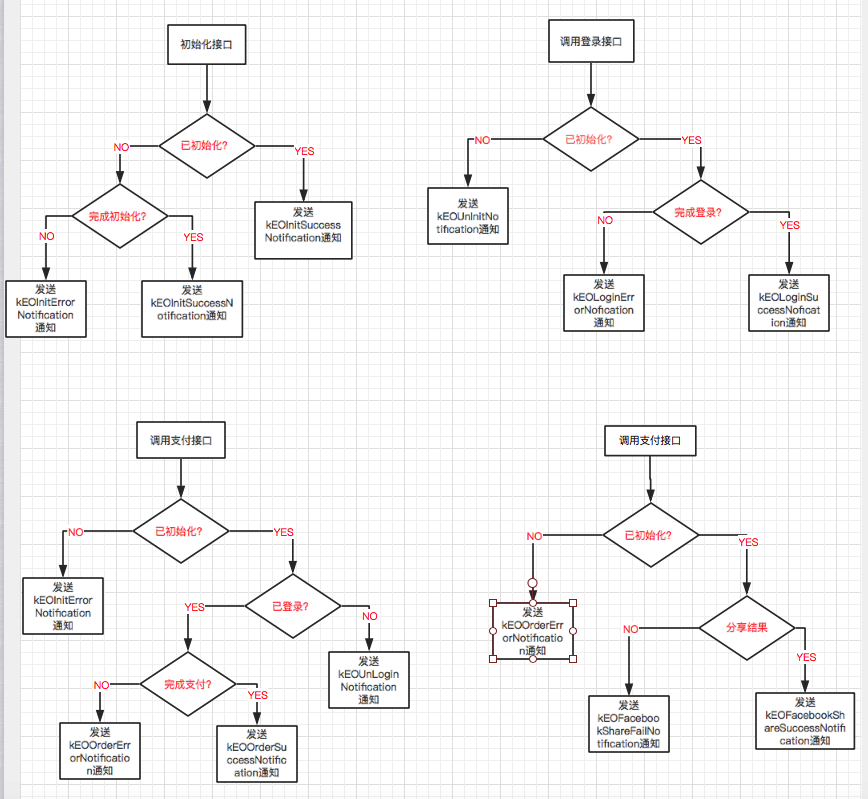
    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOFacebookShareSuccessNotification     object:self];

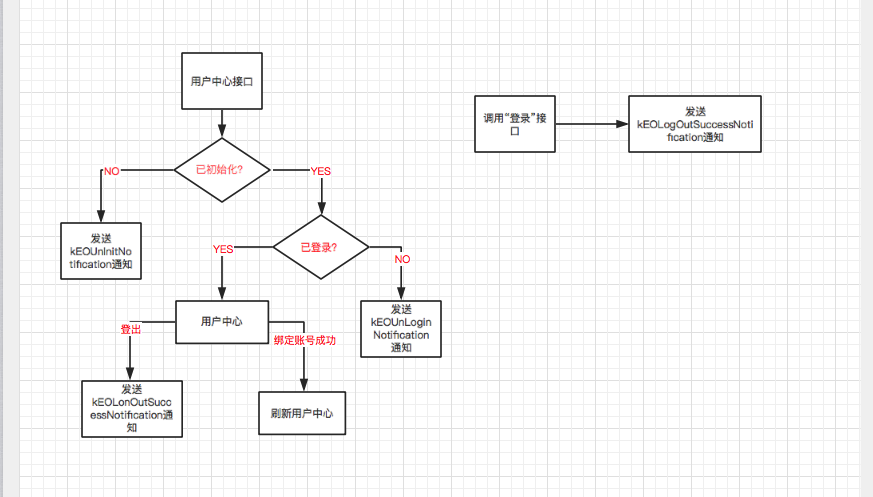
    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOGetFacebookFriendsSuccessNotification   object:self];

    [[NSNotificationCenter defaultCenter] removeObserver:self name:kEOGetFacebookFriendsFailNotification     object:self];

