

# iOS SDK Core document

## SDK Structure

Please refer to the documentation for access. If you have any questions, please refer to the demo. The demo shall prevail.

1. JYouLoginKit.framework and SSBundle.bundle are the core SDK framework and resources.

Supported platforms: iPod Touch, iPhone, iPad. System requirements: iOS12.0+,

Support frame: arm64

Environmental requirements: Xcode15.3+.

## Precautions

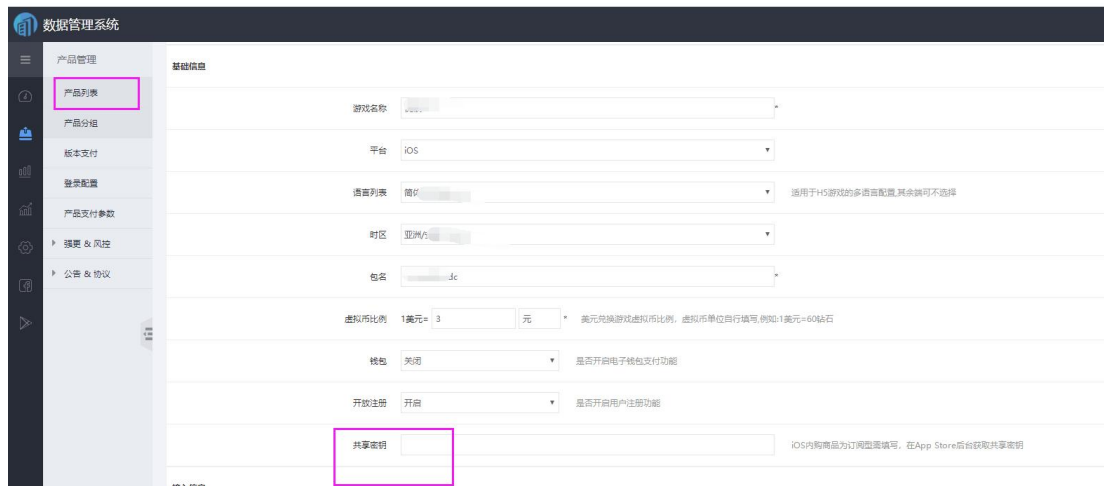
When the game supports the purchase of automatic subscription products or needs to detect Apple's refund, the following configuration is required:

Configure the server notification URL in the Apple background, the URL format: xxx/notify/apple. Where xxx is the full domain name of the manufacturer's sdk, and https must be included. The configuration example is as follows:

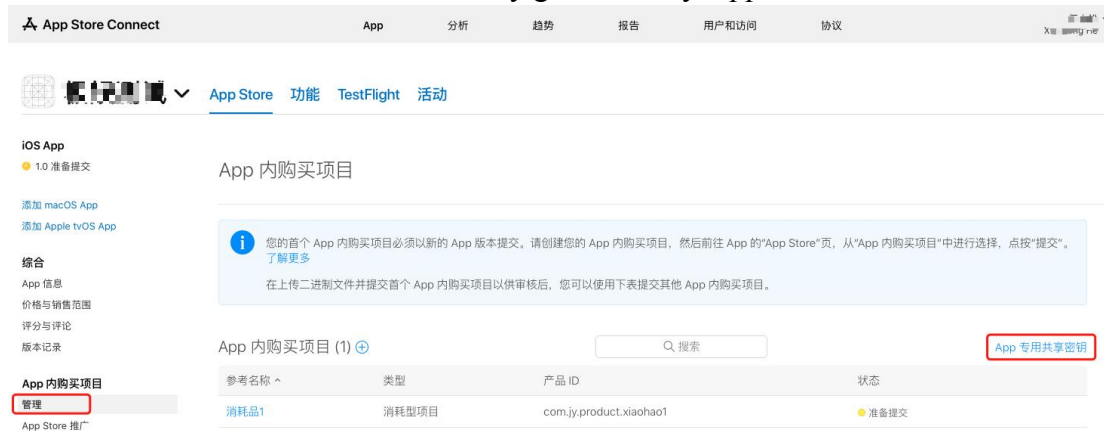
Configuration path: My app->specific app-> tap App information (under the sidebar synthesis)->App store server notification (scroll down to see it).



In addition, to support the purchase of automatic subscription products in the game, you need to configure the Apple backend shared key in the quickgame backend product details, as shown below:



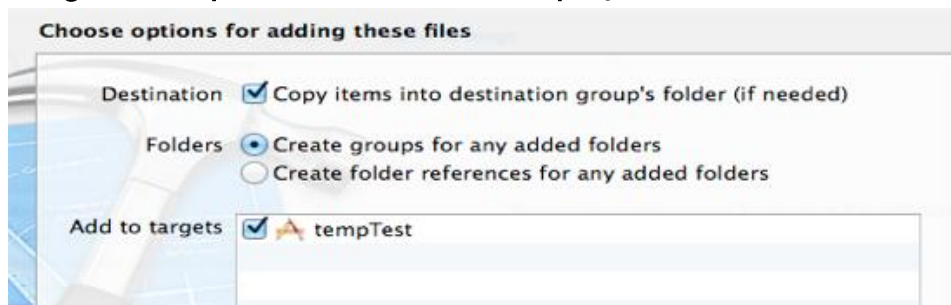
Get the entrance to the shared secret key generated by Apple's backend:



## Access steps

### Add SDK

Drag and drop the SDK file into the project, select the correct target



**For the xcode project exported using Unity version 2019.3 and later, it contains the UnityFramework dynamic library. When importing the SDK file, you need to pay attention to:**

.a/.framework and other static libraries TargetMembership need to be associated with UnityFramework;

Resource files such as .bundle, TargetMembership, need to be linked to Unity-iPhone;

The .framework dynamic library TargetMembership needs to be linked to UnityFramework and Unity-iPhone at the same time.

