咕喃 iOS 客户端开发 SDK

1. 系统说明

本框架采用 HTTP 接口和 Socket 协议混用的方式，比如登陆登出以及群聊使用了接口和 Socket 并用，即登陆，要调二次，一是调接口的，二是调 Socket 的。
jxServer.m 包括了所有的接口
JXXMPP.m 为 Socket 管理类

在移植源代码至另一新工程时，如下文件是必须要用到的：

文件	功能
JXChatViewController.*	聊天界面
JXMsgViewController.*	消息
JXGroupViewController.*	我的群组
JXNewFriendViewController.*	朋友验证消息
JXNewRoomVC.*	新建群组
JXSelFriendVC.*	选择好友
JXRoomMemberVC.*	群组成员
roomData.*	群组数据
JXConnection.*	协议请求基类
JXServer.*	协议综合处理类
JXFriendCell.*	验证好友 Cell
JXMessageObject.*	聊天记录数据类
JXMsgAndUserObject.*	消息和用户捆绑类
JXRoomObject.*	群组数据类及操作
JXRoomPool.*	群组缓冲池
JXUserBaseObj.*	用户基类
JXUserObject.*	用户类(不写数据库)
JXXMPP.*	Socket 收发处理类

2. 第三方库

框架名称（或文件夹）	来源	说明
AlipaySDK	https://docs.open.alipay.com/204/105295/	支付宝 sdk
FLAnimatedImage	https://github.com/Flipboard/FLAnimatedImage	适用于 iOS 的高性能动画 GIF 引擎
UMSocial_Sdk_5.2.1	https://developer.umeng.com/sdk/ios?spm=a211g2.181323.0.0.3cb255748VfjSw	友盟
AFNetworking	https://github.com/AFNetworking/AFNe	网络框架

	tworking	
LiveVideo	https://github.com/LaiFengiOS/LFLiveKit	直播框架
GPUImage	https://github.com/BradLarson/GPUImage	基于 GPU 的图像和视频处理的开源 iOS 框架
wechat	https://open.weixin.qq.com/cgi-bin/showdocument?action=dir_list&t=resource/res_list&verify=1&id=1417674108&token=&lang=zh_CN	微信 SDK
BMChineseStringSort	https://github.com/Baymax0/BMChineseSort	列表索引排序
MCDownloader	https://github.com/agelessman/MCDownloader	iOS 下载器
GroupHeaderView	网站已过期	群头像合成
googleMap	https://developers.google.cn/maps/documentation/	google 地图
BaiduMap	http://lbsyun.baidu.com/	百度地图
SDWebImage	https://github.com/SDWebImage/SDWebImage	具有缓存支持的异步图像下载器作为 UIImageView 类别
DMTransitions	网站已过期	VC 转场动画
amr_wav	https://github.com/feuvan/opencore-amr-iOS	语音框架
Bpush	http://push.baidu.com/	百度推送
FMDB	https://github.com/ccgus/fmdb	数据库框架
JSONModel	https://github.com/jsonmodel/jsonmodel	JSON 的神奇数据建模框架 - 允许快速创建智能数据模型
SBJson	https://github.com/kykim/SBJSON	JSON 静态库
QBPopupMenu	https://github.com/questbeat/QBPopupMenu	适用于 iOS 的可自定义弹出菜单。
Socket	https://github.com/shaojiankui/GCDAsyncSocket-TCP-UDP-Python/tree/master/SocketClient-Demo/AsyncSocket/GCD	Socket 框架
Masonry	https://github.com/SnapKit/Masonry	自动布局框架
PhotoChoose	https://github.com/RITL/RITLImagePickerDemo	图库框架（有修改）
ImageEdit	https://github.com/a130785/WWImageEdit	图片编辑器（有修改）
audioRecorder	https://github.com/shaojiankui/libmp3la	音频录制

	me.3.99.5.a	
OrganizTree	https://github.com/Augustyniak/RATreeView	组织架构
ShortVideo	https://github.com/QuintGao/GKDYVideo	短视频
ATMHud	https://github.com/atomton/ATMHud	等待符
MJRefresh	https://github.com/CoderMJLee/MJRefresh/tree/master/MJRefresh	UITableView 上拉刷新，下拉加载更多

