

iOS-SDK接入文档之核心篇

一、SDK 结构

请参照文档接入，如有疑问请参考 demo，最终以 demo 为准。

1. JYouLoginKit.framework 和 SSBundle.bundle 为 SDK 核心框架和资源。

支持平台：iPod Touch,iPhone,iPad。系统要求:iOS12.0+，

支持框架：arm64

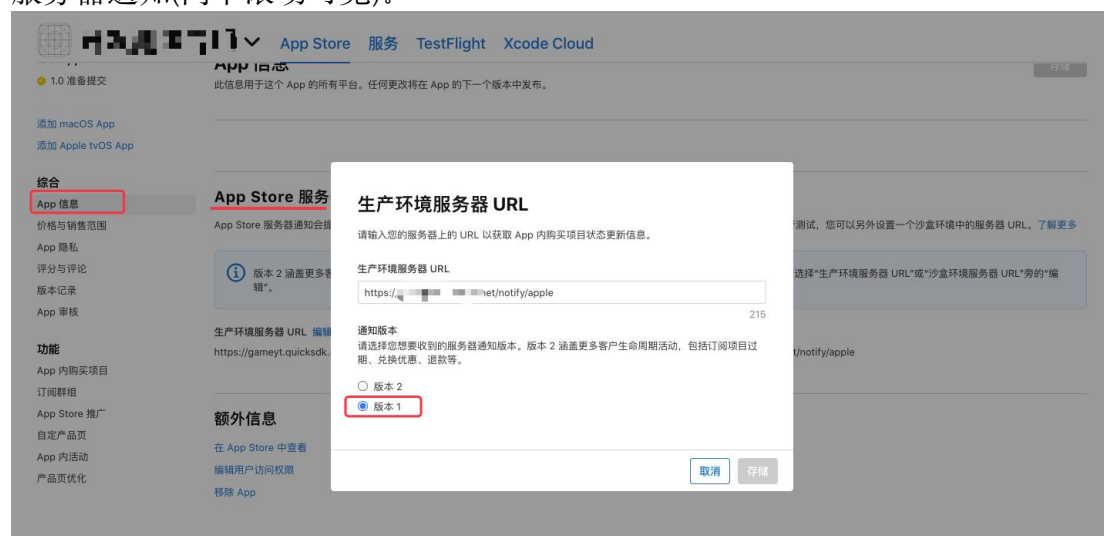
环境要求：Xcode15.3.0+。

二、注意事项

游戏内支持购买自动订阅型商品或者需要检测苹果退款时，需做如下配置：

在苹果后台配置服务器通知网址，URL 格式：xxx/notify/apple。其中 xxx 为厂商的 sdk 完整域名，必须带 https。配置示例如下图：

配置路径:我的 app->具体 app-> 点按 App 信息(侧边栏综合下方)->App store 服务器通知(向下滚动可见)。



另外，游戏内支持购买自动订阅型商品还需要在 quickgame 后台产品详情里配置苹果后台的共享密钥，

.framework 动态库 TargetMembership 需要同时关联到 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。

JYouLoginKit.framework 为静态库，TargetMembership 需要关联到 UnityFramework;

SSBundle.bundle 资源文件，TargetMembership 需要关联到 Unity-Iphone。



工程配置

在 Build Settings->Linking->Other Linking Flags 中配置 -ObjC



SDK 权限申请

当支持数据统计时，如带有 Appsflyer, Adjust, FB 等数据统计 SDK，需添加 ATT 权限，征得用户启用跟踪权限许可才能跟踪或访问其设备的广告标识符（即 IDFA）。

SDK 默认需要获取 ATT 权限，如不需要可与我们联系移除。

ATT 权限

接入工程 info.plist 文件配置如下权限，右边的文案需要接入方自定义并适配多语言。

Information Property List	Dictionary	{31 items}
Localization native development region	String	en
Privacy - Tracking Usage Description	String	您的数据只会用于给您投放个性化广告 (测试时使用正式需修改)
Executable file	String	\$(EXECUTABLE_NAME)
Icon files (iOS 5)	Dictionary	{0 items}

相机和相册权限

SDK 同时在用户中心头像选择会用到相机和相册权限，需要在接入工程 info.plist 文件配置如下权限，右边的文案需要接入方自定义(需要清楚的解释申请权限用途)并适配多语言。

Privacy - Photo Library Usage Description	String	更改头像时需要访问你的相册 (测试用,正式需修改)
Privacy - Camera Usage Description	String	更改头像时需要访问你的相机 (测试用,正式需修改)
Application supports iTunes file sharing	Boolean	YES

接口调用

引入 REDeLoginKit.h

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

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。

SDK 初始化（必接）

函数:

```
// 初始化 设置产品,最先需要调用的接口(必接) 与初始化带设置默认语言
二选一
+ (void)initSDKWithProductCode:(NSString *)productCode
callback:(id<REDeInitCallback>)initDelegate;
/** 初始化 设置产品,最先需要调用的接口(必接) 与初始化不带设置默认语言
二选一
@param productCode sdk 参数从后台获取
```

```

    * @param language sdk 显示语言，（简体中文@"zh-CN"或"zh_Hans"，中文繁体@"zh-hk" 或"zh-Hant"，德语@"de"，英语@"en-us"，法语@"fr"，日语@"ja"，韩语@"ko"，俄语@"ru"，泰语@"th"，印尼语@"id"，越南语@"vi"，土耳其语@"tr"，阿拉伯语@"ar"，）
    * @param initDelegate 初始化回调接收对象
    */
+ (void)initSDKWithProductCode:(NSString *)productCode
language:(NSString *)language
callback:(id<REDeInitCallback>)initDelegate;

