🐻 koalagamekit-ios

sdk对应的服务端接入文档，请移步： 考拉游戏平台sdk服务端接入文档

考拉游戏平台SDK接入文档【iOS端】v2.0

> 本文档力求简洁明了。接入者所需要的所有代码及说明，在下面的文档中都有详尽介绍。

- 接入前，需要从我方相关人员获得的必要参数

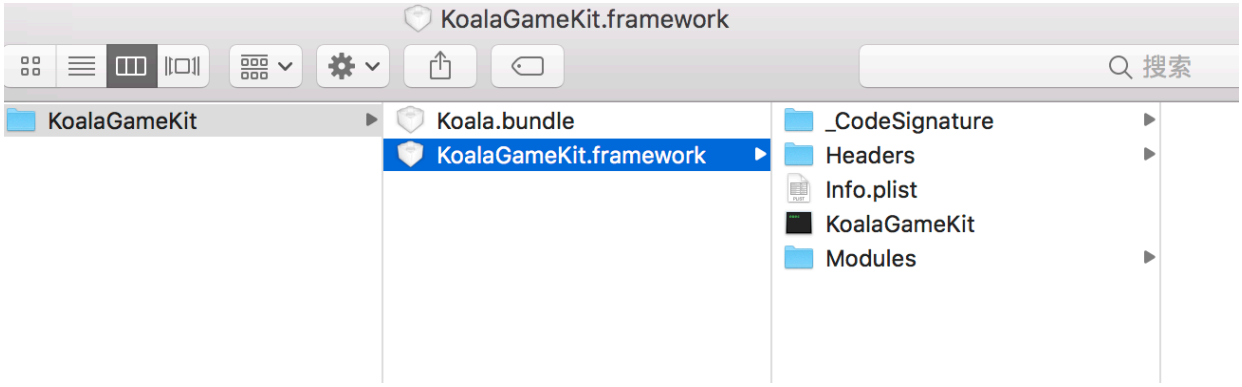
参数	是否必须	描述
appid	是	游戏id;
appkey	是	签名的key;
channel	是	渠道名称；生成规则：游戏中文名称 首字母缩写小写 加大写“IOS”，如：qzsgIOS（全站三国IOS）。
pkver	否	非必要字段，切支付用；如果不填写，sdk会有默认的赋值；详细描述可参见初始化接口的参数描述。