静态库		
静态库	功能	来源
libssl.a	OpenSSL 算法库	https://www.jianshu.com/p/b38656443e71 https://github.com/x2on/OpenSSL-for-iPhone
libcrypto.a	DES 等算法库	同上
libxml2.tbd	XML	系统
libresol.tbd		系统
libsqlite3.tbd	Sqlite	系统
libz.tbd		系统
libstdc++.6.0.9.tbd		系统
libopencore-armwb.a	AMR 音频编解码	https://github.com/feuvan/opencore-amr-iOS
libBpush.a	百度推送	http://push.baidu.com/
libidn.a	域名国际化编解码	https://github.com/robbiehanson/XMPPFramework
libmp3lame.a	MP3 音频编解码	https://github.com/shaojiankui/libmp3lame.3.99.5.a
libbz2.tbd		系统

备注：以下红色字体为 HTTP 接口名称。

3. 注册、登录、登出

注册新用户调：user/register 接口,服务端会在接口数据库 mongoDB 和 Tigase 数据库中分别产生一条记录。
登录调：JXServer.login(user/login)以及 JXXMPP.login
登出调：JXServer.logout(user/logout)以及 JXXMPP.logout
在切换到后台或关闭 APP 时须调用登出
在切换到前台或启动 APP 时须调用登录

上次登录的用户 ID 保存在：[[NSUserDefaults standardUserDefaults] stringForKey:kMY_USER_ID];
上次登录的用户密码保存在：[[NSUserDefaults standardUserDefaults] stringForKey:kMY_USER_PASSWORD];

4. 好友列表

服务器数据库列表：jxServer.listFriend(friends/list)
本地数据库列表：JXUserObject:
-(NSMutableArray*)fetchAllFriendsFromLocal;//获取所有互相关注的好友
-(NSMutableArray*)fetchAllRoomsFromLocal; //获取所有关注的群组
-(NSMutableArray*)fetchAllPayFromLocal; //获取所有单方关注的好友
-(NSMutableArray*)fetchAllUserFromLocal; //获取所有用户
-(NSMutableArray*)fetchAllBlackFromLocal; //获取所有黑名单

5. 加好友

相关处理类在：JXFriendObject:
先通过消息协议去打招呼(协议 type:500)，对方收到后显示打招呼的内容、通过验证、回话。
点击回话按钮，则发送消息协议给对方(type:502)，对方显示出来。
点击通过验证，则调用接口：jxServer.addAttention(friends/attention/add)，把关系保存在服务器，并且调用 jxUserObject.insert，保存好友关系到本地数据库；且同时发送消息协议给对方(type:501)，对方收到后在本地数据库改变好友关系；

6. 删好友

把关系在服务器上删除：jxServer.delAttention (friends/attention/delete)
成功后在 didServerResultSucces 回调中删除本地数据库：jxUserObject.delete
并且发送消息给对方知道(协议 type:505)

7. 发消息

发文本消息:

```
JXMessageObject *msg=[[JXMessageObject alloc]init];
msg.timeSend        = [NSDate date];
msg.fromUserId      = MY_USER_ID;
if([self.roomName length]>0)
    msg.toUserId = self.roomName;
else
```

```
        msg.toUserId          = chatPerson.userId;
        msg.content           = message;
        msg.type              = [NSNumber numberWithInt:kWCMessageTypeText];
        msg.isSend            = [NSNumber numberWithInt:transfer_status_ing];
        msg.isRead            = [NSNumber numberWithBool:YES];
        [msg insert:roomName];//保存到本地数据库
        [g_xmpp sendMessage:msg roomName:roomName];//发送消息
        [self showOneMsg:msg];
        [msg release];
```

发语音、图片、文件、视频时，须先调用 g_server,uploadFile 上传文件至服务器，在 didServerResultSucces 回调中进行真正的发送：

```
        NSLog(@"doSendAfterUpload");
        p.content    = [dict objectForKey:@"oUrl"];
        [p updateIsUpload:YES];
        [g_xmpp sendMessage:p roomName:roomName];//发送消息
```

在JXXMPP类里将生成的msg对象转换成ProtoBuf对象：

```
ChatMessage *message = [msg getPbobjcObjWithMsg:msg];
```

再在包头拼接对应字节的密令码和长度等，最后生成完整的发送包 data：

```
NSData *data = message.data;
data = [self getWriteDataWithCmd:kWCCommandTypeChat data:data];
```