```

功能：使用产品 id，初始化 SDK，建议启动 App 之后尽快调用。

参数：productCode, 产品 id，必填，对接商务提供。

初始化成功回调方法：

// SDK 初始化成功回调

```

// 初始化完成
- (void)qqSDKInitDone;

```

设置是否需要自动登录（选接）

```

/** 设置是否自动登录，默认自动登录 YES*/
+ (void)setNeedAutoLogin:(BOOL)autoLogin;

```

功能：是否启用自动登录上次登录的账号，需要在调用登录之前调用。默认开启。

设置游客登录是否显示（选接）

```

/**
 * 是否启用 Guest 登录，YES 启用 NO 禁用 默认 YES
 */
+ (void)guestLoginEnable:(BOOL)yesOrNo;

```

功能：设置是否显示游客登录，YES:显示 NO 不显示，默认 YES。

设置游客登录是否弹窗提示（选接）

```

/** 是否启用游客绑定提示 YES:禁用 NO 启用 默认 NO */
+ (void)guestBingTipDisable:(BOOL)yesOrNo;

```

功能：设置是否开启游客登录成功之后安全提示，YES:不弹窗 NO 弹窗，默认 NO

设置用户中心头像右侧是否显示 uid（选接）

```

/** 设置个人中心界面是否显示 uid，默认 NO 不显示*/
+ (void)setNeedShowUid:(BOOL)show;

```

功能：设置是否在用户中心头像右侧显示 UID，YES 显示 NO 不显示，默认 NO。

设置登录和登出回调监听者（必接）

函数：

```
// 设置登录回调监听对象(必接)
+
(void)setFunctionLoginCallback:(id<REDeLoginCallback>)loginDelegate;
```

功能：设置登录回调监听者需要先设置监听者再调登录方法，监听者实现如下的回调方法

必须实现方法：

```
@required
// 在 SDK 的个人中心主动退出登录
- (void)userLogout;
// 绑定回调绑定三方账号或绑定邮箱成功回调
- (void)bindUid:(NSString *)uid userToken:(NSString *)token
type:(USERCENTER_TYPE)type;
// 解绑回调解绑三方账号成功回调
- (void)unBindUid:(NSString *)uid userToken:(NSString *)token
type:(USERCENTER_TYPE)type
;tips: type 解绑类型
USERCENTER_TYPE_EMAIL = 1, //email
USERCENTER_TYPE_FB = 6, //FB
USERCENTER_TYPE_GOOGLEPLUS = 8, //Google
USERCENTER_TYPE_Line = 11, //Line
USERCENTER_TYPE_GAMECENTER = 7, //GameCenter
USERCENTER_TYPE_Apple = 16, //Apple
```

按需可选实现方法：（登录成功回调必须二选一）

```
@optional
@optional
/* 登录成功与带登录方式实现其一即可 */
- (void)loginUid:(NSString *)uid userToken:(NSString *)token;
/* 登录成功带登录方式与不带登录方式实现其一即可 */
- (void)loginUid:(NSString *)uid userToken:(NSString *)token
type:(USERCENTER_TYPE)type;
/* 调用 logout 执行成功后执行回调, 与用户在 SDK 的个人中心主动退出登录回调不同 */
- (void)gameLogoutSuccess;
/* 点击了个人中心的客服按钮时回调 */
- (void)onClickServiceCenter;
/** 登录之后游戏主动唤起 SDK 页面，关闭时会回调此方法 */
- (void)sdkUserPageWillClose;
/** 玩家取消登录，主要用于单独调起某个三方登录方法
isShow: YES:玩家在登录界面显示的情况下取消三方登录, NO:玩家在登录界面未显示的情况下取消三方登录
*/
- (void)userCancelLoginWithLoginPageShowing:(BOOL)isShow;
/** 玩家登录失败，主要用于单独调起某个三方登录方法
isShow: YES:玩家在登录界面显示的情况下登录失败, NO:玩家在登录界面未显示的情况下登录失败
message: 失败原因
*/
```



```

- (void)userLoginFailWithLoginPageShowing:(BOOL)isShow
message:(NSString *)message;
/** 玩家取消绑定，主要用于单独调起某个三方绑定方法
isShow: YES:玩家在 SDK 界面显示的情况下取消三方登录，NO:玩家在
SDK 界面未显示的情况下取消三方登录
*/
-
- (void)userCancelBindWithKitWindowShowing:(BOOL)isShow;
/** 玩家绑定失败，主要用于单独调起某个三方绑定方法
isShow: YES:玩家在 SDK 界面显示的情况下登录失败，NO:玩家在
SDK 界面未显示的情况下登录失败
message: 失败原因
*/
- (void)userBindFailWithKitWindowShowing:(BOOL)isShow
message:(NSString *)message;
/** 玩家取消解除绑定，主要用于单独调起某个三方解除绑定方法
isShow: YES:玩家在 SDK 界面显示的情况下取消三方登录，NO:玩家在
SDK 界面未显示的情况下取消三方登录
*/
-
- (void)userCancelUnbindWithKitWindowShowing:(BOOL)isShow;
/** 玩家解除绑定失败，主要用于单独调起某个三方解除绑定方法
isShow: YES:玩家在 SDK 界面显示的情况下登录失败，NO:玩家在
SDK 界面未显示的情况下登录失败
message: 失败原因
*/
-
- (void)userUnbindFailWithKitWindowShowing:(BOOL)isShow
message:(NSString *)message;
/** 登录流程事件回调，
loginEvent: 事件枚举值 枚举值定义可参考头文件 REDeDelegate.h
message: failed 事件返回失败原因，其他返回枚举值字符串
*/
- (void)onEvent:(LoginEvent)loginEvent message:(NSString *)message;

