

iOS-SDK接入文档之Facebook数据篇

引入 SDK 文件

/**

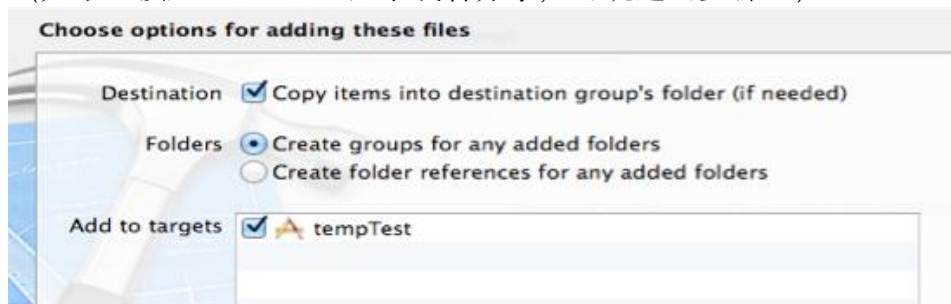
FBSDK13.0.0 及以上版本已不再支持 Xcode12，须使用 xcode13.2.1 或以上版本接入。

FBSDK15.1.0 及以上版本已不再支持 Xcode13，须使用 xcode14.0 或以上版本接入。

FBSDK17.0.0 及以上版本已不再支持 Xcode14，须使用 xcode15.0 或以上版本接入(FBSDK17.0.2 要求 Xcode15.3+)。

**/

将 SDK 文件夹下 FBSDKCoreKit.framework, FBAEMKit.framework, FBSDKCoreKit_Basics.framework 文件引入到接入工程并选择正确的 target。
(如果已接入 Facebook 登录或者分享，可跳过此步骤。)



FBSDK从17.3.0版本开始改用动态库版本，请以动态库的方式加载即把FBxxxx.framework设置为Embed & Sign。

检查target->Build Setting-> Runpath search path是否包含
@executable_path/Frameworks如果没有请添加，若有请继续下一步骤。

包含 UnityFramework 动态库特别说明:

FBSDKCoreKit.framework/FBAEMKit.framework/FBSDKCoreKit_Basics.framework/
为动态库 TargetMembership 需要关联到 UnityFramework 和 Unity_iPhone。

UnityFramework 仅关联即可，不需要也不可以设置为 Embed & Sign，否则提交 AppStore 时会报错。

Unity-iPhone 不仅需要关联，需要且必须设置为 Embed & Sign，否则启动就崩溃。

接入工程配置

FBSDK 需要接入工程支持 Swift 和 OC 混编。以下 2 种方案任选其一

方案 1: 需要在工程中创建一个 swift 文件，然后根据 Xcode 提示创建 OC 和 swift 桥接文件。

方案 2: target->Build Settings->Library Search Path 添加以下语句:

```
$(TOOLCHAIN_DIR)/usr/lib/swift/$(PLATFORM_NAME)
```

```
$(SDKROOT)/usr/lib/swift
```

Target->Build Settings->Runpath Search Path 最上面添加以下语句:

请注意确保/usr/lib/swift 加在@executable_path/Frameworks 的上面, 否则游戏运行时可能会崩溃。

```
/usr/lib/swift
```

包含 UnityFramework 动态库特别说明:

所有的操作都在 UnityFramework 上进行。

方案 1: 需要在工程中创建一个 swift 文件, 可只关联到 UnityFramework 或同时关联到 UnityFramework 和 Untiy-iphone。

方案 2: UnityFramework->Build Settings->Library Search Path 添加以下语句:

```
$(TOOLCHAIN_DIR)/usr/lib/swift/$(PLATFORM_NAME)
```

```
$(SDKROOT)/usr/lib/swift
```

UnityFramework->Build Settings->Runpath Search Path 最上面添加以下语句:

请注意确保/usr/lib/swift 加在@executable_path/Frameworks 的上面, 否则游戏运行时可能会崩溃。

```
/usr/lib/swift
```

FacebookAppID:facebook 后台应用参数;

FacebookDisplayName:facebook 后台配置的应用名称{游戏名称};

FacebookClientToken:在 facebook 后台应用设置->高级->客户端口令查看;

LSApplicationQueriesSchemes:Facebook 相关白名单。

✓ LSApplicationQueriesSchemes	◇	Array	(5 items)	
Item 0		String	fbapi	
Item 1		String	fb-messenger-api	
Item 2		String	fbauth2	
Item 3		String	fbshareextension	
Item 4		String	fb-messenger-share-api	
DTSDKBuild	◇	String	19A339	
FacebookAdvertiserIDCollectionEnabled	◇	Boolean	1	
Bundle version string (short)	◇	String	1.0.0	
> CFBundleSupportedPlatforms	◇	Array	(1 item)	
> Supported interface orientations	◇	Array	(1 item)	
BuildMachineOSBuild	◇	String	21A559	
DTPlatformBuild	◇	String	19A339	
Bundle OS Type code	◇	String	APPL	
DTXcodeBuild	◇	String	13A1030d	
Localization native development region	◇	String	English	
MinimumOSVersion	◇	String	9.0	
Bundle version	◇	String	3	
Icon already includes gloss effects	◇	Boolean	YES	
Status bar is initially hidden	◇	Boolean	YES	
FacebookAppID	◇	String	{{FacebookAppID}}	替换成游戏的 fbappid
> UIDeviceFamily	◇	Array	(2 items)	
Launch screen interface file base name	◇	String	LaunchScreen	
Bundle identifier	◇	String	com.firefantasyxx.ios	
FacebookDisplayName	◇	String	{{FacebookDisplayName}}	游戏的 fb 应用名
DTXcode	◇	String	1310	
FacebookClientToken	◇	String	{{FacebookClientToken}}	游戏的 fbclienttoken

SDK 内部已经投递事件(接入方无需再处理)

Facebook 标准事件(FBSDK 内部已经投递事件):

应用安装	新用户首次激活应用程序或应用程序首次在特定设备上启动。
应用启动	当有人启动您的应用程序时，Facebook SDK 会初始化并记录事件。但是，如果第二个应用程序启动事件在第一个应用程序启动事件的 60 秒内发生，则不会记录第二个应用程序启动事件。
在应用程序内购买	由 Apple App Store 或 Google Play 处理的购买完成后。如果您使用其他付款平台，则需要手动添加购买事件代码。
Facebook SDK 崩溃报告 (仅适用于 Facebook。)	如果您的应用程序由于 Facebook SDK 而崩溃，则当重新启动应用程序时，将生成崩溃报告并发送给 Facebook。该报告不包含用户数据，可帮助 Facebook 确保 SDK 的质量和稳定性。要选择不记录此事件，请禁用自动记录的事件。

应用内购买自动记录的事件

Apple 提供了四种不同的应用内购买类型：消耗品、非消耗品、自动续订和非续订。如果您使用 StoreKit 1 实施应用内购买，我们会自动记录每个应用内购买类型。如果您使用 StoreKit 2 实施应用内购买，我们会自动记录非消耗品、自动续订和非续订。如果您也想自动记录消耗品，您需要向 Info.plist 添加

[SKIncludeConsumableInAppPurchaseHistory](#) 键

```
<key>SKIncludeConsumableInAppPurchaseHistory</key>
<true/>
```

禁用自动事件日志记录

1. 在 Facebook 后台应用设置里面禁用(推荐此方式)
2. 在接入工程中禁用

禁用自动事件日志记录，请将 info.plist 在 Xcode 中以应用程序的 as 代码形式打开，然后将以下 XML 添加到属性字典中或者直接添加一下键值对：

```
<key> FacebookAutoLogAppEventsEnabled </ key> <false />
```