JYouLoginKit.framework is a static library, TargetMembership needs to be linked to UnityFramework;

SSBundle.bundle resource file, TargetMembership needs to be linked to Unity-iPhone.

As shown below



## Engineering configuration

Configure in Build Settings->Linking->Other Linking Flags --ObjC



## SDK permission application

When supporting data statistics, such as Appsflyer, Adjust, FB and other data statistics SDK, you need to add ATT permission, and ask the user to enable tracking permission to track or access the advertising identifier (ie IDFA) of their device.

## ATT permissions

The access project info.plist file is configured with the following permissions, and the copy on the right needs to be customized and adapted to multiple languages by the accessing party.

Key	Type	Value
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)

## camera and album permissions

The SDK also uses camera and album permissions in the user center avatar selection. The following permissions need to be configured in the access project info.plist file. The copy on the right needs to be customized by the accessing party (the application for permissions needs to be clearly explained) and has multiple adaptations. language.

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

## Interface call

```
import REDeLoginKit.h
```

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

## Application jump callback (required)

Class: [REDeLoginKit](#)

Function: Process the callback result of the third party application

```
+ (void)application:(UIApplication *)application openURL:(NSURL *)url options:(NSDictionary *)options;
+ (void)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceAnnotation:(id)annotation;
```

The above methods need to be called in the following system callback methods:

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplicationAnnotation:(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;
}
```

## SDK initialization (required)

```
// Initialization Set the product, the first interface that needs
to be called (required) Either initialize with setting the
default language.
+ (void)initSDKWithProductCode:(NSString *)productCode
callback:(id<REDeInitCallback>)initDelegate;
/** Initialization set the product, the first need to call the interface (required)
and either initialize without setting the default language..
 * @param productCode sdk parameter from the background to get
 * @param language sdk display language, (Simplified Chinese @"zh-CN" or
"zh_Hans", Traditional Chinese @"zh-hk" or "zh-Hant", German @"de",
English @"en-us", French @"fr", Japanese @"ja", Korean @"ko", "ja", "ja",
"ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja", "ja").
Korean @ "ko", Russian @ "ru", Thai @ "th", Indonesian @ "id", Vietnamese
@ "vi", Turkish @ "tr", Arabic @ "ar",)
 * @param initDelegate Initializes the callback receiver object.
 */
+ (void)initSDKWithProductCode:(NSString *)productCode
language:(NSString *)language
callback:(id<REDeInitCallback>)initDelegate;
Function: Use the product id to initialize the SDK.
```

Parameters: productCode, product id, **required**, to be provided by business.

Callback method for successful initialization:

// SDK initialization success callback

```
@protocol REDeInitCallback <NSObject>
//initialization success
- (void)qqSDKInitDone;
@end
```

## Set whether automatic login is required (optional)

```
/** Set whether to auto-login, default YES */
+ (void)setNeedAutoLogin:(BOOL)autoLogin;
```

Function: Whether to enable automatic login to the last login account, it needs to be called before calling login. It is turned on by default.

## Set whether to enable guest login (optional)

```
/** Whether to enable Guest login, YES enable NO disable
Default YES
 */
+ (void)guestLoginEnable:(BOOL)yesOrNo;
```

### Set whether to pop-up tips for guest login (optional)

```
/** Whether to enable vsitor binding prompt YES:Disable NO  
Enable Default NO  
*/  
+ (void)guestBingTipDisable:(BOOL)yesOrNo;
```

Function: Set a pop-up tips for guest login or not  
, it needs to be called before calling login. It is default NO.

### Set whether to show uid on the right side of user center avatar

(optional)

```
/** whether to show uid on the right side of usercenter  
avatar,default NO:not show */  
+ (void)setNeedShowUid:(BOOL)show;
```

Function: whether to show uid on the right side of usercenter avatar, YES:show,NO:  
not show.default NO:not show.

### Set login and logout callback listeners (required)

```
// Set login callback listener object (required)  
+  
(void)setFunctionLoginCallback:(id<REDeLoginCallback>)loginDelega  
te;
```

Function: Set the login callback listener and then the listener can receive the login  
event by implementing the following callback method

Required:

```
@protocol REDeLoginCallback <NSObject>  
@required  
//will be call after the user clicks on the personal center of  
the SDK to switch accounts  
- (void)userLogout;  
//will be called after a successful binding  
- (void)bindUid:(NSString *)uid userToken:(NSString *)token  
type:(USERCENTER_TYPE)type;  
//will be called after a successful unbinding  
- (void)unBindUid:(NSString *)uid userToken:(NSString *)token  
type:(USERCENTER_TYPE)type;
```

tips:

type

```
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
```

Optional:

```
@optional  
//will be call after successful login without Login Method
```

```

//This method or loginUid:userToken:type: must be implemented
- (void)loginUid:(NSString *)uid userToken:(NSString *)token;
// will be call after successful login with Login Method
//This method or loginUid:userToken: must be implemented
- (void)loginUid:(NSString *)uid userToken:(NSString *)token
type:(USERCENTER_TYPE)type;
//will be call after the logout call which is different from the
active logout callback of the user's personal center in the SDK
- (void)gameLogoutSuccess;
//will be call when user clicks on the User Center customer
service
- (void)onClickServiceCenter;
/** The game actively calls the SDK page after login. This method will be
called back when closing
*/
- (void)sdkUserPageWillClose;
/** Player cancel login, mainly used to call up a third sdk login
method individually.
isShow: YES: the SDK login page is shown when the player cancel
the third SDK login, NO: the SDK login page is not shown when the
player cancel the third SDK login.
*/
- (void)userCancelLoginWithLoginPageShowing:(BOOL)isShow;
/** Failed to log in, mainly used to call up a third sdk login method
individually.
isShow: YES: the SDK login page is shown when the player failed to
login, NO: the SDK login page is not shown when the player failed to
login.
*/
message: reason for failure message: 失败原因
*/
- (void)userLoginFailWithLoginPageShowing:(BOOL)isShow
message:(NSString *)message;
/** Players cancel binding, mainly used to call a three-party
binding method separately
isShow: YES: Players cancel three-party login when the SDK interface
is displayed, NO: Players cancel three-party login when the SDK
interface is not displayed
*/
-
- (void)userCancelBindWithKitWindowShowing:(BOOL)isShow;
/** Players fail to bind, mainly used to call a three-party binding
method separately
isShow: YES: Players fail to log in when the SDK interface is
displayed, NO: Players fail to log in when the SDK interface is not
displayed
message: Failure reason
*/
- (void)userBindFailWithKitWindowShowing:(BOOL)isShow
message:(NSString *)message;
/** Players cancel unbinding, mainly used to call a three-party
binding method separately
isShow: YES: Players cancel three-party login when the SDK interface
is displayed, NO: Players cancel three-party login when the SDK
interface is not displayed
*/

```



```

-
- (void)userCancelUnbindWithKitWindowShowing:(BOOL)isShow
OW;
/** Player unbinding failed, mainly used to call a three-party
unbinding method separately
isShow: YES: Player failed to log in when the SDK interface is
displayed, NO: Player failed to log in when the SDK interface is not
displayed
message: Reason for failure
*/-
- (void)userUnbindFailWithKitWindowShowing:(BOOL)isShow
message:(NSString *)message;
/** Login process event callback.
loginEvent: event enumeration value Enumeration value definition
can be found in the header file REDeDelegate.h.
message: failed event returns the reason for the failure, others
return the enumerated value string.
*/
- (void)onEvent:(LoginEvent)loginEvent message:(NSString *)message;

```

Get whether the last login account information exists -Optional

```

/** Get whether the last login account information exists. YES:
exists, NO: does not exist */
+ (BOOL)checkLoginValid;

```

Function: Get whether there is the last login information. If it exists, return YES, otherwise return NO.

If YES is returned, calling quick login will log in the last logged-in account.

Login (required, this maybe ignore if use fastlyStartGame)

```

/**Enter the user login page. Called after receiving the
notification of switching users or actively logging out the user
(required)
@method isShowMenu YES:show NO:not show
*/
+ (void)loginWithMenuShow:(BOOL)isShowMenu;

```

Function: Enter the user login page. It needs to be called after the initialization is successful, otherwise the call is invalid. When the game starts, you can call this interface to display a user login interface. After receiving the user logout callback, the usual solution is to call this interface after returning to the game login interface.

isDisplay: YES: Automatically display the buoy after login NO: Do not automatically display the buoy after login.

Quick login (silent login)-select on-demand

```

// Automatic registration and automatic login process, the player
does not need to register for the first game, to achieve the

```



```
purpose of fast game, suitable for games without a login button
to be called after startup
+ (void)fastlyStartGame;
```

Function: Log in silently without displaying the login interface. If there is no account locally, a new guest account will be created based on this device. You can bind and unbind operations and switch accounts in the user center. If the SDK has not been initialized successfully when this interface is called, it will automatically call this interface once when the SDK is initialized successfully. This interface can be used to quickly enter the game. After the user logs out, log in again to use the login interface.

### Quickly invoke login methods other than password login -select on-demand

```
// Unsolicited call to log in other than password login, no SDK
login screen
+ (void)loginAccountType:(USERCENTER_TYPE)type;
```

Function: Do not display the SDK login interface, and evoke a certain login method other than the account password login alone. Include visitor login and other three-party login.

### Obtain all binding information- Optional

```
//Get channel binding information
+ (NSDictionary *)getUserBindInfo;
```

Function: Obtain the binding information of the user center.

key	Value Type	Key Annotation
USERCENTER_TYPE_EMAIL	NSNumber	Email
USERCENTER_TYPE_FACEBOOK	NSNumber	FaceBook
USERCENTER_TYPE_GOOGLEPLUS	NSNumber	Google
USERCENTER_TYPE_LINE	NSNumber	Line
USERCENTER_TYPE_GAMECENTER	NSNumber	GameCenter

key	Value Type	Key Annotation
USERCENTER_TYPE_APPLE	NSNumber	Apple

Code example:

```
NSMutableDictionary *dic = [REDeLoginKit getUserBindInfo];
if ([dic[@"USERCENTER_TYPE_EMAIL"] boolValue]) {
    NSLog(@"bind the email");
}
```

### Set role information (required)

```
//required:Called when entering the game or when character
information changes
+ (void)setGameRoleInfo:(REDeRoleInfo *)roleInfo;
```

Function: Set the role information, which is called when the role is selected to enter the game or when character information changes.

### Actively call the bound account (optional)

```
//call to bind the account in some way
+ (void)bindAccountType:(USERCENTER_TYPE)type;
```

Function: Actively call the bound account externally

### Get user ID (optional)