最终用 Socket 发送生成的包

```
[self.socket writeData:data withTimeout:- 1 tag:0];
```

8. 收消息

在 JXXMPP.m 中统一处理，先从包里截出对应字节，判断消息长度和密令码 cmd，根据 cmd 做对应处理：

```
- (void)readData:(NSData *)data {

    if (self.lastData.length > 0) {
        NSMutableData *mData = [[NSMutableData alloc] init];
        [mData appendData:self.lastData];
        [mData appendData:data];
        data = [mData copy];
    }

    // 取出协议版本号
    NSData *da = [data subdataWithRange:NSMakeRange(0, 1)];
    Byte bversion[1];
```

```
[da getBytes:bversion length:sizeof(bversion)];
int version = bversion[0];

//取出消息标志位
da = [data subdataWithRange:NSMakeRange(1, 1)];
Byte bflag[1];
[da getBytes:bflag length:sizeof(bflag)];
int flag = bflag[0];

// 取出密令码
da = [data subdataWithRange:NSMakeRange(2, 2)];
short cmd;
[da getBytes:&cmd length:sizeof(cmd)];
NTOHS(cmd);

// 取出消息长度
da = [data subdataWithRange:NSMakeRange(4, 4)];
int len;
[da getBytes:&len length:sizeof(len)];
NTOHL(len);

if (len > data.length) {
    self.lastData = data;
    return;
}

self.lastData = nil;
// 取出消息体
da = [data subdataWithRange:NSMakeRange(8, len)];

NSLog(@"dataLength == %ld", data.length);

if (data.length - 8 > len) {
    [self readData:[data subdataWithRange:NSMakeRange(8 + len, data.length - (8 + len))]];
}

//二进制数据反序列化为对象
switch (cmd) {
    case kWCCommandTypeAuthResp:     // 登录结果
        [self authResult:da];
        break;
    case kWCCommandTypeChat:         // 聊天消息
        [self chatResult:da];
```

```
            break;
        case kWCCommandTypeTypeSuccess: // 请求成功
            [self chatReceipt:da];
            break;
        case kWCCommandTypeRoomRequestResult:    // 群组请求结果
            [self roomRequestResult:da];
            break;
        case kWCCommandTypePullMessageResp: // 请求群离线消息结果
            [self roomPullMessageResp:da];
            break;
        case kWCCommandTypeLoginConflict:    // 被挤下线
            [self loginConflict:da];
            break;
        case kWCCommandTypeError:          // 错误消息
            [self messageError:da];
            break;
        default:
            break;
    }
}
```

9. 群组列表

调用 jxServer.listRoom 获取服务器的列表(room/list)
调用 jxUserObject. fetchAllRoomsFromLocal 获取本地的列表

10. 新建群组

先调用接口(room/add):

```
[g_server addRoom:_room isPublic:_publicSwitch.on isNeedVerify:self.GroupValidationSwitch.on
category:category toView:self];
```

创建群组，然后在 didServerResultSuccess 回调中将 Socket 加入群组：

```
    _chatRoom = [[JXXMPP sharedInstance].roomPool createRoom:_room.roomJid
title:_roomName.text];
    _chatRoom.delegate = self;
```

然后保存群组到本地数据库，并调用 JXSelectFriendsVC 选择好友，邀请加入。

11. 加入群组

http 加入群组调用接口(room/member/update):
[g_server addRoomMember:room.roomId userArray:_userIds toView:self];//用接口即可
Socket 加入群组调用方法:
[g_xmpp.roomPool joinRoom:user.userId title:user.userNickname isNew:NO];

并在 xmppRoomDidJoin 回调中，保存群组到本地数据库

12. 删除群组

一种是退出群组:

```
-(void)onQuitRoom{
    _delete = -1;
    [JXUserObject deleteUserAndMsg:room.roomJid];
    [g_server delRoomMember:room.roomId userId:[g_myself.userId intValue]
toView:self];(room/member/delete)
}

- (void)xmppRoomDidLeave:(XMPPRoom *)sender{//成功退出的回调
    [g_notify postNotificationName:kQuitRoomNotifaction object:chatRoom userInfo:nil];
    [self actionQuit];
}
```

一种是物理删除群组:

```
-(void)onDelRoom{
    [g_server delRoom:room.roomId toView:self];(room/delete)
}

- (void)xmppRoomDidDestroy:(XMPPRoom *)sender{//成功删除的回调
    [g_notify postNotificationName:kQuitRoomNotifaction object:chatRoom userInfo:nil];
    [self actionQuit];
}
```