```

获取是否存在上次登录账号信息（选接接）

```

/** 获取是否存在上次登录账号信息 YES: 存在， NO: 不存在 */
+ (BOOL)checkLoginValid;

```

功能：获取是否存在上次登录信息，如果存在返回 YES，不存在返回 NO。
返回 YES 的情况下调用快捷登录会登录上次登录的账号。

登录（必接如果使用了静默登录可不接）

函数：

```
/**进入用户登录页面。收到切换用户通知或者主动注销用户后调用（必接）
@method      isShowMenu      YES:登录自动显示浮标    NO: 登录不自动显示浮标
*/
+ (void)loginWithMenuShow:(BOOL)isShowMenu;
```

功能：进入用户登录页面。需要在初始化成功之后调用，否则调用无效。游戏开始时可以调用此接口显示一个用户登录界面。在收到用户注销回调后通常处理办法是回到游戏登录界面调用后调用此接口。

isDisplay： YES:登录自动显示浮标 NO: 登录不自动显示浮标。

快捷登录(静默登录)—按需选接

函数：

```
// 自动注册和自动登录流程, 玩家首次游戏无需注册, 达到快速游戏的目的, 适合于没有
// 登录按键的游戏在启动后调用
+ (void)fastlyStartGame;
```

功能：不显示登录界面静默登录，如果本地没有账号将基于本设备创建一个新的游客账号，可以在用户中心进行绑定和解绑操作、切换账号。如果调用此接口时 SDK 还没有初始化成功，则 SDK 初始化成功时会自动调用此接口一次。快速进入游戏可以使用此接口，用户注销后重新登录使用 login 接口。

快捷调起除账密登录外其他登录方式登录—按需选接

```
// 主动调用除账密登录外的其他方式登录，无 SDK 登录界面
+ (void)loginAccountType:(USERCENTER_TYPE)type;
```

功能：不显示 SDK 登录界面，单独唤起除账密登录外的其他某一登录方式。包含游客登录和其他三方登录。

获取所有绑定信息—按需选接

函数：

```
// 获取当前账号绑定信息
+ (NSDictionary *)getUserBindInfo;
```

功能：获取当前用户三方账号和邮箱的绑定信息。

字典 key	字典 value 类型	字典 key 注释
USERCENTER_TYPE_EMAIL	NSNumber	Email

字典 key	字典 value 类型	字典 key 注释
USERCENTER_TYPE_FACEBOOK	NSNumber	FaceBook
USERCENTER_TYPE_GOOGLEPLUS	NSNumber	Google
USERCENTER_TYPE_LINE	NSNumber	Line
USERCENTER_TYPE_GAMECENTER	NSNumber	GameCenter
USERCENTER_TYPE_APPLE	NSNumber	Apple

代码实例:

```

NSDictionary *DIC = [REDELOGINKIT GETUSERBINDINFO];
IF ([DIC[@"USERCENTER_TYPE_EMAIL"] BOOLVALUE]) {
    NSLog(@"绑定了邮箱");
}

```

设置角色信息（必接）

函数:

```

// 设置角色信息 选择角色进入游戏或者角色信息发生变化时需要调用 必接
+ (void)setGameRoleInfo:(REDeRoleInfo *)roleInfo;

```

功能: 上报角色信息, 在创角、进入游戏、等级变化时调用。

主动调用绑定账号—按需选接

函数:

```

// 主动调用绑定账号
+ (void)bindAccountType:(USERCENTER_TYPE)type;

```

功能: 在外部主动调用绑定账号。

获取用户 ID—按需选接

函数:

```

/**
 * @method userID
 * @return 返回用户 id。如未登录, 返回空。
 * //通常在登录成功回调中调用
 */
+ (NSString *)userID;