```
/**
 * @method userID
 * @return return userID. Not login will be nil
 * Usually called in the login success callback
 */
+ (NSString *)userID;
```

Function: Get the user ID of the currently logged in user. Called after the user has successfully logged in

Return value: the user ID of the currently logged in user, if not logged in, nil will be returned.

### Get the token after login (optional)

```
// The user verification code is used to verify the authenticity
of the user on the server side.
+ (NSString *)getUserToken;
```

Function: Get the user token of the currently logged in user. Called when the user logs in successfully

Return value: The user token of the currently logged-in user, used to verify the validity of the user, and returns nil if not logged in.

### Get whether the current user is a guest account(optional)

```
// Judge whether it is a guest account, YES is a guest account, NO is not a guest account.  
+ (BOOL)isNewUser;
```

Function: Get whether the current user is a guest account

Return value: YES is a guest account, No is not a guest account.

### Get whether the current user is a new user (optional)

```
// Judge whether it is a new user, YES is a new user, NO is a registered user  
+ (BOOL)isNewUser;
```

Function: Get whether the current user is a new user

Return value: YES is a new user, No is a registered user.

### Get Device ID(optional)

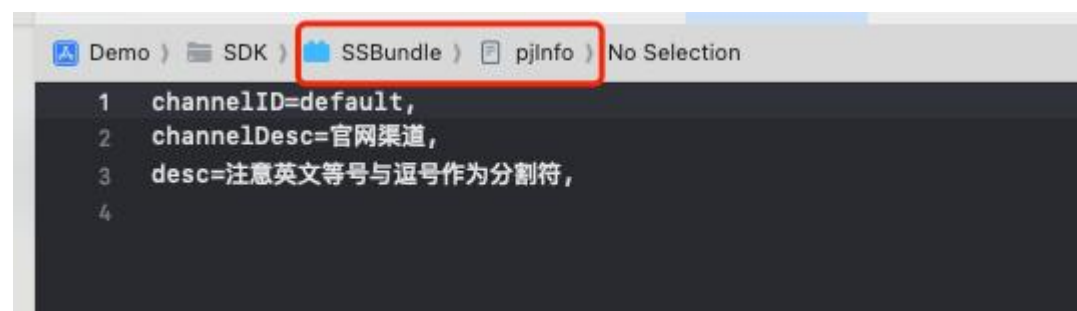
```
// get sdk device id  
+ (NSString *)getDeviceID;
```

Function: get current device id

### Configure Channel Code (optional)

Find the pjInfo.txt file under the SSBundle.bundle file in the SDK and modify the channelId and channelDesc.

Note that you must add the English ‘,’ as a separator at the end.



## Get Channel Code (optional)

```
// get channel code:default:default  
+ (NSString *)channelCode;.
```

Function: get channel code

## Get Countrycode(optional)

```
// get country code  
. + (NSDictionary *)getNationCode;
```

Function: Get the country code of the current device setting area

Returns an example of results:

```
Country Code: {  
    countryCode = JP;  
    ip = "xxx.xxx.xxx.xxx";  
    ipcountryCode = CN;  
}
```

countryCode = JP; //country code set by the device system

ip = "xxx.xxx.xxx.xxx"; //device network ip address

ipcountryCode = CN; //country code obtained from ip

## Log out (required)

