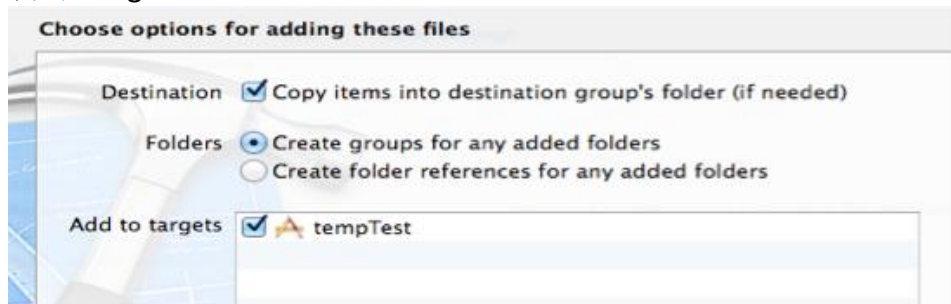


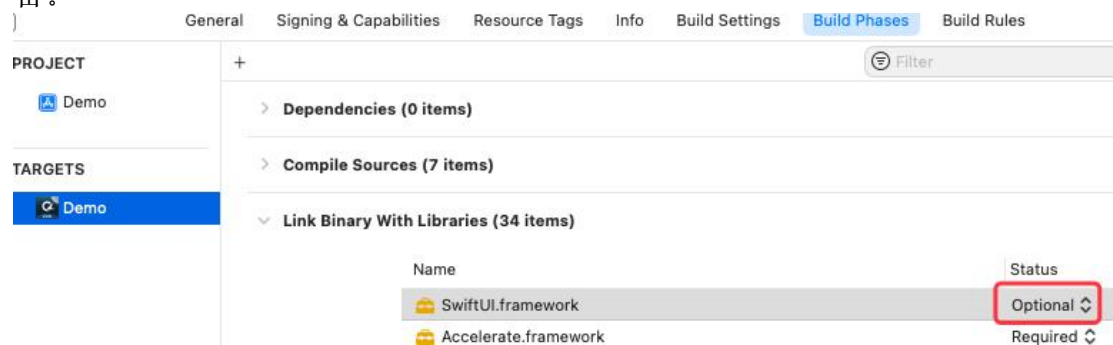
# iOS-SDK接入文档之Firebase数据篇

## 引入 SDK 文件

拖入 SDK 文件夹下 FirebaseAnalytics 文件夹下的相关 framework 并选择正确的 target。



firebaseSDK9.1.0及以上版本需要在target->Build Phases->Link Binary With Libraries加入系统依赖库SwiftUI.framework并设置为可选以兼容iOS12及以下设备。



包含 UnityFramework 动态库特别说明:

Analytics 文件夹下.framework 文件均为静态库，TargetMembership 需要关联到 UnityFramework；

.bundle 资源文件 TargetMembership 需要关联到 Unity-iPhone。

## 接入工程配置

### 1. 获取 iOS 应用的配置文件

1.1 登录 Firebase，然后打开您的项目。



1.2 点击  并选择项目设置。

1.3 在您的应用卡片中，从列表中选择您需要为其获取配置文件的应用的软件包 ID。

1.4 点击  GoogleService-Info.plist。

1.5 把 plist 文件放在 xcode 工程根目录下

注:没有引入此 plist 文件 xcode 工程会异常崩溃

## SDK 内部已经投递事件(接入方无需再处理)

Firebase SDK 内部已经投递安装、启动、购买(包括订阅和续订)、崩溃、更新等事件，本 SDK 内已投递事件：

1. 注册事件：kFIREEventSignUp;
2. 登录事件：kFIREEventLogin。

若需投递其他事件需要游戏在适当的时机调用下面的投递自定义事件接口进行事件投递。

附firebase官方自动收集事件官方文档：

[Firebase自动收集事件](#)

## 接口调用

引入 REDeLoginKit.h

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

## 初始化

**\*\*建议在在应用系统代理启动方法**

**didFinishLaunchingWithOptions 中**

**调用此方法初始化 firebase。 \*\***

```
/** 开始初始化 */
```

```
+ (void)configFireBase;
```

调用代码示例：

```
- (BOOL)application:(UIApplication *)application  
didFinishLaunchingWithOptions:(NSDictionary  
*)launchOptions {  
    [REDeLoginKit configFireBase];  
}
```

## 投递购买事件接口

Firebase SDK 内部会自动上报购买事件，但是 Firebase 后台无法查看沙盒订单，需要查看沙盒订单需要调用此方法手动上报，并且无需上报正式订单，否则会与 Firebase 内部自动的上报的购买重复。

```
/** 上报购买事件-选接
```

```
* 在游戏发货成功之后调用
```

```

* @param currency 购买事件币种 ISO 4217 代码，如果不传将使用
下单时传入的币种信息
* @param orderNo SDK 订单号，会在支付回调返回
* @param productName 商品名称
* @param productId 商品 id
* @param totalPrice 总价，也是支付的金额
* @param price 商品单价
* @param quantity 商品数量
* @param transactionIdentifier 内购交易 id，会在支付回调
返回
*/
+ (void)logFIRAnalyticsPurchaseWithCurrency:(NSString
*)currency orderNo:(NSString *)orderNo
productName:(NSString *)productName
productId:(NSString *)productId totalPrice:(NSString
*)totalPrice price:(NSString *)price
quantity:(NSString *)quantity
transactionIdentifier:(NSString
*)transactionIdentifier;

```

#### 投递自定义事件接口

```

/** 统计任意事件 eventName:事件名 paramDict:统计参数 */
+ (void)logEventWithName:(NSString *)eventName
    andWithParam:(NSDictionary *)paramDict;

```

#### 投递虚拟货币收入事件接口

```

/** 统计虚拟货币收入 */
+ (void)logGetVirtualCurrencyWithCurrencyName:(NSString
*)currencyName
    andWithValue:(NSString *)value;

```

#### 投递加入群组事件接口

```

/** 统计加入群组 */
+ (void)logJoinGroupWithGroupId:(NSString *)groupId;

```

#### 投递角色升级事件接口

```

/** 统计角色升级 */
+ (void)logLevelUpWithLevel:(NSString *)level
    andWithCharacter:(NSString *)character;

```

#### 投递虚拟货币支出事件接口

```

/** 统计虚拟货币支出 */
+ (void)logSpendVirtualCurrencyWithItemName:(NSString *)itemName

```

```
        andWithVirtualCurrencyName:(NSString *)currencyName  
        andWithValue:(NSString *)value;
```

投递开始学习事件接口

```
/** 统计开始学习 */  
+ (void)logTutorialBegin;
```

投递学习结束事件接口

```
/** 统计学习结束 */  
+ (void)logTutorialComplete;
```

在 Xcode 调试控制台中查看日志

您可以启用详细日志记录功能以监控 SDK 的事件记录，从而帮助验证是否正确记录了事件，包括自动和手动记录的事件。

您可以按如下方式启用详细日志记录功能：

在 Xcode 中，依次选择 Product > Scheme > Edit scheme...

从左侧菜单中选择 Run。

选择 Arguments 标签页。

在 Arguments Passed On Launch 部分，添加 -

FIRAnalyticsVerboseLoggingEnabled。

下次您运行应用时，事件将显示在 Xcode 调试控制台中，有助于您即时验证事件是否正在发送。