```

功能: 获取当前登录用户的用户 ID。在用户登录成功后调用。

返回值：当前登录用户的用户 ID,未登录返回 nil。

获取登陆后 token—按需选接

函数：

```
// 获取用户登录账号。如未登录, 返回空。  
+ (NSString *)getUserAccount;
```

功能：获取当前登录用户的用户 token。用户登录成功时调用。

返回值：当前登录用户的用户 token，用以验证用户有效性,未登录返回 nil。

获取当前用户是否游客账号-选接

函数：

```
// 用户是否是游客, 请先判断用户是否有登录  
+ (BOOL)isUserGuest;
```

功能：获取当前用户是否为游客账号。

返回值：YES 为游客账号，No 为非游客账号。

获取设备 ID-选接

函数：

```
// 获取设备 ID  
+ (NSString *)getDeviceID;
```

功能：获取当前设备 SDK 内设置的 ID。

配置渠道号-选接

找到 SDK 内 SSBundle.bundle 文件下 pjInfo.txt 文件修改 channelId、channelDesc。

注意末尾必须添加英文“,”作为分割符。



获取渠道号-选接

函数：

```
// 获取渠道号  
+ (NSString *)channelCode;
```

功能：获取当前包体渠道号，默认 default。

获取国家码-选接

函数:

```
// 获取国家码  
+ (NSDictionary *)getNationCode;
```

功能: 获取当前设备设置地区所在国家码。

返回结果示例:

```
国家码: {  
    countryCode = JP;  
    ip = "xxx.xxx.xxx.xxx";  
    ipcountryCode = CN;  
}
```

countryCode = JP;//设备系统设置的国家地区代码

ip = "xxx.xxxx.xxx.xxx";//设备网络 ip 地址

ipcountryCode = CN;//根据 ip 获取的国家码

获取当前用户是否是新用户

函数:

```
// 判断是否是新用户, YES 为新用户, NO 为已经注册过的用户  
+ (BOOL)isNewUser;
```

功能: 获取当前用户是否为新用户。

返回值: YES 为新用户, No 为已经注册用户。

退出登录—必接

函数:

```
//@method    logout 退出登录。  
+ (void)logout;
```

功能: 立即将当前用户登出。

获取 Storefront 国家码—选接

```
/** 同步获取 Storefront 国家码  
 */  
+ (NSString *)getStorefrontCountryCode;  
/** 异步获取 Storefront 国家码-推荐  
 */  
+  
(void)getStorefrontCountryCodeWithCompletion:(void(^)(NSString *countryCode))completion;
```

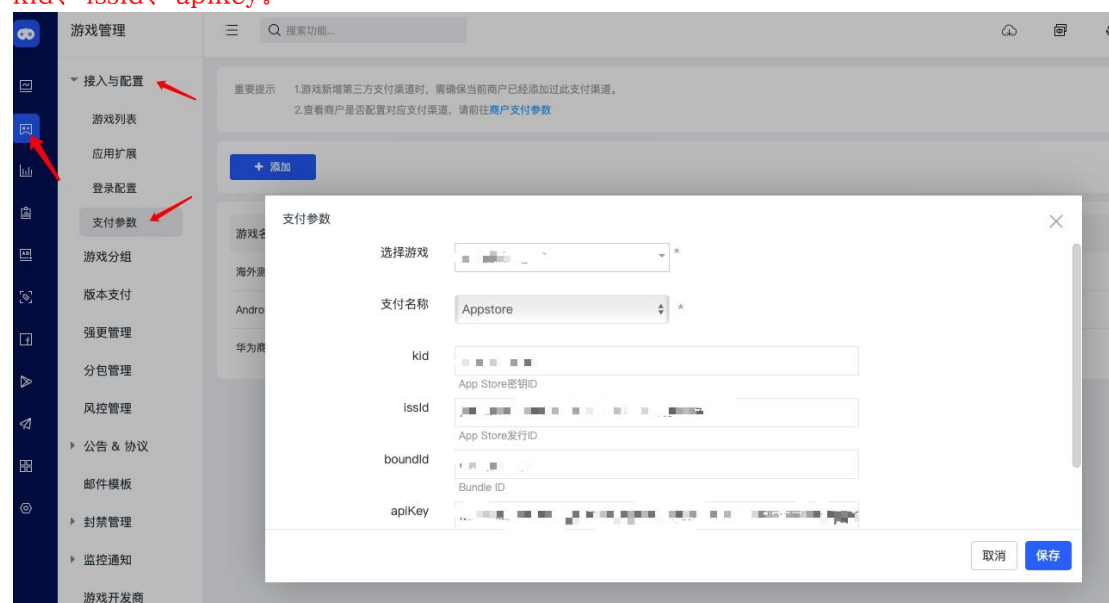
```

/** 设置 Storefront 国家码更新 Block 回调
 */
+
(void)configStorefrontCountryCodeUpdatedBlock:(void (^)(NSString * countryCode))updateBlock;

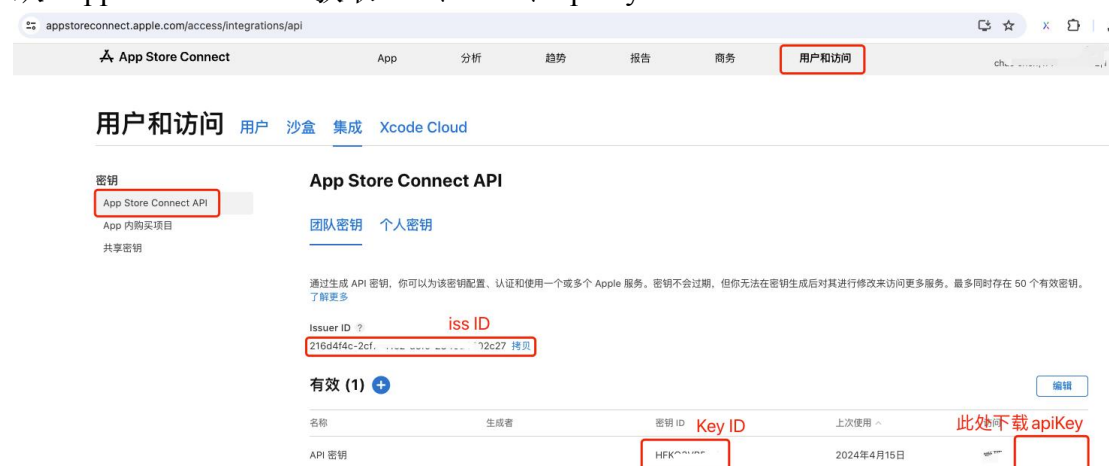
```