```
//@method    logout  
+ (void)logout;
```

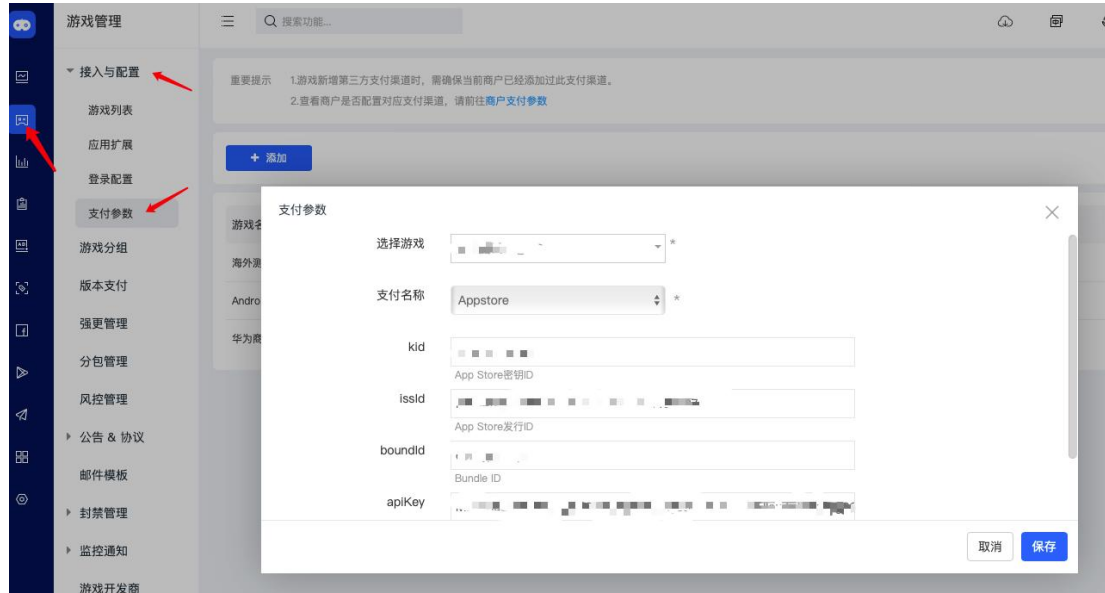
Function: Log out the current user immediately

## Get Storefront CountryCode (Optional)

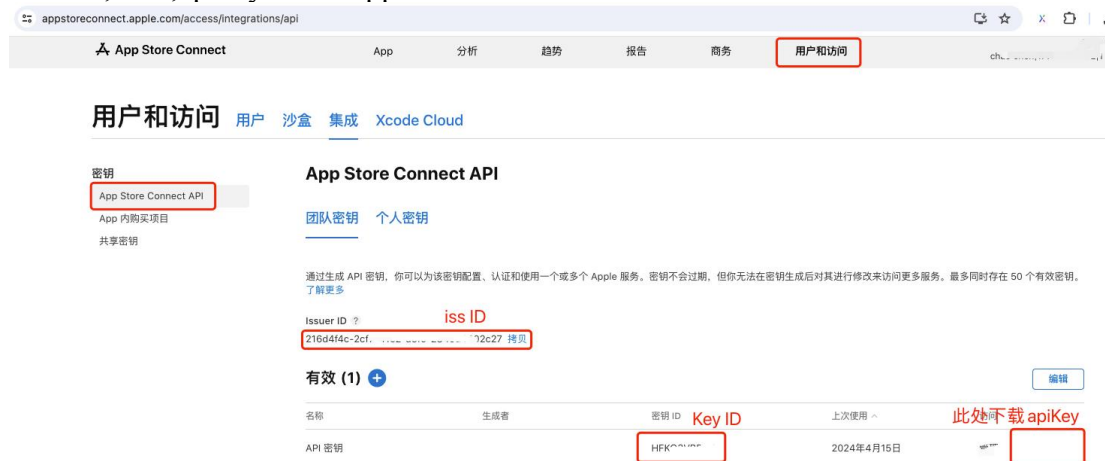
```
/** Retrieve Storefront country codes synchronously  
*/  
+ (NSString *)getStorefrontCountryCode;  
/** Asynchronously retrieve Storefront country code  
*/  
+  
(void)getStorefrontCountryCodeWithCompletion:(void(^)(NSString *countryCode))completion;  
/** Set Storefront country code update block callback  
*/  
+  
(void)configStorefrontCountryCodeUpdatedBlock:(void(^)(NSString *countryCode))updateBlock;
```

## Payment (required)

SDK has enabled Storekit2 (Only supports iOS15.0+, below iOS15.0 use Storekit1) since 2.0.4.3, must configure kid, issid, apikey in SDK service backend (Game Management->Access & Configuration->Payment Parameters) first.



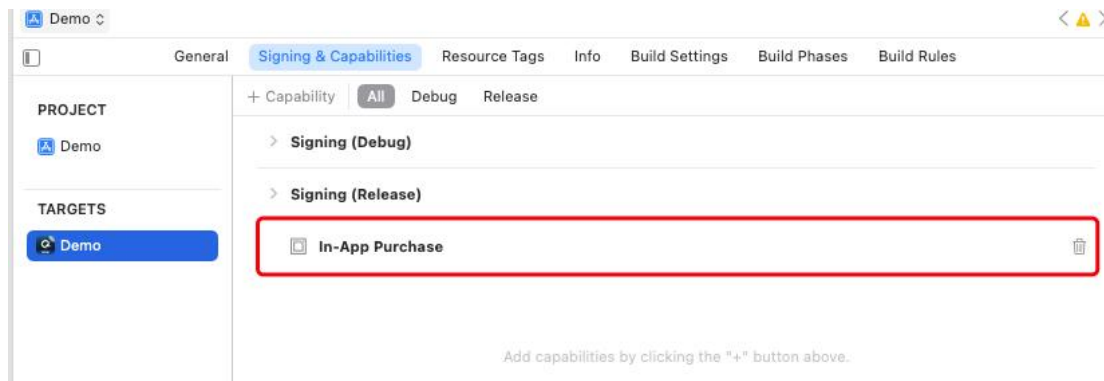
Get kid,issid,apikey from App Store Connect:



## Creating API keys to authorize API requests

## Xcode Project Configuration

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



```
// Pay(required)
+ (void)IAPWithParameter:(REDeOrderInfo *)param;
```

Function: Enter the recharge module.

Parameter: param recharge parameter information, refer to REDeOrderInfo.h file.

Note: Do not pass special symbols in the extension field. If it is unavoidable, it is recommended to encode base64 before sending it; the callback address can be configured in the background, and the callback address is configured in the background. Whichever is configured

Note: Asynchronous recharge, the result of the SDK notification is for reference only, subject to server-side synchronization.

ReDeOrderInfo model property:

```
@property (copy, nonatomic) NSString
*productId; //required Product ID in developer
backend, productId
@property (copy, nonatomic)
NSString*product_order_no; //required Game product order
number
@property (copy, nonatomic) NSString
*subject; //required product name
@property (copy, nonatomic) NSString
*desc; //OptionalProduct description
@property (copy, nonatomic) NSString
*price; //OptionalCommodity price
@property (assign, nonatomic) unsigned int
quantity; //Optionalpurchase quantity default 1
@property (copy, nonatomic) NSString
*total; //required Total price of the item
Mandatory, affects purchase statistics and callbacks
@property (copy, nonatomic) NSString
*callback_url; //Optional callback url string[200]
optional, (can be configured in the background, if the
```