13. 群发消息

和发单条消息一样，只不过目标用户不同:

```
JXMessageObject *msg=[[JXMessageObject alloc]init];
if([self.roomName length]>0)
```

```
msg.toUserId = self.roomName;
```

14. 收取群组离线消息

单聊的离线消息当用户在线后会主动推送过来，而群组的离线消息是需要用户在线后，传入一个时间段来拉取这个时间段的离线消息

```
NSMutableArray *listArr = [NSMutableArray array];

        // 获取到群组本地最后一条消息
        for (NSInteger i = 0; i < array1.count; i ++) {
            NSDictionary *dict = array1[i];
            if ([[dict objectForKey:@"isRoom"] intValue] == 1) {
                // 获取最近一条记录
                NSArray *arr = [[JXMessageObject sharedInstance] fetchMessageListWithUser:[dict
objectForKey:@"jid"] byAllNum:0 pageCount:20 startTime:[NSDate
dateWithTimeIntervalSince1970:0]];
                JXMessageObject *lastMsg = arr.lastObject;
//                for (NSInteger i = arr.count - 1; i > 0; i --) {
//                    JXMessageObject *firstMsg = arr[i];
//                    if ([firstMsg.type integerValue] != kWCMessageTypeRemind) {
//                        lastMsg = firstMsg;
//                        break;
//                    }
//                }
                if (!lastMsg) {
//                    continue;
                    lastMsg = arr.firstObject;
                }
                JXUserObject *user = [[JXUserObject sharedInstance] getUserById:dict[@"jid"]];
                NSMutableDictionary *taskDic = [NSMutableDictionary dictionary];
                [taskDic setObject:[dict objectForKey:@"jid"] forKey:@"userId"];
                [taskDic setObject:[NSDate dateWithTimeIntervalSince1970:[dict[@"timeSend"]
longLongValue]] forKey:@"lastTime"];
                if (lastMsg) {
                    [taskDic setObject:lastMsg.timeSend forKey:@"startTime"];
                    if (lastMsg.messageId) {
                        [taskDic setObject:lastMsg.messageId forKey:@"startMsgId"];
                    }
                }
                if (user.roomId) {
                    [taskDic setObject:user.roomId forKey:@"roomId"];
```

```
                }
                if ([g_myself.chatSyncTimeLen longLongValue] != -2) {

                    [self createSynTask:taskDic];

                    [_taskArray addObject:taskDic];
                }

                NSString *listStr = [NSString
stringWithFormat:@"%@,%ld",dict[@"jid"],(long)([lastMsg.timeSend
timeIntervalSince1970]*1000)];
                [listArr addObject:listStr];
            }

        }

        if (listArr.count > 0) {
            [g_xmpp pullBatchGroupMessageReqWithJidListArray:listArr];
        }
```

15. 邀请群成员

客户端调用:[g_server addRoomMember]保存到服务器。(room/member/update)
服务端会发送对应的协议消息给被邀请方(协议：907)
被邀请方收到对应的消息后做相应处理：Socket 加入此群，将群组保存到本地数据库

16. 踢出成员

先调用接口 g_server delRoomMember 在服务器删除(room/member/delete)
服务端会发送对应的协议消息给被邀请方(协议：904)
被踢群成员收到对应的消息后做相应的处理：Socket 退出此群，并从本地数据库删除该群组

17. 禁言

调用 g_server setDisableSay 进行禁言，talkTime 记录着什么时候可以说话，单位为秒，为 0 则不禁言。(room/member/update)
在 JXChatViewController.m 中获取这个值，在输入前作出判断，能否说话。

18. 屏蔽群消息

把群组拉入黑名单即可：

```
if([_user.status intValue] == friend_status_black){
    _user.status = [NSNumber numberWithInt:friend_status_friend];
    [[JXXMPP sharedInstance].blackList removeObject:_user.userId];
    p.text = @"屏蔽群消息";
}
else{
    _user.status = [NSNumber numberWithInt:friend_status_black];
    [[JXXMPP sharedInstance].blackList addObject:_user.userId];
    p.text = @"接收群消息";
}
```

19. 拉入黑名单

把单个用户拉入黑名单：

[g_server addBlacklist:user.userId toView:self];(friends/blacklist/add)

发送消息给被拉黑者(协议type:507)

取消黑名单类似处理，调用HTTP接口成功后，发送Socket消息给被取消者(协议type:509)