支付（必接）

SDK 从 2.0.4.3 开始启用 Storekit2（仅支持 iOS15.0+，iOS15.0 以下使用 Storekit1），须先在 SDK 服务后台（游戏管理->接入与配置->支付参数）配置 kid、issid、apikey。



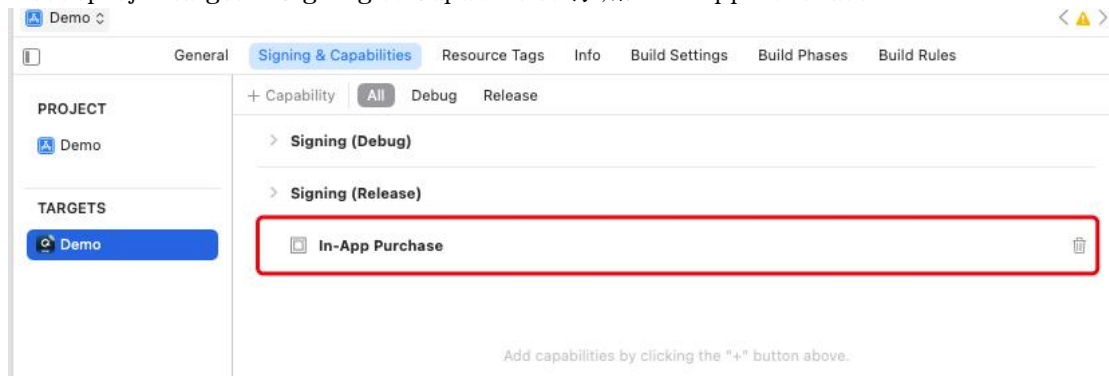
从 App Store Connect 获取 kid、issid、apikey:



App Store Connet 创建密钥

工程配置

Xcodeproj->target->Signing & Capabilities 添加 In-App Purchase.



函数:

```
// 购买接口(必接)
+ (void)IAPWithParameter:(REDeOrderInfo *)param;
```

功能: 进入充值模块, 目前仅支持苹果内购。

参数: param 充值参数信息,参考 REDeOrderInfo .h 文件 , 注意: 其中扩展字段请勿传特殊符号, 如果无法避免建议先进行 base64 编码后再传; 回调地址可后台配置, 后台配置了回调地址就以后台配置的为准

说明: 异步回调, sdk 通知的结果仅供参考, 以服务器端同步为准。

REDeOrderInfo 模型属性:

```
@property (copy, nonatomic) NSString *productId; //必填
开发者后台的商品 ID, productId
@property (copy, nonatomic) NSString*product_order_no; //
游戏产品订单号必传
@property (copy, nonatomic) NSString *subject; //虚拟
商品名称 必填
@property (copy, nonatomic) NSString *desc; //商品
描述 可选
@property (copy, nonatomic) NSString *price; //
商品单价 可选
@property (assign, nonatomic) unsigned int quantity; //
购买数量 可选 默认 1
@property (copy, nonatomic) NSString *total; //商品总
价 必填, 影响购买统计和后台回调
@property (copy, nonatomic) NSString
*callback_url; //回调通知地址 string[200] 可选, (可后台
配置, 后台配置了回调地址就以后台配置的为准)
@property (copy, nonatomic) NSString
*extras_params; //扩展参数 string[500] 可选, 透传字段, 注
```

意：扩展字段请勿传特殊符号，如果无法避免建议先进行 base64 编码后再传，用于查找商品信息返回商品属性时返回货币符号

```
@property (copy, nonatomic) NSString * displayPrice;/** 商品显示价格 用于查找商品信息返回商品属性*/
@property (copy, nonatomic) NSString * currencySymbol;/** 商品货币符号 用于查找商品信息返回商品属性*/
@property (copy, nonatomic) NSString * currencyCode;/** 商品货币代码 用于查找商品信息返回商品属性*/
/** 是否家庭共享 可选字段，主要用于查找商品信息返回商品属性 */
@property (assign, nonatomic) BOOL isFamilyShareable
API_AVAILABLE(ios(14.0));
/** 商品类型 1 消耗品 2 非消耗类商品 3 自动续期订阅 4 非续期订阅 可选字段，主要用于 storekit2 查找商品信息返回商品属性 */
@property (assign, nonatomic) NSInteger productType
API_AVAILABLE(ios(15.0));
/** 订阅品是否有资格享受优惠 可选字段，主要用于 storekit2 查找商品信息返回商品属性 */
@property (assign, nonatomic) BOOL isEligibleForIntroOffer
API_AVAILABLE(ios(15.0));
```