```

background is configured with the callback url, the one
configured in the background will prevail)
@property (copy, nonatomic) NSString
*extras_params; //Optional extended parameters
string[500] optional, pass-through fields, Note: Extended
fields do not pass special symbols, if it can not be
avoided it is recommended that the first base64 encoding
and then pass, used to find the product information to
return to the product attributes return currency Symbol

@property (copy, nonatomic) NSString * displayPrice;/**
The display price of the product used to find the product information to return
the product attributes*/
@property (copy, nonatomic) NSString * currencySymbol;/**
Commodity Currency Symbol Used to find information about a product to
return product properties*/
@property (copy, nonatomic) NSString * currencyCode;/**
Commodity Currency Code Used to find commodity information to return
commodity attributes*/
/** Optional field, mainly used to find product information to return product
attributes*/
@property (assign, nonatomic) BOOL isFamilyShareable
API_AVAILABLE(ios(14.0));
/** Commodity type 1 consumable 2 non-consumable 3 auto-renewal
subscription 4 non-renewal subscription Optional field, mainly used by
storekit2 to find commodity information to return commodity properties */
@property (assign, nonatomic) NSInteger productType
API_AVAILABLE(ios(15.0));
/** Whether a subscription is eligible for a discount Optional field, mainly
used by storekit2 to find product information to return product attributes */
@property (assign, nonatomic) BOOL isEligibleForIntroOffer
API_AVAILABLE(ios(15.0));

```

Note:ReDeOrderInfo has new properties displayPrice, currencySymbol, currencyCode, isFamilyShareable, productType, isEligibleForIntroOffer since 2.0.4.3 for (iOS15.0+) when querying product information callback, but not when calling payment interface.

### Set up payment callback listener (required)

```

// Set payment callback listener object (required)
+ (void)setFunctionBuyCallback:(id<REDeBuyCallback>)buyDelegate;

```

Function: Set the payment callback listener and then the listener can receive the payment event by implementing the following callback method:

```

@protocol REDeBuyCallback <NSObject>
@optional

```



```

#pragma mark - Either of the two purchase failure callbacks can
be used to implement the callback
/* Purchase failed, no return parameters */
*/
- (void)purchaseFail;
/* Purchase failed, with return parameters
productId In-app purchase product ID
orderNo SDK order number
gameOrderNo Game order number, an empty string will be returned
if the local cache is cleared
errorCode Error code 2: User cancels purchase -1: SDK error -2:
In-app purchase is not enabled on the device -3: Product
information cannot be found 0: Unknown error code Other servers
return errors or Apple returns error codes
message Error description
*/
- (void)purchaseFailWithProductId:(NSString
*)productId gameOrderNo:(NSString *)gameOrderNo
errorCode:(NSInteger)errorCode message:(NSString
*)message;
#pragma mark - just implement a callback to any of the four
purchase success callbacks
//callback of successful payment, The success here cannot be used
as a basis for shipment
//
    pay successs callback
    productId :IAP productid
    orderNo:SDK order number
    gameOrderNo:game order number
transactionIdentifier: transactionIdentifier
receiptString:appStoreReceiptBase64EncodedString
//this or purchaseDoneProductId:orderNo:gameOrderNo: must be
achieved
iOS15.0 and above with storekit2
appStoreReceiptBase64EncodedString as @"
- (void)purchaseDoneProductId:(NSString *)productId
orderNo:(NSString *)orderNo gameOrderNo:(NSString
*)gameOrderNo transactionIdentifier:(NSString
*)transactionIdentifier
appStoreReceiptBase64EncodedString:(NSString
*)receiptString;
//callback of successful payment, The success here cannot be used
as a basis for shipment
//
    pay successs callback
    productId :IAP productid
    orderNo:SDK order number
    gameOrderNo:game order number
- (void)purchaseDoneProductId:(NSString *)productId orderNo:(NSString
*)orderNo gameOrderNo:(NSString *)gameOrderNo;
//callback of successful payment, The success here cannot be used
as a basis for shipment
//pay successs callback
    productId :IAP productid

```

```

    orderNo:SDK order number
    gameOrderNo:game order number
    receiptString:appStoreReceiptBase64EncodedString
    //this or purchaseDoneProductId:orderNo:gameOrderNo: must be
    achieved
    iOS15.0 and above with storekit2
    appStoreReceiptBase64EncodedString as @""
- (void)purchaseDoneProductId:(NSString *)productId orderNo:(NSString
*)orderNo gameOrderNo:(NSString *)gameOrderNo
appStoreReceiptBase64EncodedString:(NSString *)receiptString;
/* The success callback here is not a shipping basis.
   Purchase completion callback
   productId The product Id of the purchased product.
   orderNo SDK order number
*/
- (void)purchaseDoneProductId:(NSString *)productId
orderNo:(NSString *)orderNo;

/** iOS15.0 and above User submits refund request successfully
*/
- (void)storeKit2RequestRefundSuccess:(NSString *)transactionID;
/** iOS15.0 and above User cancels submission of refund requests
*/
- (void)storeKit2CancelRequestRefund:(NSString *)transactionID;
/** iOS15.0 and above User failed to submit refund request
*/
- (void)storeKit2RequestRefundFail:(NSString *)transactionID
message:(NSString *)message;
@end

```

### Resumption of purchase of non-consumables (optional)

```

// Get purchased non-consumable products or subscription products,
and the product information is returned through callbacks
(usually the game itself can also obtain the information of these
products, and it supports the option of purchasing automatic
subscription products or non-consumable products)
+
(void)restoreNonConsumptionProducts:(id<REDeRestoreCallback>)rest
oreDelegate;

```

Function: Get the purchased non-consumable products or subscription products, and the product information is returned through callbacks. The following callback methods need to be implemented:

```

@protocol REDeRestoreCallback <NSObject>
//Successfully restore non-consumable products, return product id
information
- (void)restoreSuccess:(NSArray *)products;
//restore fail
- (void)restoreFail:(NSString *)msg;
@end

```

## Get product information (optional)

```
//Get the product information according to the incoming product id list, and the result will be returned through the callback  
+ (void)findProductInfoWithProductIds:(NSArray *)productArr  
delegate:(id<REProductInfoCallback>)productDelegate;
```

Function: Obtain product information according to the incoming product id list, and return the product information through callbacks.

Parameters: an array of product IDs, the product ID type is a string.

## Get All configured product information (optional)

```
//Get product information through the product ID list configured in the SDK background, and return the result through callback  
+ (void)findProductInfoWithDelegate:(id<REProductInfoCallback>)productDelegate;
```

Function: Obtain product information according to the incoming product id list, and return the product information through callbacks.

Parameters: an array of product IDs, the product ID type is a string.

```
@protocol REProductInfoCallback <NSObject>  
//Find the product information successfully, the array element is REDeOrderInfo instance  
- (void)findProductInfoSuccess:(NSArray *)products;  
// Callback for failure to find product information, When the product information is successfully inquired, this method will be called back. The callback parameter type is a string to clarify the reason for the failure  
- (void)findProductInfoFail:(NSString *)msg;  
@end
```

Function: When the product information is successfully queried, this method will be called back. The callback parameter type is an array and the elements are instances of REDeOrderInfo. Examples are as follows:



showManageSubscriptions (optional)

```
/** showManageSubscriptions*/
+ (void)showAllSubscriptionsUI API_AVAILABLE(ios(15.0));
```

beginRefundRequest (optional)

```
/** request Refund transactionID:iap transactionID */
+ (void)showRefundWithTransactionID:(NSString *)transactionID
API_AVAILABLE(ios(15.0));
callback:ReDeBuyCallback
- (void)purchaseDoneProductId:(NSString *)productId orderNo:(NSString *)orderNo gameOrderNo:(NSString *)gameOrderNo
appStoreReceiptBase64EncodedString:(NSString *)receiptString;
/** iOS15.0 and above User submits refund request successfully */
- (void)storeKit2RequestRefundSuccess:(NSString *)transactionID;
/** iOS15.0 and above User cancels submission of refund requests */
- (void)storeKit2CancelRequestRefund:(NSString *)transactionID;
/** iOS15.0 and above User failed to submit refund request */
- (void)storeKit2RequestRefundFail:(NSString *)transactionID
message:(NSString *)message;
```

presentOfferCodeRedeemSheet (optional)

```
/** presentOfferCodeRedeemSheet */
+ (void)showOfferCodeRedeemSheetUI API_AVAILABLE(ios(16.0));
```

Enter user center (optional)

```
// enter user center
+ (void)enterUserCenter;
```

Function: Enter the user center page to display information and operation portals associated with other platform accounts.

### Dismiss user center (optional)

```
// dismissUserCenter  
+ (void)dismissUserCenter;
```

Function: hide personal center page.

### Show floating menu (optional)

```
/** Show buoy, isLeft buoy left or not originalY buoy vertical  
position start point */  
+ (void)showFloatButtonIsLeft:(BOOL)isLeft  
buttonOriginalY:(CGFloat)originalY;
```

Function: Display the floating menu in the game window.

### Hide floating menu (optional)

```
// Hide floating menu  
+ (void)dismissMenu;
```

Function: Hide the floating menu displayed in the game window.

Enter the customer service center—if there is no agreement during deployment, please ignore this method, otherwise select as needed

```
/** Enter the customer service center */  
+ (void)enterServiceCenter;
```

Function: Arouse the customer service interface, which is convenient for users to contact customer service personnel.

### Account deletion

```
/** Set the text content that prompts the user to delete the  
account data. If nil or an empty string is passed, the SDK  
default text will be used. */  
+ (void)configAccountDeletionTipContent:(NSString *)tipContent;  
Parameters: tipcontent the text content that prompts the user to delete the account  
data. If nil or an empty string is passed, the SDK default text will be used. If you need  
to use custom text, you need to call this method before calling accountDeletion.
```

```
/** Delete account,user confirmation required*/  
+ (void)accountDeletion;
```

Function: The game developer will call this SDK method to pop up the confirmation of account deletion interface, after the user clicks on the confirmation, the account deletion logic will be executed.

The personal center page of clicking the buoy also has an account logout entry, which has the same function as calling this method directly.

```
/** Delete account,no need to confirm by user*/  
+ (void)accountDeletionWithoutConfirm;
```

Function: Game developers call this SDK method to execute account deletion logic immediately without user confirmation.