目录

1. 工程配置

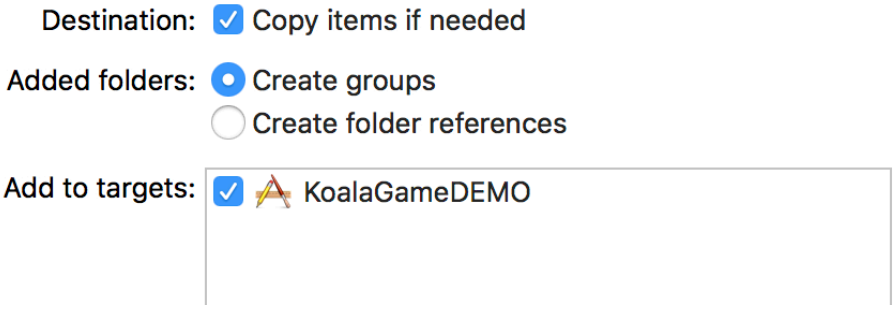
1.1 引入KoalaGameKit

1.1.1 SDK文件清单

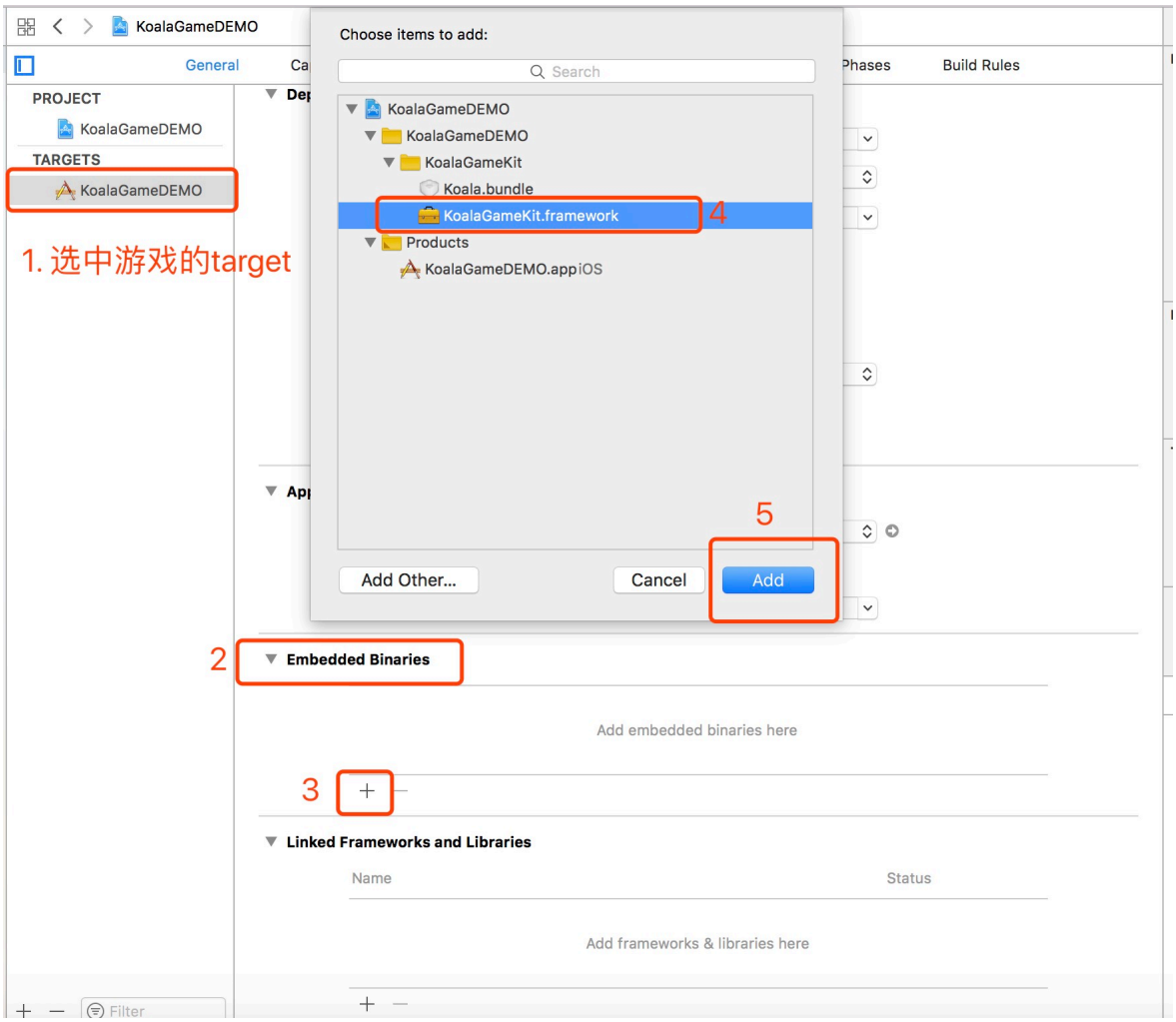


1.1.2 引入项目文件

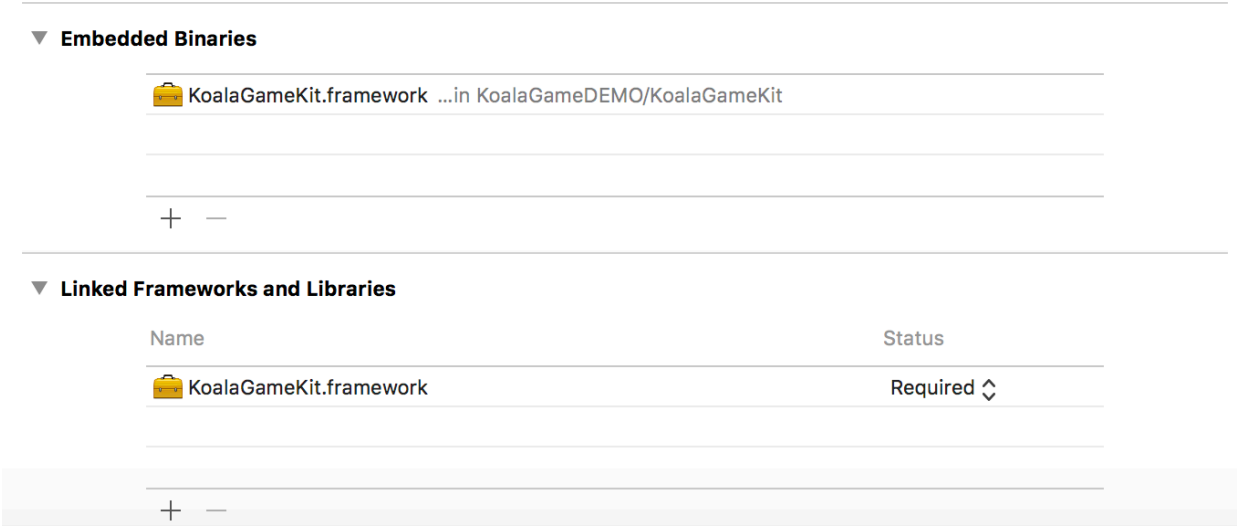
1. 直接把`sdk`文件夹，拖入项目中，建议勾选`Create groups`项。