注:ReDeOrderInfo 从 2.0.4.3 开始新增新属性 displayPrice、currencySymbol、currencyCode、isFamilyShareable、productType、isEligibleForIntroOffer，供 (iOS15.0+)查询商品信息回调时使用，调用支付接口时可不传。

设置支付回调监听者（必接）

函数：

```
// 设置支付回调监听对象(必接)
+ (void)setFunctionBuyCallback:(id<REDeBuyCallback>)buyDelegate;
+ (void)setFunctionBuyCallback:(id<REDeBuyCallback>)buyDelegate;
```

功能：设置支付回调监听者然后监听者实现如下的回调方法就可以收到支付事件：

```
@protocol REDeBuyCallback <NSObject>
@optional
#pragma mark - 两个购买失败回调任选其一实现回调即可
/* 购买失败，无回传参数 */
- (void)purchaseFail;
/* 购买失败，带回传参数
productId 内购商品 Id
orderNo SDK 订单号
gameOrderNo 游戏订单号，如果本地缓存被清理会返回空字符串
errorCode 错误码 2:用户取消购买 -1:sdk 内错误 -2:设备未开启内购 -3:找不到商品信息 0:未知错误编码 其他服务端返回错误或者 apple 返回错误码
```



```

message 错误描述
*/
- (void)purchaseFailWithProductId:(NSString *)productId gameOrderNo:(NSString *)gameOrderNo
errorCode:(NSInteger)errorCode message:(NSString *)message;
#pragma mark - 四个购买成功回调任选其一实现回调即可
/* 这里的成功回调不能作为发货依据
    购买完成回调
    productId 内购商品 Id
    orderNo SDK 订单号
    gameOrderNo 游戏订单号, 如果本地缓存被清理会返回空字符串
    transactionIdentifier 票据 id
    receiptString 票据凭证
*/
- (void)purchaseDoneProductId:(NSString *)productId
orderNo:(NSString *)orderNo gameOrderNo:(NSString *)gameOrderNo
transactionIdentifier:(NSString *)transactionIdentifier
appStoreReceiptBase64EncodedString:(NSString *)receiptString;
/* 这里的成功回调不能作为发货依据
    购买完成回调
    productId 内购商品 Id
    orderNo SDK 订单号
    gameOrderNo 游戏订单号, 如果本地缓存被清理会返回空字符串
    receiptString 票据凭证
    iOS15.0 以上版本使用 storekit2
    appStoreReceiptBase64EncodedString 为@""
*/
- (void)purchaseDoneProductId:(NSString *)productId
orderNo:(NSString *)orderNo gameOrderNo:(NSString *)gameOrderNo
appStoreReceiptBase64EncodedString:(NSString *)receiptString;
/* 这里的成功回调不能作为发货依据
    购买完成回调
    productId 内购商品 Id
    orderNo SDK 订单号
    gameOrderNo 游戏订单号, 如果本地缓存被清理会返回空字符串
*/
- (void)purchaseDoneProductId:(NSString *)productId
orderNo:(NSString *)orderNo gameOrderNo:(NSString *)gameOrderNo;
/* 这里的成功回调不能作为发货依据

```

```

    购买完成回调
    productId 内购商品 Id
    orderNo SDK 订单号
    */
- (void)purchaseDoneProductId:(NSString *)productId
orderNo:(NSString *)orderNo;
/** iOS15+用户提交退款申请成功回调 */
- (void)storeKit2RequestRefundSuccess:(NSString
*)transactionID;
/** iOS15+用户取消申请退款回调 */
- (void)storeKit2CancelRequestRefund:(NSString
*)transactionID;
/** iOS15+用户提交退款申请失败回调 */
- (void)storeKit2RequestRefundFail:(NSString
*)transactionID message:(NSString *)message;
@end

```

恢复购买非消耗品—按需选接

函数:

```

// 获取已购买的非消耗商品或者订阅商品, 商品信息通过回调返回(通常游戏自己也能获取到这些商品信息, 支持购买自动订阅商品或者非消耗商品时选接)
+
(void)restoreNonConsumptionProducts:(id<REDeRestoreCallback>)restoreDelegate;

```

功能: 获取已购买的非消耗商品或者订阅商品, 商品信息通过回调返回, 需实现如下回调方法:

//恢复非消耗商品成功, 返回商品 id 信息

```

// 恢复非消耗商品成功, 返回商品 id 信息
- (void)restoreSuccess:(NSArray *)products;
//恢复失败

```

```

// 恢复失败
- (void)restoreFail:(NSString *)msg;

```

获取商品信息-按需选接

函数

```

// 根据传入的商品 id 列表获取商品信息, 结果通过回调返回
+ (void)findProductInfoWithProductIds:(NSArray *)productArr
delegate:(id<REProductInfoCallback>)productDelegate;

```

功能: 根据传入的商品 id 列表获取商品信息, 商品信息通过回调返回。

参数: 商品 id 组成的数组, 商品 id 类型为字符串, productDelegate: 用于接收商品信息回调的对象。

通过后台配置的商品列表获取商品信息-按需选接

函数

```

// 通过 SDK 后台配置的商品 id 列表获取商品信息, 结果通过回调返回

```

```
+(void)findProductInfoWithDelegate:(id<REProductInfoCallback>)productDelegate;
```