20. 离线推送

如接收方不在线，除了对方在下次上线时会收到离线消息外，也支持自动推送系统通知，在服务端进行配置即可实现。推送为官方 APNS 推送

21. 目录结构

文件夹	功能	移植是否需要
目录:		
Live	直播	
audioRecorder	MP3 录音器	是
OrganizTree(组织架构)	我的同事	
cell	cell 类	是
3rd	第三方控件	部分要
common	通用的一些 UI，选择当前城市	是

control	通用的一些 UI 控件	是
controller	主要 UI	部分要
data	数据层(内存)	是
emojl	表情	是
friendBlog	朋友圈	
image	图片	
map	地图相关	
model	数据模型类(Db)	是
network	网络协议层	是
person	个人 UI	
Register&Login	登录注册	
redpacket	红包	
videoRecorder	录像类	是
Socket	即时通讯	是
ShortVideo	短视频	
SKShare	咕喃第三方分享	
PhotoChoose	选择照片	是
util	工具类	是
transfer	转账相关	是
meeting	音视频通话	
videoRecorder	录像类	是
blog	朋友圈相关	是
Alerts	弹框类相关	是

22. 文件列表

文件或目录名	功能	移植是否需要
AFNetworking	HTTP 请求	
alipay	支付	
amr_wav	wav 转 amr	
BaiduMap	百度地图	
BPush	百度推送	

DMTransitions		
FMDB	数据库操作框架	
googleMap	google 地图	
GroupHeaderView		
JSONModel	json	
LiveVideo	直播框架	
map	地图控件视图	
MCDownloader	文件下载管理	
Photo.h		
Photo.m		
QBPopupMenu	cell 长按弹出 menu	
SBJson	json 解析	
SDWebImage	异步图片下载管理库	
Socket	Socket 框架	是
ShortVideo	短视频相关	
SKShare	第三方分享进来相关	
PhotoChoose	选择相片	是
AlertView.h	自定义 alertView	
AlertView.m		
ConfirmView.h		
ConfirmView.m		
JXWaitingView.h	停止等待符	
JXWaitingView.m		
audioRecorder		是
JXAudioPlayer.h	音频播放器	是
FileListCell.h	我的文件 cell	
FileListCell.m		
JXCell.h	通用 cell	
JXCell.m		
JXFriendCell.h	好友 cell	
JXFriendCell.m		

JXMediaCell.h		
JXMediaCell.m		
JXNearCell.h	附近 cell	
JXNearCell.m		
JXRecordTBCell.h		
JXRecordTBCell.m		
JXRPacketListCell.h	红包领取 cell	
JXRPacketListCell.m		
JXSettingsCell.h	设置 cell	
JXSettingsCell.m		
JXShareFileTableViewCell.h	群共享文件列表 cell	
JXShareFileTableViewCell.m		
JXTelAreaCell.h	国家编码选择 cell	
JXTelAreaCell.m		
JXAudioCell.h	聊天语音 cell	是
JXAudioCell.m		
JXAVCallCell.h	聊天通话 cell	是
JXAVCallCell.m		
JXBaseChatCell.h	聊天 cell 父类	是
JXBaseChatCell.m		
JXCardCell.h	聊天名片 cell	是
JXCardCell.m		
JXFileCell.h	聊天文件 cell	是
JXFileCell.m		
JXGifCell.h	聊天 gift 动图 cell	是
JXGifCell.m		
JXImageCell.h	聊天图片 cell	是
JXImageCell.m		
JXLinkCell.h	聊天连接 cell	是
JXLinkCell.m		
JXLocationCell.h	聊天位置 cell	是
JXLocationCell.m		
JXMessageCell.h	聊天文本 cell	是

JXMessageCell.m		
JXRedPacketCell.h	聊天红包 cell	是
JXRedPacketCell.m		
JXRemindCell.h	聊天通知消息 cell	是
JXRemindCell.m		
JXSystemImage1Cell.h	单条图文消息 cell	是
JXSystemImage1Cell.m		
JXSystemImage2Cell.h	多条图文消息 cell	是
JXSystemImage2Cell.m		
JXVideoCell.h	聊天视频 cell	是
JXVideoCell.m		
ImageBrowserViewController.h	图片浏览器	是
ImageBrowserViewController.m		
JXInputValueVC.h	输入页	
JXInputValueVC.m		
JXTipBlackView.h	tip view	
JXTipBlackView.m		
PhotoView.h	图片浏览器的图片子 view	是
PhotoView.m		
selectAreaVC.h	选择地区	
selectAreaVC.m		
selectCityVC.h	选择城市	
selectCityVC.m		
selectCountryVC.h	选择国家	
selectCountryVC.m		
selectProvinceVC.h	选择省份	
selectProvinceVC.m		
selectTreeVC.h	弃用	
selectTreeVC.m		
selectValueVC.h	valuePicker	
selectValueVC.m		
webpageVC.h	web 页面	
webpageVC.m		
admobViewController.h	视酷 ViewController 父类	是

