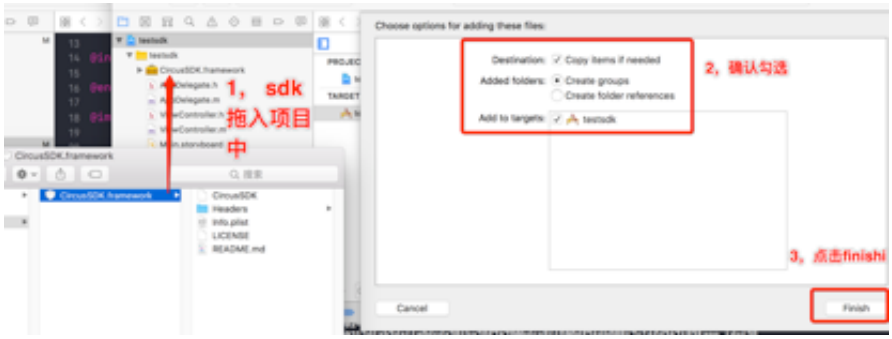
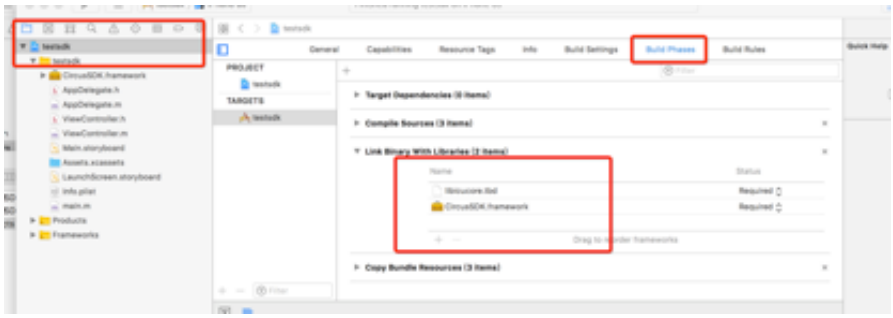
推币机SDK接入文档

1.将sdk导入项目中，暂不支持pod安装

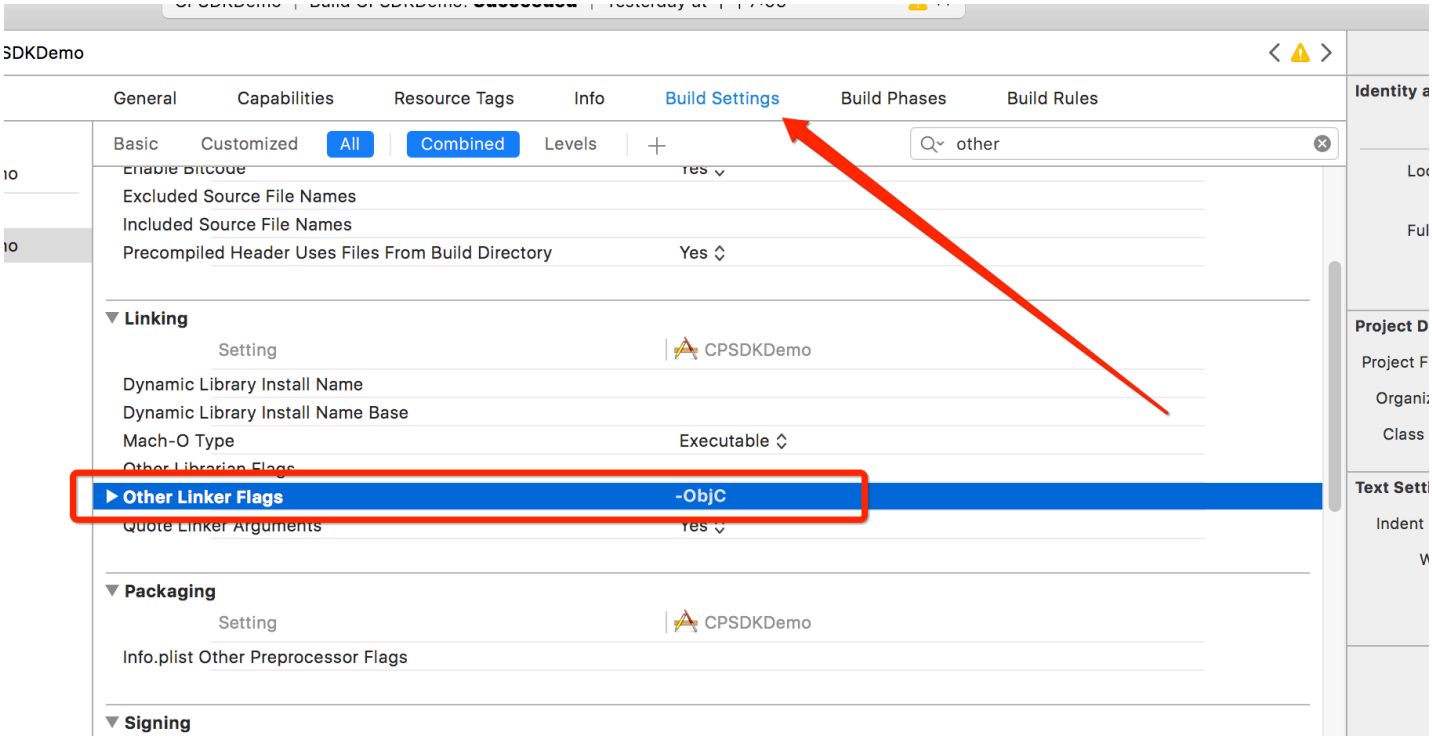


2.将framework添加到Build Phase中，并引入：libicucore.tbd



推币机SDK中有依赖两个第三方库：AFNetworking、SocketRocket，如果项目中没有，请导入项目中.