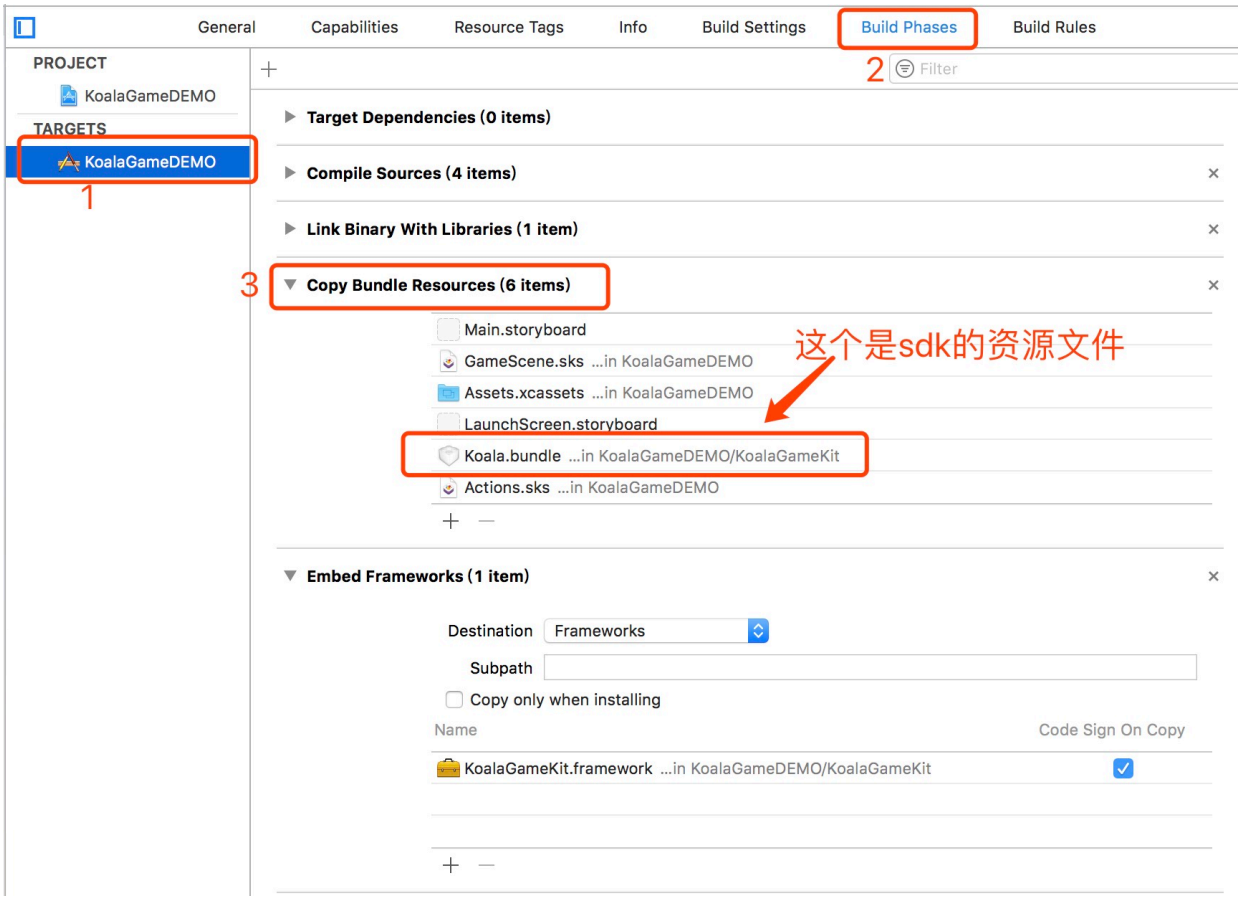


2. 在项目中引入`framework`,如图所示：



3. 对比下图，查看framework和资源文件是否被正确引入：





1.2 info.plist文件

- 在`info.plist`文件中添加如下代码：

```
<!--  ats  -->
<key>NSAppTransportSecurity</key>
```

```
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
<!--  这里要使用到保存到相册的权限(描述文字可以自由发挥哦～)  -->
<key>NSPhotoLibraryAddUsageDescription</key>
<string>游戏要在你的相册里面保存这个截屏哦～</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>游戏要在你的相册里面保存这个截屏哦～</string>

<!--  客服qq要跳转到qq客户端的白名单  -->
<key>LSApplicationQueriesSchemes</key>
<array>
    <string>mqq</string>
</array>
```

1.3 其他说明

> 经过上面的配置，该`sdk`已经配置完毕；目前，改`sdk`并不需要配置额外的依赖，或者url schemes等。
>
> build一下，如无报错，恭喜你，配置部分完成；下面开始代码接入部分。

2. 快速接入

> 引入头文件：

```
#import <KoalaGameKit/KoalaGameKit.h>
```

2.1 初始化

2.1.1 直接上代码

```
// 初始化
[KKConfig sharedConfig].appid = @"<#appid#>"; // app id
[KKConfig sharedConfig].channel = @"<#渠道名称#>";
[KKConfig sharedConfig].appkey = @"<#签名的key#>";  // 签名的key

// 如果无特殊要求，pkver可以不用设置（sdk本身已经有处理）
// [KKConfig sharedConfig].pkver = @"<#版本号去掉小数点#>"; // cp出的ipa包的版本号