admobViewController.m		
hud		
imagePicker		
ImageResize.h	图片处理	
ImageResize.m		
JXBadgeView.h	未读消息数量 view	是
JXBadgeView.m		
JXCollectionView.h	封装 CollectionView	
JXCollectionView.m		
JXDatePicker.h	时间选择器	
JXDatePicker.m		
JXEmoji.h	emoji 表情	是
JXEmoji.m		
JXImageView.h	点击事件的 imageView	
JXImageView.m		
JXLabel.h	点击事件的 label	
JXLabel.m		
JXLocPerImageVC.h	地图带头像大头针 View	
JXLocPerImageVC.m		
JXSelectImageView.h	聊天底部面板	是
JXSelectImageView.m		
JXTabButton.h	tabbar 按钮	
JXTabButton.m		
JXTableView.h	封装 tableview	
JXTableView.m		
JXTableViewController.h	封装 tableview VC	
JXTableViewController.m		
JXTabMenuView.h		
JXTabMenuView.m		
JXTextView.h	封装 textview	
JXTextView.m		
JXTopMenuView.h		
JXTopMenuView.m		
JXTopSiftJobView.h	顶部自定义步进器	
JXTopSiftJobView.m		
JXWaitView.h	停止等待符	

JXWaitView.m		
LXActionSheet		
menuImageView.h		
menuImageView.m		
MJRefresh	下拉刷新	
QCheckbox	选择 button	
SCGIFImageView.h	gift 解析	
SCGIFImageView.m		
UIFactory.h	工厂类	
UIFactory.mm		
JX_SelectMenuView.h	下拉列表 View	
JX_SelectMenuView.m		
JXAboutVC.h	关于我们 VC	
JXAboutVC.m		
JXAudioRecorderViewController.h	音频录制 VC	
JXAudioRecorderViewController.m		
JXChatViewController.h	聊天 VC	是
JXChatViewController.mm		
JXFileDetailViewController.h	文件详情下载页	
JXFileDetailViewController.m		
JXFileViewController.h	群共享文件列表 VC	
JXFileViewController.m		
JXFriendViewController.h	我的好友 VC	
JXFriendViewController.m		
JXGooMapVC.h	地图,google 版	
JXGooMapVC.m		
JXGroupViewController.h	群组列表	是
JXGroupViewController.m		
JXImageScrollVC.h	头像放大	
JXImageScrollVC.m		
JXInputVC.h	文本输入 VC	
JXInputVC.m		
JXLocMapVC.h	地图,baidu 版	
JXLocMapVC.m		

JXMainViewController.h	主视图	
JXMainViewController.m		
JXMsgViewController.h	消息列表	是
JXMsgViewController.m		
JXMyFile.h		
JXMyFile.m		
JXNearVC.h	附近的人 VC	
JXNearVC.m		
JXNewFriendViewController.h	新朋友	
JXNewFriendViewController.m		
JXNewRoomVC.h	新建群组	是
JXNewRoomVC.m		
JXProgressVC.h	进度条 VC	
JXProgressVC.m		
JXRelayVC.h	转发	是
JXRelayVC.m		
JXReportUserVC.h	举报	
JXReportUserVC.m		
JXRoomMemberVC.h	群组信息	
JXRoomMemberVC.m		
JXSearchUserVC.h	搜索用户	
JXSearchUserVC.m		
JXSelFriendVC.h	选择朋友	
JXSelFriendVC.m		
JXSettingsViewController.h	设置详情	
JXSettingsViewController.m		
JXSettingsViewController.xib		
JXSettingVC.h	设置	
JXSettingVC.mm		
JXUserInfoVC.h	用户详情页	
JXUserInfoVC.m		
myMediaVC.h	我的附件	
myMediaVC.m		
userInfoVC.h	弃用	
userInfoVC.m		

