

# iOS SDK Document for Facebook Data

## Import SDK files

/\*\*

FBSDK13.0.0 and above no longer support Xcode12, you must use xcode13.2.1 or above.

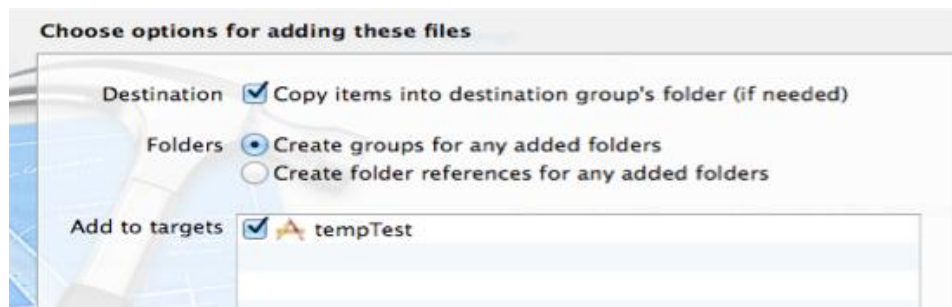
FBSDK15.1.0 and above no longer support Xcode13, you must use xcode14 or above.

FBSDK17.0.0 and above no longer support Xcode14, you must use xcode15.0 or above(FBSDK17.0.2 required xcode15.3+).

\*\*/

Import the FBSDKCoreKit.framework, FBAEMKit.framework, and FBSDKCoreKit\_Basics.framework files in the SDK folder into the access project and select the correct target.

(If you have accessed Facebook to log in or share, you can skip this step.)



FBSDK has changed to dynamic library version since 17.3.0, please load it as dynamic library, i.e. set FBxxxx.framework to Embed & Sign.

Check if the target->Build Setting-> Runpath search path contains @executable\_path/Frameworks, if it doesn't, please add it, if it does, please continue to the next step.

**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:**

FBSDKCoreKit.framework and FBAEMKit.framework and FBSDKCoreKit\_Basics.framework are dynamic libraries, and TargetMembership needs to be linked to UnityFramework and Unity-iPhone at the same time.

UnityFramework only needs to be associated, it doesn't need to be and can't be set to Embed & Sign, otherwise it will report an error when submitting to the AppStore.

Unity-iPhone not only needs to be associated, it needs to be and must be set to Embed & Sign, otherwise it will crash on startup.

## Xcode configuration

FBSDK needs access to engineering support for Swift and OC mashups. You can choose one of the following 2 options

Option 1: You need to create a swift file in your project, and then follow the Xcode prompts to create an OC and swift bridge file.

Option 2: Go to Project->target->BuildSettings -> Search path->Library Search Paths, and add the code:

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

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

Then Project->target->BuildSettings -> Linking -> Runpath search path and add the following code:

```
//needs to be first on the list
```

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

**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:**

All operations are performed on UnityFramework.

Option 1: A swift file needs to be created in the project, which can be associated to UnityFramework only or to both UnityFramework and Untiy-iphone.

Option 2: Go to Project->UnityFramework->BuildSettings -> Search path->Library Search Paths, and add the code:

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

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

Then Project->UnityFramework->BuildSettings -> Linking -> Runpath search path and add the following code:

```
//needs to be first on the list
```

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

FacebookAppID: application parameters in the FB background;

FacebookDisplayNam: The name of the application configured in the FB background {game name};

FacebookClientToken: Check the Facebook background application settings->Advanced->Client password;

LSApplicationQueriesSchemes: Facebook related whitelist.

✓ LSAApplicationQueriesSchemes	◇	Array	(5 items)	
Item 0		String	fbapi	
Item 1		String	fb-messenger-api	白名单
Item 2		String	fbauth2	
Item 3		String	fbshareextension	
Item 4		String	fb-messenger-share-api	
DT SDK Build	◇	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	游戏的fb应用名
FacebookDisplayName	◇	String	{{FacebookDisplayName}}	
DTXcode	◇ + -	String	1310	
FacebookClientToken	◇	String	{{FacebookClientToken}}	游戏的fbclienttoken

The event has been delivered inside the SDK (the access party does not need to process it)

Facebook standard events (events have been posted inside FBSDK):

Event	Details
App Install	The first time a new user activates an app or the first time an app starts on a particular device.
App Launch	When a person launches your app, the Facebook SDK is initialized and the event is logged. However, if a second app launch event occurs within 60 seconds of the first, the second app launch event is not logged.
In-App Purchase	When a purchase processed by the Apple App Store or Google Play has been completed. If you use other payments platforms, you will need to add purchase event code manually.
Facebook SDK Crash Report (For Facebook Use Only.)	If your app crashed due to the Facebook SDK, a crash report is generated and sent to Facebook when your

Event	Details
	app is restarted. This report contains no user data and helps Facebook ensure the quality and stability of the SDK. To opt out of logging this event, <a href="#">disable automatically logged events</a> .