/**
 初始化
```

```
 @param completeHandler 初始化的回调
 */
[Koala kgk_initGameKitWithCompletionHandler:^(KKResult *result) {

    if (result.result.boolValue) {

        NSLog(@"初始化成功：%@", result.msg);
    }
    else {

        NSLog(@"初始化失败：%@", result.msg);
    }
}];
```

2.1.2 配置模型

- 初始化需要一个配置模型，需要cp填写以下参数。

> `pkver`参数的说明：
>
> 如无特殊需求，可以不用设置（`sdk`已经做了处理）；
>
> `sdk`的默认处理是：`channel` + `_` + 版本号(去掉小数点) + `_` + `build`号(去掉小数点);
>
> > 如`qzsg_102_11`(全站三国，1.0.2版本，build是11或者1.1);
> >
> > 如需自定义的话，可以参考下面的生成规则：
> >
> > 游戏的版本号字符串（去掉小数点）
> >
> > 1、相同`channel`情况下，需要保证唯一性的一个字符串（需要去掉小数点）； 2、建议生成规则：游戏版本号去掉小数点（出包时的版本号，去掉小数点）；
> >
> > Tips: 使用上面建议的生成规则给`pkver`赋值的话，会覆盖`pkver`的默认值。

```
/** 应用ID  */
@property(copy, nonatomic) NSString *_Nonnull appid;

/** 渠道名称  */
@property(copy, nonatomic) NSString *_Nonnull channel;

/**
    可以为空(sdk会赋默认值)
 0、游戏的版本号字符串（去掉小数点）
 1、相同ver情况下，保证唯一性的一个字符串；
 2、建议生成规则：游戏版本号去掉小数点（出包时修改后的版本号，去掉小数点）；
 */
```

```
@property(copy, nonatomic) NSString *_Nullable pkver;

/** 签名的key */
@property(copy, nonatomic) NSString *_Nonnull appkey;
```