}

1. 接口和通知流程图





1. 初始化接口

PS：必须在初始化游戏主 window 之后再调用该方法初始化SDK

|  |  |
| --- | --- |
| **功能描述** | **设置SDK初始化参数** |
| 功能接口 | +(void)initWithConfig:(EOInitConfig \*)config; |
| 相关通知 | kEOInitCompletionNotification //初始化完成通知 |
| 参数字段 | 参数说明 |
| gameCode | 游戏海外发行提供 |
| gameKey | 游戏海外发行提供 |
| gameName | 游戏海外发行提供 |
| gameLanguage | 游戏海外发行提供 |
| debugMode | 正式环境时请填NO，测试环境填写YES |

示例：

EOInitConfig \*config = [SGInitConfig new];

config.gameCode = @"gameCode";//游戏海外发行提供

config.gameKey = @"gameKey";//游戏海外发行提供

config.gameName = @"gameName";//游戏海外发行提供

config.gameLanguage = @"zh-tw";//游戏海外发行提供

config.debugMode = YES; //正式环境时请填NO

**实现处理初始化相关的通知方法**

#pragma mark 处理"初始化完成通知"

- (void)EOInitCompletion {

#pragma mark 在接收到"初始化完成通知"后调用login接口

    NSLog(@"初始化完成");

}

1. 登录接口（接收到初始化成功的通知后调用）

|  |  |
| --- | --- |
| **功能描述** | **登录** |
| 功能接口 | + (void)login; |
| 使用示例 | [SandClass login]; |
| 相关通知 | kEOUnInitNotification //未始化完成通知 |
| kEOLoginSuccessNotification //登录成功通知 |
| kEOLoginErrorNotification //登录失败通知 |

实现处理 **登录相关通知** 的方法

#pragma mark -- 实现处理登陆通知的方法

- (void)EOLoginSuccess:(NSNotification \*)notify {

/\*

返回数据类型为NSDictionary包含以下字段: tk 登录token

\*/

NSDictionary \*resultDic = notify.object;

NSString \*token = [resultDic valueForKey:@"tk"];

NSLog(@"token = %@",token);

//TODO: 请将token发送到服务端验证登录的token是否正确

}

- (void)EOUnInit {

    //无返回数据

    NSLog(@"未初始化");

}

- (void)EOLoginFail:(NSNotification \*)notify

{

    //返回具体的登录失败原因,NSString类型

    NSString \*errorMsg = notify.object;

    NSLog(@"%@",[NSString stringWithFormat:@"登录失败 失败原因:%@",errorMsg]);

}

1. 绑定接口
2. 登出接口（切换账号）

|  |  |
| --- | --- |
| **功能描述** | **登出接口** |
| 功能接口 | + (void)logOut |
| 使用示例 | [SandClass logOut]; |
| 相关通知 | kEOLogoutSuccessNotification//初始化完成通知 |

实现处理 登出相关通知 的方法（必须）:

#pragma mark -- 实现处理登出通知的方法

- (void)EOLogOutSuccess

{

NSLog(@"登出成功");

//若在游戏中，请做好登出时，角色的数据的清理，并转跳到登录界面。

}

备注：切换账号的实现，可以先调用 登出接口 再 调用登录接口，实现切换账号

1. 支付接口

## 9、支付接口

|  |  |
| --- | --- |
| **功能描述** | **调用苹果商店内购** |
| 功能接口 | + (void)payWithRoleInfo:(SGRoleInfo \*)roleInfo payInfo:(EOPayInfo \*)payInfo; |
| 相关通知 | kEOOrderErrorNotification//支付失败通知 |
| kEOOrderSuccessNotification//支付成功通知 |
| 调用示例 | [SandClass payWithRoleInfo:roleInfo payInfo:payInfo]; |
| 参数字段 | 参数说明 |
| roleId | 角色id |
| roleName | 角色名 |
| roleLevel | 角色等级 |
| serverId | 服务器ID |
| gameOrderNo | 游戏支付订单号，数字字母或下划线组成，不要包含特  殊字符或空格，或传输需编码的字符 |
| productId | 商品id，苹果支付时需要用到。 |
| produceName | 商品名称 |
| price | 商品价格，**单位分**，例如：4.99美元的道具，传入@499 |
| produceExt | 透传参数。 |

示例：

 EORoleInfo\* roleInfo  = [EORoleInfo new];

    roleInfo.roleId       = @"1";

    roleInfo.roleName     = roleName;

    roleInfo.roleLevel    = @99;

    roleInfo.serverId     = @"emdxctest";