## In-App Purchase Automatically Logged Events

Apple provides four different In-app purchase types: consumable, non-consumable, auto-renewable subscription, and non-renewing subscription. If you implement In-App Purchases with StoreKit 1, we will automatically log each of these In-app purchase types. If you implement In-App Purchases with StoreKit 2, we will automatically log non-consumables, auto-renewable subscriptions, and non-renewing subscriptions. If you would like to also automatically log consumables, you will need to add the [SKIncludeConsumableInAppPurchaseHistory](#) key to your **Info.plist**:

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

## Disable Automatically Logged Events

1. Disable it in FaceBook background application settings (this is recommended).
2. Disable it in the access project

To disable automatic event recording, please add Key in Xcode's info.plist:

**FacebookAutoLogAppEventsEnabled** The corresponding value is set to: **false**

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

If automatic event logging is disabled, and if the game still needs Facebook data statistics, the access party should call the following interface at an appropriate time for event delivery.

The default events recorded by this SDK are:

- 1.Login success: fb\_custom\_login\_user\_name (Custom event) ;
- 2.Register: FBSDKAppEventParameterNameRegistrationMethod (Standard event) .

## Interface call

import REDeLoginKit.h

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

## Delivery tutorial completion event interface

```
/**
 * Statistics for completing the tutorial
 * contentData Feature content
 * contentId Feature content id
 * success Did you complete the tutorial
 * tip:
 *   fb standard RBI event, event name: Complete Tutorial
 */
+ (void)logCompleteTutorialEvent:(NSString *)contentData
                             contentId:(NSString *)contentId
                             success:(BOOL)success;
```

## Post role upgrade event interface

```
/**
 * Role upgrade statistics
 * level Role level
 * tip:
 *   fb standard RBI event, event name: Achieve Level
 *
 *   fb standard RBI event, event name: FB is already connected by
 *   default
 */
+ (void)logPurchase:(double)purchaseAmount
                currency:(NSString *)currency
                parameters:(NSDictionary<NSString *, id> *)parameters;
```

## Delivery achievement unlock event interface

```
/**
 * Achievement unlocked
 * description Description of achievement unlock
 * type Achievement unlock type
 * tip:
 *   fb standard RBI event, event name: Unlock Achievement
 */
+ (void)logUnlockAchievementEvent:(NSString *)description
                                type:(NSString *)type;
```

## Delivery initiated checkout event interface

```
/**
 * Initiate checkout
 * contentData product information can be json ex:[{"id\":"1234\", \"quantity\": 2}, {"id\":"5678\", \"quantity\": 1}]
 * contentId product id
```

```

* contentType product or product_group
* numItems Number of product
* currency Currency type
* totalPrice totalPrice
* tip:
    fb standard RBI event, event name: Initiate Checkout
*/
+ (void)logInitiateCheckoutEvent:(NSString *)contentData
    contentId:(NSString *)contentId
    contentType:(NSString *)contentType
    numItems:(NSInteger)numItems
    paymentInfoAvailable:(BOOL)paymentInfoAvailable
    currency:(NSString *)currency
    valueToSum:(double)totalPrice;

```

Delivery custom event interface

```

/**
* Custom event
* eventName eventName
* valueToSum After it is turned on, the system will associate a
value according to this event, and calculate the total according
to all the occurrences of this event, which is convenient for
checking the average value. When it is 0, the default is not
turned on.
* parameters With parameters, you can add parameters to the
event, and the parameter key is customized as: @{@"order":orderid}
* tip:
    fb custom event
*/
+ (void)logEvent:(NSString *)eventName
valueToSum:(double)valueToSum parameters:(NSDictionary<NSString *,
id> *)parameters;

```

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 *)source
annotation:(id)annotation;

```

The above methods need to be called in the system callback methods.

For example:

```

- (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;
}
```