2.1.3 特殊说明

- 初始化接口，一般放在`AppDelegate.m`里面的`application:didFinishLaunchingWithOptions:`方法中去执行。
- 这里只是单纯的调用初始化接口。如需马上调用登录接口，可以在初始化成功的回调中调用登录接口。

2.2 登录

2.2.1 直接上代码

```
/**
 登录

 @param viewController 登录框需要显示在这个vc的上面；可为空，默认为key window的root view controlloer
 @param isAllowUserAutologin 是否允许用户自动登录
 @param floatBallInitStyle 悬浮球第一次展示时的位置样式
 @param isRememberFloatBallLocation 是否记住悬浮球的位置（用户最后一次拖动到的位置）
 @param completeHandler 登录的回调
 */
[Koala kgk_loginWithViewController:<#ur game vc#> isAllowUserAutologin:<#yes:可以自动登录；no:不允许自动登录#> floatBallInitStyle:<#FloatBallStyle#> isRememberFloatBallLocation:<#是否记住悬浮球的位置#> completeHandler:^(KKResult * _Nonnull result) {

    if (result.isSucc) {
        // 登录成功，data是一个user模型：KKUser
        KKUser *user = result.data;
        NSLog(@"登录成功：%@", user);
    }
    else {

        // 登录失败：并不会有登录失败的回调，如网络错误，密码错误等，sdk已自行处理。
        NSLog(@"登录失败:%@", result.msg);
    }
}];
```

2.2.2 用户模型

- 如果用户登录成功，`result.data`是一个用户模型`KKUser`

```
/** 用户id */
@property(nonatomic, copy) NSString *uid;

/** 用户名 */
@property(nonatomic, copy) NSString *username;

/** 时间戳  */
@property(nonatomic, copy) NSString *time;

@property(nonatomic, copy) NSString *sessid;
@property(nonatomic, copy) NSString *gametoken;
```

2.2.3 特殊说明

- `isAllowUserAutologin`参数的说明：如果允许用户自动登录（`YES`）的话，初始化完毕，调用登录接口，sdk会执行自动登录流程；如果不允许(`NO`)，sdk会弹出正常的账号密码登录页面。
- 建议：当第一次初始化成功时，允许用户自动登录；当业务是要切换账号时，不允许用户自动登录，以弹出正常的登录页面。

2.3 上传用户扩展信息

2.3.1 直接上代码

```
KKRole *role = [KKRole new];
role.serverid = @"<#区服id#>";
role.servername = @"<#区服名称#>";
role.roleid = @"<#角色id#>";
role.rolename = @"<#角色名称#>";
role.rolelevel = @"<#角色等级#>";
[Koala kgk_postRoleInfoWithModel:role completionHandler:^(KKResult *
_Nonnull result) {

    NSLog(@"角色上报结果：%@", result);
    if (result.isSucc) {

            NSLog(@"角色上报完成");
        }
        else {

            NSLog(@"角色上报失败：%@", result.msg);
        }
```

```
}];
```

2.3.2 角色模型

```
/** 区服id */
@property(nonatomic, copy) NSString *serverid;

/** 区服名称 */
@property(nonatomic, copy) NSString *servername;

/** 角色ID */
@property(nonatomic, copy) NSString *roleid;

/** 角色名称 */
@property(nonatomic, copy) NSString *rolename;

/** 角色等级 */
@property(nonatomic, copy) NSString *rolelevel;
```

2.4 登出

2.4.1 直接上代码

- 注册登出的通知

```
// 在需要接收登出通知的页面，注册下面的通知，并在实现通知对应的方法里面，做登出成功的处理。

// 注册登出当前账号的通知
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(logoutSuccessNoti:) name:KKNotiLogoutSuccessNoti
object:NULL];
```

- 登出成功的处理

```
- (void)logoutSuccessNoti:(NSNotification *)noti {

    NSLog(@"CP:  登出成功了，cp在这里处理登出游戏账号等操作~");
}
```