```
/** Set whether to show account deletion in UserCentre  
interface, default display YES */  
(void)setNeedShowAccountDeletionInUserCenter:(BOOL)show
```

Parameters: set whether to show account deletion in User Centre. Default show if need to hide need to call this method before calling login and pass NO.

### Set DMA Privacy Option (optional)

```
/** Whether to display the EU DMA popup, YES: the privacy configuration  
popup will be displayed if the first startup is in the EU, NO: not displayed  
default YES */  
+ (void)setNeedPrivacyOption:(BOOL)yesOrNo;
```

### Ask the user to rate or review your app (optional)

```
/** Open the comment interface appid is used to jump to  
the AppStore app details page to edit the comment when  
the in-app rating interface is unavailable can be null,  
pass null then no jump */  
+ (void)showAppCommentWithAppID:(NSString *)appid;
```

### Set whether verification code is required when binding email (optional)

```
/** Set whether verification code is required when binding email YES requires  
verification code, NO does not require it Default is YES */  
+ (void)setNeedEmailVerifyCode:(BOOL)yesOrNo;
```

Note: This method needs to be used with the sdk background settings, both the client and the sdk background need to verify the code by default.

Background Setting Path 1: sdk background->Game Manage->Configuration->Game Conf->xxxxGame->Ext Configure->BaseSet->Binding Email Verification->Set to Yes/No.

Background setting path 2: sdk background->Game Manage->Configuration->More Conf->xxxxGame->Ext Configure-> BaseSet->Binding Email Authentication->Set to Yes/No.

### Get third SDK AccessToken for three-party login (optional)

```
/** Get three-way accesstoken */  
+ (NSString *)getAccessToken;
```

Note: Only when the current account is using three-way login will return the corresponding accessToken, otherwise it will return an empty string.

### Set whether to display the switch account button when logging in automatically (optional)

```
/** Set whether to show switch account button when auto login, default is YES */.+  
(void)setNeedSwitchWhenAutoLogin:(BOOL)yesOrNo;
```

### Set whether to display the switch account button in the User Center (optional)

```
/** Set whether the switch button is displayed in the Personal Center interface, the default display is YES */.+  
(void)setNeedShowSwitchInUserCenter:(BOOL)show;
```

## ATT(AppTrackingTransparency) And iOS14

### 1.ATT Permission acquisition

Starting from iOS 14, if the developer sets App Tracking Transparency to apply for tracking authorization from the user, IDFA will not be available until the user authorizes it.

To obtain App Tracking Transparency permissions, please update your Info.plist to add the NSUserTrackingUsageDescription field and custom text description. Code example:

```
<key>NSUserTrackingUsageDescription</key>  
<string>This identifier will be used to serve you personalized ads</string>
```

Please fill in the copy according to your actual needs, and you need to use Xcode 12.0 and above.

Note: The copy should be in a language consistent with the localized language of the application.

**\*\* The game does not need to handle the Att permission, the SDK has built-in method to obtain the permission, and will apply for the Att permission in the UIApplicationDidBecomeActiveNotification notification callback \*\***

### 2.SKAdNetwork

The SKAdNetwork infrastructure (part of Apple's iOS) can help advertisers measure the success of their campaigns while maintaining user privacy (that is, users disagreeing with ATT permission applications). The SKAdNetwork infrastructure does not require IDFA or other advertising IDs to operate. Therefore, the solution can be implemented without user consent.

Developers do not need to operate, the SDK will automatically call the necessary SKAdNetwork API, namely registerAppForAdNetworkAttribution() and updateConversionValue().



## Privacy List

### 1. Related notes

According to the latest App Store Privacy Policy published by Apple, starting in Spring 2024, apps hitting the App Store will need to carry a copy of the app's Privacy Inventory file. Starting May 1, 2024, App Store Connect will not accept apps that do not describe their use of the Required Reasons API in a Privacy Inventory document.

### 2. Adaptation Processing

Please select the corresponding program according to the actual situation of the access project.

1. PrivacyInfo.xcprivacy file does not exist in the access project.
  - 1.1 Open the access project and create a new file through Xcode->New File->Resource->App Privacy, name it PrivacyInfo and check the required Targets.
  - 1.2 Select the PrivacyInfo.xcprivacy file in the access project directory.
  - 1.3 Fill the contents of PrivacyInfo.xcprivacy in SDK into the PrivacyInfo.xcprivacy file in the project directory (you can right-click Open As->Source Code to open it for quick paste and copy).
2. Accessing the existing PrivacyInfo.xcprivacy file in the project.
  - 2.1 Select the PrivacyInfo.xcprivacy file in the access project.
  - 2.2 Supplement the contents mentioned in the PrivacyInfo.xcprivacy of SDK but still missing in the PrivacyInfo.xcprivacy file of the access project to the PrivacyInfo.xcprivacy file of the access project.

### Configure SDK request url-select on-demand

Please find the ColorStyle.plist file under the SSBundle.bundle file in the SDK folder, and then modify the default value default corresponding to the mainurl key to the SDK domain name).

### Configure whether to activate the pop-up privacy agreement and service agreement – select on demand

Please find the ColorStyle.plist file under the SSBundle.bundle file in the SDK folder(the key is privacyPolicyTip, the value is 1 will pop up, and 0 will not pop up). If it is set to 1, the privacy agreement and service agreement pop-up window will pop up when the game calls to log in. The user must agree to the privacy agreement and the service agreement to continue the game. If it is set to 0, it will not pop up, and the user will enter the game normally.

### Configure whether loginpage show privacy agreement and service agreement – select on demand

Please find the ColorStyle.plist under the SSBundle.bundle file in the SDK and add/modify the key-value pair (the key is hasLoginAgreement with a value of 1 to show 0 without display). Configured to 1, the login screen displays the service agreement and privacy agreement portal, the user must check the box to indicate agreement to the service agreement and privacy agreement before proceeding to the

next step, while the registration screen does not display the service agreement and privacy agreement portal. Configured to 0, the login screen does not display the Service Agreement and Privacy Agreement portal, the registration screen displays the Service Agreement and Privacy Agreement portal, and the user must check the box to agree to the User Agreement and Privacy Agreement before registering.