功能：根据 SDK 后台配置的商品 id 列表获取商品信息，商品信息通过回调返回。

参数：productDelegate：用于接收商品信息回调的对象。

//查询商品信息成功回调

```
// 查找商品信息成功数组元素为 REDeOrderInfo 实例
```

```
-(void)findProductInfoSuccess:(NSArray *)products;
```

功能：查询商品信息成功时，回调此方法，回调参数类型为数组，元素为 REDeOrderInfo 实例，示例如下：



//查询商品信息失败回调

```
// 查找商品信息失败
```

```
-(void)findProductInfoFail:(NSString *)msg;
```

功能：查询商品信息成功时，回调此方法，回调参数类型为字符串，阐明失败缘由。

展示订阅管理界面(仅支持 iOS15.0+)—按需选接

```
/** 展示订阅管理界面 */
```

```
+(void)showAllSubscriptionsUI API_AVAILABLE(ios(15.0));
```

APP 内退款申请—按需选接

```
/** 退款申请 transactionID 苹果交易 id */
```

```
+(void)showRefundWithTransactionID:(NSString *)transactionID  
API_AVAILABLE(ios(15.0));
```

退款申请结果回调<ReDeBuyCallback>

```
/** iOS15+用户提交退款申请成功回调 */
```

```
-(void)storeKit2RequestRefundSuccess:(NSString *)transactionID;
```

```
/** iOS15+用户取消申请退款回调 */
```

```
- (void)storeKit2CancelRequestRefund:(NSString *)transactionID;
/** iOS15+用户提交退款申请失败回调 */
- (void)storeKit2RequestRefundFail:(NSString *)transactionID message:(NSString *)message;
```

APP 内优惠码兑换界面—按需选接

```
/** 展示优惠码兑换界面 */
void showOfferCodeRedeemSheetUI APL_AVAILABLE 16.0
```

进入用户中心—按需选接

函数：

```
// 进入用户中心
+ (void)enterUserCenter;
```

功能：进入用户中心页面，显示关联其他平台账号的信息和操作入口。

隐藏用户中心—按需选接

函数：

```
/** 关闭用户中心 */
+ (void)dismissUserCenter;
```

功能：关闭 SDK 个人中心界面，需要在登录之后调用。

显示浮动菜单—按需选接

函数：

```
/** 显示浮标，isLeft 浮标是否居左 originalY 浮标纵向位置起始点 */
+ (void)showFloatButtonIsLeft:(BOOL)isLeft
buttonOriginalY:(CGFloat)originalY;
```

功能：在游戏窗口显示悬浮菜单。

隐藏浮动菜单—按需选接

函数：

```
// 隐藏浮动菜单
+ (void)dismissMenu;
```

功能：隐藏显示在游戏窗口内的悬浮菜单。

进入客服中心—如果部署时没有约定则没有此方法请忽略，否则按需选接

函数：

```
/** 进入客服中心 */
```

```
+ (void)enterServiceCenter;
```

功能：唤起客服界面，方便用户联系客服人员。

账号删除

```
/** 设置删除账户提示文案，不设置将使用默认文案 */
```

```
+ (void)configAccountDeletionTipContent:(NSString *)tipContent;
```

参数：tipContent 提示用户即将删除账号数据的文案内容，如果传 nil 或者传空字符串将使用 SDK 默认文案。如果需要使用自定义文案需要在调用 accountDeletion 前调用此方法。

```
/** 删除账户，需要用户确认 */
```

```
+ (void)accountDeletion;
```

功能：游戏开发者调用此 SDK 方法即会弹出确认注销账号界面，用户点击确认后，执行账号删除逻辑。

点击浮标的个人中心页面也有账号删除入口，功能与直接调用此方法相同。

```
/** 直接删除账户，不需要用户确认 */
```

```
+ (void)accountDeletionWithoutConfirm;
```

功能：游戏开发者调用此 SDK 方法，无需用户确认，立马执行账号删除逻辑。

```
/** 设置个人中心界面是否显示删除账号入口，默认显示 YES */
```

```
+ (void)setNeedShowAccountDeletionInUserCenter:(BOOL)show;
```

参数：设置在个人中心入口是否显示删除账号入口。默认显示如果需要隐藏需要在调用 login 前调用此方法并传入 NO。

设置欧盟 DMA 弹框配置—按需选接

```
/** 是否显示欧盟 DMA 弹窗，YES：首次启动是欧盟地区就会显示隐私配置弹框，NO：不显示，默认 YES */
```

```
+ (void)setNeedPrivacyOption:(BOOL)yesOrNo;
```

请求用户评价或评论 APP—按需选接

```
/** 打开评价界面 appid 用于应用内评分界面无法使用时跳转到 AppStore 应用详情页编辑评论可为空，传空则不跳转 */
```

```
+ (void)showAppCommentWithAppID:(NSString *)appid;
```

设置绑定邮箱时是否需要验证码—按需选接

```
/** 设置绑定邮箱时是否需要输入验证码 YES 需要验证码，NO 不需要 默认 YES */
```

```
+ (void)setNeedEmailVerifyCode:(BOOL)yesOrNo;
```