    EOPayInfo \*payInfo    = [EOPayInfo new];

    payInfo.gameOrderNO   = gameOrderNO;                   //游戏支付订单号，数字字母或下划线组成，不要包含特殊字符，空格，或传输需编码的字符

    payInfo.productId     = @"com.zwzj.pay8";       //商品id，苹果支付支付需要用到.

    payInfo.productName   = @"6元";                  //商品名称，某些渠道下会显示给用户

    payInfo.price         = @4999;

//    payInfo.productId    = @"com.iOS.yuanbao1000";

//    payInfo.productName    = @"com.iOS.yuanbao1000";

//    payInfo.price         = @1000;                         //商品价格,单位是分(整数) 例如:6元的道具 传入@600

    UInt64 recordTime     = [[NSDate date] timeIntervalSince1970] \* 1000;

    NSString \*time        = [NSString stringWithFormat:@"%llu",recordTime];

    payInfo.productExt    = [NSString stringWithFormat:@"{test: %@}",time];//扩展字段，建议使用json字符串

    [SandClass payWithRoleInfo:roleInfo payInfo:payInfo];

实现处理支付相关的通知方法：

- (void)EOOrderCompletion:(NSNotification \*)notify {

    /\*

     返回SGPayInfo类型数据，和发起支付接口传入的payInfo数据一致

     以下为SGPayInfo包含的字段

     \*/

    EOPayInfo \*payInfo = (EOPayInfo \*)notify.object;

    NSLog(@"订单成功 userInfo = %@",notify.userInfo);

    NSString \*gameOrderNO = payInfo.gameOrderNO;        //CP订单号

    NSString \*productId = payInfo.productId;            //内购商品ID

    NSString \*productName = payInfo.productName;        //内购商品名

    NSString \*productDesc = payInfo.productDesc;        //内购商品描述

NSString \*productExt = payInfo.productExt;          //扩展字段

    NSString \*sdkExt = payInfo.sdkExt;                  //SDK扩展字段

    NSUInteger price = [payInfo.price integerValue];    //订单价格，单位为分

}

1. Facebook 分享

|  |  |
| --- | --- |
| **功能描述** | **分享一个条信息到Facebook的TimeLine** |
| 功能接口 | +(void)EOFacebookShare☹UIViewController \*)viewController; |
| 调用示例 | [SandClass EOFacebookShare:self]; |
| 相关通知 | kEOFacebookShareSuccessNotification//分享成功通知 |
| kEOFacebookShareFailNotification //分享失败通知 |
| 参数字段 | 参数说明 |
| viewController | 游戏界面的viewController |

#pragma mark -- 实现处理订单相关通知的方法

- (void)EOOrderFail:(NSNotification \*)notify {

//返回订单失败原因，NSString类型

NSString \*errorMsg = notify.object;

NSLog(@"订单失败 失败原因notify.object = %@",errorMsg);

if(notify.userInfo)

{

NSLog(@"订单失败 失败原因userInfo = %@",notify.userInfo);

}

}

**实现 分享相关 的通知方法：**

#pragma mark facebook

- (void) EOFacebookShareFail

{

NSLog(@"facebook分享失败");

}

- (void) EOFacebookShareSuccess

{

NSLog(@"facebook分享成功");

}

1. Facebook 好友获取

|  |  |
| --- | --- |
| **功能描述** | **分享一个条信息到Facebook的TimeLine** |
| 功能接口 | +(void)EOGetFacebookFriend:(UIView \*)view; |
| 调用示例 | [SandClass EOGetFacebookFriend:self]; |
| 相关通知 | kEOFacebookFriendsSuccessNotification//获取好友成功通知 |
| kEOFacebookFriendsFailNotification //获取好友失败通知 |
| 参数字段 | 参数说明 |
| view | 游戏界面的view |

实现facebook 好友获取 相关的方法

|  |  |
| --- | --- |
| **功能描述** | **分享一个条信息到Facebook的TimeLine** |
| 功能接口 | +(void)EOGetFacebookFriend:(UIView \*)view; |
| 调用示例 | [SandClass EOGetFacebookFriend:self]; |
| 相关通知 | kEOFacebookFriendsSuccessNotification//获取好友成功通知 |
| kEOFacebookFriendsFailNotification //获取好友失败通知 |
| 参数字段 | 参数说明 |
| view | 游戏界面的view |