constant.db	视酷常量数据库	是
JXConstant.h	常量管理类	是
JXConstant.m		
resumeData.h		
resumeData.m		
roomData.h	群组 model	是
roomData.m		
searchData.h	搜索 model	
searchData.m		
EmojiTextAttachment.h		是
EmojiTextAttachment.m		
emojiViewController.h	表情 VC	是
emojiViewController.m		
FaceViewController.h	emoji 表情	是
FaceViewController.m		
gif.bundle	动图表情资源包	是
gifViewController.h	gift 动图表情	是
gifViewController.m		
JXEmoji.h	富文本	是
JXEmoji.m		
NSAttributedString+EmojiExtension.h		
NSAttributedString+EmojiExtension.m		
SCGIFImageView.h	gift 图片解析类	是
SCGIFImageView.m		
en.lproj	plist 英文国际化	
im_live	直播 plist	
image	图片	
shiku_im-Prefix.pch	pch 文件	是
zh-Hans.lproj	plist 中文简体国际化	
zh-Hant.lproj	plist 中文繁体国际化	

BulletGroupView.h	弹幕显示 view	
BulletGroupView.m		
BulletView.h	单条弹幕 view	
BulletView.m		
ChooseGiftView.h	选择礼物	
ChooseGiftView.m		
JXAnchorViewController.h	直播间主播 VC	
JXAnchorViewController.m		
JXCreateLiveRoomVC.h	新建直播间	
JXCreateLiveRoomVC.m		
JXLiveCell.h	直播列表 cell	
JXLiveCell.m		
JXLiveChatCell.h	直播聊天 cell	
JXLiveChatCell.m		
JXLiveGiftCell.h	直播礼物 cell	
JXLiveGiftCell.m		
JXLiveGiftObject.h	直播礼物 model	
JXLiveGiftObject.m		
JXLiveJidManager.h	直播间 jid 管理单例类	
JXLiveJidManager.m		
JXLiveMemberCell.h	直播间成员列表 cell	
JXLiveMemberCell.m		
JXLiveMemDetailView.h	直播间成员管理 view	
JXLiveMemDetailView.m		
JXLiveMemObject.h	直播间成员 model	
JXLiveMemObject.m		
JXLiveRoomViewController.h	直播间父类	
JXLiveRoomViewController.m		
JXLiveViewController.h	直播列表	
JXLiveViewController.m		
JXPlayerViewController.h	直播间观看者 VC	
JXPlayerViewController.m		
LiveGiftGiveView	礼物显示 View	
JXAutoSizeImageView.h		

JXAutoSizeImageView.m		
JXCustomButton.h	自定义 button	
JXCustomButton.m		
JXMeetingObject.h	语音会议管理类	
JXMeetingObject.mm		
settings	会议配置文件	
libs	会议静态库	
images	会议界面图片	
AskCallViewController.h	假拨号界面	
AskCallViewController.mm		
AudioCallViewController.h	语音通话 VC	
AudioCallViewController.mm		
AudioMeetingViewController.h	弃用	
AudioMeetingViewController.mm		
CallViewController.h	通话 VC 父类	
CallViewController.mm		
NumpadViewController.h	拨号 VC	
NumpadViewController.mm		
Setting_CheckView.xib		
Setting_CheckViewController.h		
Setting_CheckViewController.m		
Setting_CheckViewViewController.h		
Setting_CheckViewViewController.m		
Setting_CodecsViewController.h	编解码设置	
Setting_CodecsViewController.mm		
Setting_IdentityViewController.h	账户设置	
Setting_IdentityViewController.mm		
Setting_MediaViewController.h	媒体设置	
Setting_MediaViewController.mm		
Setting_NetworkViewController.h	网络设置	
Setting_NetworkViewController.mm		
Setting_TraversalViewController.h	穿越设置	
Setting_TraversalViewController.mm		
SettingCell.h	会议设置 cell	
SettingCell.mm		