- 移除通知

```
// remove self from 登出当前账号的通知
[[NSNotificationCenter defaultCenter] removeObserver:self
name:KKNotiLogoutSuccessNoti object:NULL];
```

2.4.2 切换账号接口

> 这个接口为非必要接口（🐻`sdk`内部也有提供登出/切换账号的入口）；
>
> 如果游戏另有注销/切换之类的入口，可以接入这个接口；
>
> 会发出一个登出成功的通知：`KKNotiLogoutSuccessNoti`（登出回调的处理，参看上一步骤~）；
>
> 登出失败是没有回调的，🐻`sdk`自己处理登出失败.

```
/** 切换账号 */
[Koala kgk_switchAccounts];
```

2.4.3 特殊说明

- CP应该在这里处理游戏账号的退出等操作。

2.5 支付

2.5.1 直接上代码

```
KKOrder *order = [KKOrder new];
order.subject = @"<#商品名称#>";
order.amount = @"#金额（单位元）#";
order.billno = @"#订单号#";
order.iapId = @"#内购ID#";
order.extrainfo = @"<#额外信息#>";
order.serverid = @"<#服务器id#>";
order.rolename = @"<#角色名称#>";
order.rolelevel = @"<#角色等级#>";

/**
 支付
 小说明：由于某些原因（你懂的），所以，sdk里一些关键词会做回避，如注释里面的"支付"会以“制服”代替，望知悉。
 @param order 订单模型
 @param completionHandler 支付回调
 */
[Koala kgk_settleBillWithOrder:order completionHandler:^(KKResult *
_Nonnull result) {
```

```
    // 支付结果：请以服务器的回调为准。
    NSLog(@"支付结果：%@", result);

    if (result.isSucc) {

        NSLog(@"支付成功");
    }
    else if (result.result.integerValue == KKSettleBillStatusNotConfirm) {

        // 跳去第三方支付（web）了
        NSLog(@"跳去第三方支付了，支付结果以服务器之间的回调为准");
    }
    else {

        NSLog(@"支付失败：%@", result.msg);
    }
}];
```

2.5.2 订单模型

```
/** 商品名称  */
@property(nonatomic, copy) NSString *subject;

/** 金额（单位元）  */
@property(nonatomic, copy) NSString *amount;

/** 订单号  */
@property(nonatomic, copy) NSString *billno;

/** 内购id  */
@property(nonatomic, copy) NSString *iapId;

/** 额外信息  */
@property(nonatomic, copy) NSString *extrainfo;

/** 服务器id */
@property(nonatomic, copy) NSString *serverid;

/** 角色名称 */
@property(nonatomic, copy) NSString *rolename;

/** 角色等级 */
@property(nonatomic, copy) NSString *rolelevel;
```

2.5.3 特殊说明

- 支付结果：请以服务器的回调为准。

- 支付分为：内购和第三方支付；如果cp方需要测试不同的支付方式，请联系我方相关人员切换。
- 小说明：由于某些原因（你懂的），所以，sdk里一些关键词会做回避，如注释里面的"支付"会以“制服”代替，望知悉。

3. FAQ

3.1 常见崩溃问题及解决方案

Q1. 动态库没有被正确的加载，而导致的崩溃.

> 症状描述：崩溃。

崩溃的打印：

```
dyld: Library not loaded: @rpath/KoalaGameKit.framework/KoalaGameKit
  Referenced from:
/Users/kaola/Library/Developer/CoreSimulator/Devices/F826E19E-D2A0-4476-
8D8A-BEFA7C014AB1/data/Containers/Bundle/Application/DE458685-5301-4B87-
B8C0-B471C643B4D1/KoalaGameKitDemo.app/KoalaGameKitDemo
  Reason: image not found
(lldb)
```

- 解决方案：

> 崩溃的原因是找不到framework的镜像。解决方案是：
>
> 1. 点击游戏的target -> General -> Embedded Binaries -> "+";
> 2. 从弹出的下拉框中，选择“KoalaGameKit.framework”；
> 3. 重新编译游戏，这个问题将不再出现。

4. 遇到解决不了的问题怎么办?

请联系我们!

- QQ：379636211（运营）
- QQ：2909746131（技术）