1. 进入游戏日志

- (void) EOFacebookFriendSuccess:(NSNotification \*)notify

{

NSLog(@"获取facebook好友成功");

NSLog(@"friends = %@",notify.object); // 返回一个用户数组,包括了当前玩家的信息，格式： ({ln = "xxx";picture="https://xxx";uid = xx},{...},...) ln为用户昵称，uid为用户userID，picture为头像地址。

NSArray\* friends = (NSArray\*)notify.object;

NSDictionary\* friendInfo = (NSDictionary\*)friends[0];

NSLog(@"uid = %@ , ln = %@, img = %@",[friendInfo valueForKey:@"uid"],[friendInfo valueForKey:@"ln"],[friendInfo valueForKey:@"picture"]);

}

|  |  |
| --- | --- |
| **功能描述** | **发送进入游戏日志** |
| 功能接口 | + (void)eoEntryGame☹EORoleInfo\*) roleInfo; |
| 调用示例 | [SandClass EOEntryGame:roleInfo]; |
| 参数字段 | 参数说明 |
| roleId | 角色id |
| roleName | 角色名 |
| roleLevel | 角色等级 |
| serverId | 服务器ID |
| serverName | 服务器名称 |

示例：

EORoleInfo\* roleInfo = [EORoleInfo new];

roleInfo.roleId = @"game\_id\_000001";

roleInfo.roleName = @"roleName";

roleInfo.roleLevel = @"16";

roleInfo.serverId = @"001";

roleInfo.serverName = @"测试服01";

[SandClass EOEntryGame:roleInfo];

1. 角色创建接口

|  |  |
| --- | --- |
| **功能描述** | **发送角色创建日志** |
| 功能接口 | + (void)EOEntryGame:(EORoleInfo\*) roleInfo; |
| 调用示例 | [SandClass EOEntryGame:roleInfo]; |
| 参数字段 | 参数说明 |
| roleId | 角色id |
| roleName | 角色名 |
| roleLevel | 角色水平 |
| serverId | 服务器ID |
| serverName | 服务器名称 |

示例：

EORoleInfo\* roleInfo = [EORoleInfo new];

roleInfo.roleId = @"game\_id\_000002";

roleInfo.roleName = @"roleName2";

roleInfo.roleLevel = @"1";

roleInfo.serverId = @"001";

roleInfo.serverName = @"测试服01";

[SandClass EOEntryGame:roleInfo];

1. 退出接口（非必要接入）

|  |  |
| --- | --- |
| **功能描述** | **发送退出游戏日志** |
| 功能接口 | + (void)EOQuitGame; |
| 调用示例 | [SandClass EOQuitGame]; |

1. 用户中心接口

|  |  |
| --- | --- |
| **功能描述** | **打开用户中心界面** |
| 功能接口 | + (void)userCenter; |
| 调用示例 | [SandClass userCenter]; |

1. 设置SDK语言

|  |  |
| --- | --- |
| **功能描述** | **打开用户中心界面** |
| 功能接口 | + (void)gameLanguage:(NSString\*)language |
| language | Sdk要使用的语言,示例：zh-tw, en-us |
| 调用示例 | [SandClass gameLanguage:@"zh-tw"]; |

1. 删除未完成的测试订单接口（测试环境下可用）

|  |  |
| --- | --- |
| **功能描述** | **debug阶段移除”存在未完成的订单”** |
| 功能接口 | + (void)debugDeleteUnCompleteOrder; |
| 调用示例 | [SandClass debugDeleteUnCompleteOrder]; |
| //如果测试阶段支付过程中提示"当前支付未完成,请重新进入游戏或联系客服"  //在登录完成后调用下面的方法 | |

1. 应用生命周期调用

-(BOOL)application:(UIApplication\*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

{

//facebook回调

[SandClass facebookApplication:application didFinishLaunchingWithOptions:launchOptions];

return YES;

}

-(BOOL)application:(UIApplication\*)application

openURL:(NSURL\*)url sourceApplication:(NSString \*)sourceApplication annotation:(id)annotation

{

// facebook回调

return [SandClass facebookApplication:application openURL:url sourceApplication:sourceApplication annotation:annotation];

}