如果禁用了自动事件日志记录，如果游戏仍需要 Facebook 数据统计，请接入方在适当的时机调用下面的接口进行事件投递。

本 SDK 默认埋点

- 1.登录成功: fb_custom_login_user_name (自定义事件)；
- 2.注册来源 FBSDKAppEventParameterNameRegistrationMethod (标准事件)。

接口调用

引入 REDeLoginKit.h

```
#import <JYouLoginKit/REDeLoginKit.h>
```

投递教程完成事件接口

```
/**
 * 完成教程的统计
 * contentData 特点内容
 * contentId 特点内容 id
 * success 是否完成教程
 * tip:
 *     fb 的标准打点事件，事件名: Complete Tutorial
 */
+ (void)logCompleteTutorialEvent:(NSString *)contentData
                        contentId:(NSString *)contentId
                        success:(BOOL)success;
```

投递角色升级事件接口

```
/**
 * 角色升级的统计
 * level 角色等级
 * tip:
 *     fb 的标准打点事件，事件名: Achieve Level
 *
 *     fb 的标准打点事件，事件名: Purchase FB 默认已经接入可不接
 */
+ (void)logPurchase:(double)purchaseAmount
                currency:(NSString *)currency
                parameters:(NSDictionary<NSString *, id> *)parameters;
```

投递成就解锁事件接口

```
/**
 * 成就解锁
 * description 成就解锁的描述
 * type 成就解锁类型
 * tip:
 *     fb 的标准打点事件，事件名: Unlock Achievement
 */
+ (void)logUnlockAchievementEvent:(NSString *)description
                                type:(NSString *)type;
```

投递发起结账事件接口

```
/**
```

```

* 发起结账
* contentData 商品名称, 可以是 json 对象列表如[{"id": "1234",
"quantity": 2}, {"id": "5678", "quantity": 1}]
* contentId 商品 id
* contentType 商品类型 product or product_group
* numItems 商品数量
* currency 货币类型
* totalPrice 总价
* tip:
    fb 的标准打点事件, 事件名: Initiate Checkout
*/
+ (void)logInitiateCheckoutEvent:(NSString *)contentData
    contentId:(NSString *)contentId
    contentType:(NSString *)contentType
    numItems:(NSInteger)numItems
    paymentInfoAvailable:(BOOL)paymentInfoAvailable
    currency:(NSString *)currency
    valueToSum:(double)totalPrice;

```

投递自定义事件接口

```

/**
* 自定义事件
* eventName 事件名
* valueToSum 开启后系统会根据这个事件关联一个值, 并按这个事件发生的全部情况
求出总和, 便于查看平均值, 当为 0 默认不开启
* parameters 附带参数, 可以为该事件添加参数, 参数 key 自定义如:
 @{@"order":orderid}
* tip:
    fb 的自定义事件
*/
+ (void)logEvent:(NSString *)eventName
    valueToSum:(double)valueToSum parameters:(NSDictionary<NSString *,
id> *)parameters;

```

应用跳转回调(必接)

类: [REDeLoginKit](#)

函数: `+(void)application:(UIApplication *)application openURL:(NSURL *)url`

`options:(NSDictionary *)options;`

功能: 处理第 3 方应用回调结果

在 `-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url`

`options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options;` 中调用。

类: [REDeLoginKit](#)

函数: `+(void)application:(UIApplication *)application openURL:(NSURL *)url`

`sourceApplication:(NSString *)source annotation:(id)annotation;`

功能: 处理第 3 方应用回调结果

在 `-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url`

sourceApplication:(NSString*)sourceApplication annotation:(id)annotation; 中调用。

示例:

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString*)sourceApplication annotation:(id)annotation
{
    [REDeLoginKit application:application openURL:url sourceApplication:sourceApplication annotation:annotation];
    return YES;
}

- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {
    [REDeLoginKit application:application openURL:url options:options];
    return YES;
}
```