然后在 Build Settings -> other Linker Flags 中加入 –ObjC，如图：



然后运行项目，确认项目能够运行成功。

3. AppDelegate中初始化SDK

SKD中所需的APPID和SECRET请与友游客服联系获取

```
#import <CircusSDK/CPSDK.h>

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NS
Dictionary *)launchOptions {

    /**
    *  sdk初始化  必须保证init成功后才能进行下一步操作
    *  @param appId       appId
    *  @param appSecret       appsecret    生成APPID和SECRET请与友游客服联系
    *  @param completionBlock 初始化成功的回调
    */
    [CPSDK initWithAppid:@"435" appSecret:@"ql4ub857c68e2kft6vono7of4e" callback:^
(NSDictionary * _Nullable resultDic) {

    }];

    return YES;
}
```

返回值：

```
{“result”:”初始化成功”}
```

4.初始化成功后绑定用户

```
#import <CircusSDK/CPSDK.h>

/**
 *   用户id绑定
 *
 *  @param uid      用户Id
 *  @param successBlock 绑定成功的回调
 */
[[CPSDK sharedCPSDK] bindingUid:self.uidTextField.text success:^(NSDictionary * _N
ullable responseObject) {

    self.isBinded = YES;

    if ([[responseObject objectForKey:@"result"] isEqualToString:@"绑定成功"]) {
        UIAlertController *alertVC = [UIAlertController alertControllerWithTitle:@
"提示" message:@"用户绑定成功" preferredStyle:UIAlertControllerStyleAlert];
        UIAlertAction *cancleAction = [UIAlertAction actionWithTitle:@"确定" style:
UIAlertActionStyleCancel handler:nil];
        [alertVC addAction:cancleAction];
        [self presentViewController:alertVC animated:YES completion:nil];
    }
}];
```

返回值：

```
{“result”:”绑定成功”,"uid":"xxxxx"}
```

5.获取设备列表，可参考文档：https://web.cpo2o.com/flowchat/liveClawDevDoc.html

```
/**
 *  获取设备列表
 *
 *  @param roomtype  int 列表类型（0：全部 | 1:最新 | 2：最热）默认全部
 *  @param getpage  int 要展示哪个分页的数据(默认 1)
 *  @param pageCount  int   每页显示的数量(默认 20)
 */
[[CPSDK sharedCPSDK] getDeviceListWithRoomType:0 currentPage:0 countPerPage:0 succ
ess:^(CPDeviceList * _Nullable deviceList) {
    NSLog(@"%zd", deviceList.total);
    NSLog(@"%zd", deviceList.currentPage);

    self.data = deviceList.devices;
    [self.collectionView reloadData];
}];
```

返回值：

CPDeviceList 类

字段	类型	备注
total	int	总设备数
currentPage	int	当前页数
devices	List	设备列表

CPDevice 类

字段	类型	备注
id	string	设备id
state	int	设备状态：0:空闲,1:游戏中,2:维修

6. 用户进入和离开房间，进入的时候会建立websockte 连接，离开的时候会断开连接

```
/**
*   用户进入房间
*
*/
- (void)joinRoom;

/**
*   用户离开房间
*
*/
- (void)leaveRoom;
```

7. 获取消费档位

```
/**
 *  获取消费档位列表
 *
 *  @param deviceId  int   设备id
 *  @param successCallback    获取数据成功后的回调
 */
[self.circus getConsumeListWithDeviceId:self.deviceId success:^(NSArray<CPConsume
*> * _Nullable consumeList) {

    NSLog(@"%@",consumeList);
}];
```

返回值：

CPConsume 类

字段	类型	备注
id	string	消费档位id
price	int	价格（元）
coins	int	所需币数

8. 使用消费档位玩游戏

```
/**
 *  使用消费档位游戏，游戏中会通过websocket接受到实时游戏结果，只需实现代理方法即可，如果该机器30s内收不到websocket通知的游戏结果，将自动结束游戏
 *
 *  @param deviceId  int   设备id
 *  @param consumeId  int   消费档位ID
 */
- (void)playGameWithConsumeId:(int)consumeId deviceId:(int)deviceId success:(CPSDKSuccessBlock _Nullable )successBlock;
```

调用成功后，推币机会开始掉币。

9. SDK 销毁

```
/**
 *  SDK销毁
 *
 *  @return  是否成功
 */
- (BOOL)destroy;
```

10.websocket 的代理方法，实现这几个方法即可接收到游戏结果的对调

```
/**
*  websocket 接受到消息的回调
*
*  @param message   接受到的消息
*/
-(void)CPwebSocketdidReceiveMessage:(CPMessage *)message {

//  30s 内收不到游戏消息，则手动退出游戏，释放该机器

}

/**
 *  websocket 连接成功的回调
 *
 */
- (void)CPwebSocketdidOpen {

     NSLog(@"%s", __func__);
}

/**
 *  websocket 连接失败的回调
 *
 *  @param error   错误信息
 */
- (void)CPwebSocketDidFailWithError:(NSError * _Nullable)error {

}
```

客户端接收到的消息类型：

CPMessage 类

字段	类型	备注
type	string	消息类型，详情见下表
uid	string	用户id
points	string	掉落币数，即从推币机托盘中掉落的币数，系统会将掉落的币数所对应的钱数加到用户钱包中
bonuses	string	奖励币数，游戏过程中 中奖奖励的推币数

消息类型：

字段	类型	备注
gameStart	string	游戏开始
gameEnd	string	游戏开始
userLogin	string	通知有用户进入房间
bonuses	string	奖励推币数广播
addPoint	string	游戏掉币广播