注意：调用此方法需要与 sdk 后台设置搭配使用，客户端跟 sdk 后台均默认需要验证码。

后台设置路径 1: sdk 后台->游戏管理->接入与配置->应用扩展->xxxx 游戏->扩展配置->基础配置->绑定邮箱验证->设置为是/否。

后台设置路径 2: sdk 后台->游戏管理->接入与配置->游戏列表->xxxx 游戏->扩展配置->基础配置->绑定邮箱验证->设置为是/否。

三方登录时获取三方 AccessToken-按需选接

```
/** 获取三方 accesstoken */  
+ (NSString *)getAccessToken;
```

注意：仅当前账号是采用三方登录方式时会返回对应的 accessToken,其他情况将会返回空字符串。

设置自动登录时是否显示切换按钮-按需选接

```
/** 设置自动登录时是否显示切换账号，默认显示 YES*/  
+ (void)setNeedSwitchWhenAutoLogin:(BOOL)yesOrNo;
```

设置个人中心是否显示切换按钮-按需选接

```
/** 设置个人中心界面是否显示切换按钮，默认显示 YES*/  
+ (void)setNeedShowSwitchInUserCenter:(BOOL)show;
```

ATT(AppTrackingTransparency)与 iOS14

1.ATT 权限获取

从 iOS 14 开始，若开发者设置 App Tracking Transparency 向用户申请跟踪授权，在用户授权之前 IDFA 将不可用。

要获取 App Tracking Transparency 权限，请更新您的 Info.plist，添加 NSUserTrackingUsageDescription 字段和自定义文案描述。代码示例：

```
<key>NSUserTrackingUsageDescription</key>  
<string>该标识符将用于向您投放个性化广告</string>
```

文案请根据自己实际需求填写，需要使用 Xcode12.0 及以上版本。

注:文案应使用与应用本地化语言一致的语言。

****游戏不需要处理 Att 权限，SDK 已内置获取权限方法，将在 UIApplicationDidBecomeActiveNotification 通知回调中申请 Att 权限。****

2.SKAdNetwork

SKAdNetwork 基础结构 (Apple 的 iOS 的一部分) 可帮助广告商在保持用户隐私(即用户不同意 ATT 权限申请)的同时衡量广告系列的成功程度。SKAdNetwork 基础结构无需 IDFA 或其他广告 ID 即可运行。因此，无需用户同意即可实施该解决方案。

开发人员无需操作，SDK 会自动调用必要的 SKAdNetwork API，即 registerAppForAdNetworkAttribution()和 updateConversionValue()。

隐私清单

1.相关说明

根据苹果公司公布的最新 [App Store 隐私政策](#)，自 2024 年春季开始，上架 App Store 的应用需要携带一份 App 的隐私清单文件。

从 2024 年 5 月 1 日开始，App Store Connect 不接受未在隐私清单文件中描述其使用所需原因 API 的应用程序。

2.适配处理

请根据接入工程实际情况选择对应方案进行处理

1.接入工程中不存在 PrivacyInfo.xcprivacy 文件

1.1 打开接入工程通过 Xcode->New File->Resource->App Privacy 新建文件，命名为 PrivacyInfo，同时勾选需要的 Targets

1.2 在接入工程目录中选中 PrivacyInfo.xcprivacy 文件

1.3 将 SDK 的 PrivacyInfo.xcprivacy 中的内容填充到工程目录

PrivacyInfo.xcprivacy 文件中(可右键 Open As->Source Code 的方式打开进行快速粘贴复制)。

2.接入工程中已存在 PrivacyInfo.xcprivacy 文件

2.1 在接入工程中选中 PrivacyInfo.xcprivacy 文件

2.2 补充 SDK 的 PrivacyInfo.xcprivacy 有提及但接入工程的

PrivacyInfo.xcprivacy 文件中仍缺失的内容到接入工程中的 PrivacyInfo.xcprivacy 文件。

配置请求 url –按需选接

找到 SDK 内 SSBundle.bundle 文件下 ColorStyle.plist 新增/修改键值对(键为 mainurl 值为请求的 url)。此方法可修改 SDK 请求的 api 域名。

配置是否启动弹出隐私协议和服务协议 –按需选接

找到 SDK 内 SSBundle.bundle 文件下 ColorStyle.plist 新增/修改键值对(键为 privacyPolicyTip 值为 1 弹出 0 不弹出)。配置为 1 则游戏调用登录时弹出隐私协议和服务协议弹窗，用户必须同意隐私协议和服务协议才可以继续游戏，配置为 0 则不弹出，用户正常进入游戏。

配置登录首页是否展示隐私协议和服务协议 –按需选接

找到 SDK 内 SSBundle.bundle 文件下 ColorStyle.plist 新增/修改键值对(键为 hasLoginAgreement 值为 1 展示 0 不展示)。配置为 1 则登录界面显示服务协议和隐私协议入口，用户必须勾选表示同意服务协议和隐私协议才可以继续下一

步操作，同时注册界面不显示服务协议和隐私协议入口。配置为 0 则登录界面不显示服务协议和隐私协议入口，注册界面显示服务协议和隐私协议入口，用户必须勾选表示同意用户协议和隐私协议方可注册。