SettingCell.xib		
SettingCheckCell.h		
SettingCheckCell.mm		
SettingCheckCell.xib		
SettingLabelCell.h		
SettingLabelCell.mm		
SettingLabelCell.xib		
SettingSwitchCell.h		
SettingSwitchCell.mm		
SettingSwitchCell.xib		
SettingTextCell.h		
SettingTextCell.mm		
SettingTextCell.xib		
SettingView.xib		
SettingViewController.h		
SettingViewController.mm		
TransparentToolbar.h		
TransparentToolbar.mm		
VideoCallViewController.h	视频通话 VC	
VideoCallViewController.mm		
GDataXMLNode.h		
GDataXMLNode.m		
JXBlogObject.h	朋友圈 model	
JXBlogObject.m		
JXBlogRemind.h	回复消息	
JXBlogRemind.m		
JXFriendObject.h	好友 model	
JXFriendObject.m		
JXGetPacketList.h		
JXGetPacketList.m		
JXMediaObject.h		
JXMediaObject.m		
JXMessageObject.h	聊天记录数据类	是

JXMessageObject.m		
JXMsgAndUserObject.h	消息和用户捆绑类	是
JXMsgAndUserObject.m		
JXMyTools.h		
JXMyTools.m		
JXPacketObject.h	红包 model	
JXPacketObject.m		
JXRoomMember.h	群组成员 model	是
JXRoomMember.m		
JXRoomObject.h	群组 xmpp 连接类	是
JXRoomObject.m		
JXRoomPool.h	群组缓冲池	是
JXRoomPool.m		
JXRoomRemind.h	群组提醒消息	是
JXRoomRemind.m		
JXShareFileObject.h	共享文件 model	
JXShareFileObject.m		
JXUserBaseObj.h	用户基类	是
JXUserBaseObj.m		
JXUserObject.h	用户类(不写数据库)	是
JXUserObject.m		
JXXMPP.h	Socket 管理类	是
JXXMPP.m		
FileInfo.h		
FileInfo.m		
JXConnection.h	协议请求基类	是
JXConnection.m		
JXServer.h	协议综合处理类	是
JXServer.m		
JXServer+Live.h	直播接口协议综合处理类	
JXServer+Live.m		
versionManage.h	版本管理器	是
versionManage.m		

DepartObject.h	组织 model	
DepartObject.m		
EmployeeTableViewCell.h	员工 cell	
EmployeeTableViewCell.m		
EmployeObject.h	员工 model	
EmployeObject.m		
JXAddDepartViewController.h	添加部门	
JXAddDepartViewController.m		
JXSelDepartViewController.h	选择部门	
JXSelDepartViewController.m		
OrganizObject.h	组织架构节点 model 父类	
OrganizObject.m		
OrganizTableViewCell.h	组织架构节点 cell 父类	
OrganizTableViewCell.m		
OrganizTreeViewController.h	组织架构 VC	
OrganizTreeViewController.m		
RATreeView	树状 tableview 封装	
View		
PSMyViewController.h	我的	
PSMyViewController.m		
PSRegisterBaseVC.h		
PSRegisterBaseVC.m		
PSUpdateUserVC.h		
PSUpdateUserVC.m		
JXOpenRedPacketVC.h	打开红包	
JXOpenRedPacketVC.m		
JXOpenRedPacketVC.xib		
JXRechargeVC.h	充值	
JXRechargeVC.m		
JXRechargeVC.xib		

JXRecordCodeVC.h		
JXRecordCodeVC.m		
JXRedInputView.h	红包编辑 View	
JXRedInputView.m		
JXredPacketDetailVC.h	红包详情	
JXredPacketDetailVC.m		
JXSendRedPacketViewController.h	发红包 VC	
JXSendRedPacketViewController.m		
forgetPwdVC.h	忘记密码	
forgetPwdVC.m		
inputPhoneVC.h	输入手机号	
inputPhoneVC.m		
inputPwdVC.h	输入密码	
inputPwdVC.m		
JXTelAreaListVC.h	国家编号选择 VC	
JXTelAreaListVC.m		
loginVC.h	登录 VC	
loginVC.m		
DESUtil.h	DES 加密工具类	
DESUtil.m		
JXLocation.h	位置工具类	
JXLocation.m		
NSString+ContainStr.h		
NSString+ContainStr.m		
UIImageView+FileType.h	文件类型图片	
UIImageView+FileType.m		
UIView+ScreenShot.h	view 转图片	
UIView+ScreenShot.m		

images		
ImageSelectorCollectionCell.h	图片选择 cell	
ImageSelectorCollectionCell.m		
ImageSelectorViewController.h	图片选择 VC	
ImageSelectorViewController.m		
JXCaptureMedia.h	视频拍摄核心	是
JXCaptureMedia.m		
JXConvertMedia.h	文件格式转化	是
JXConvertMedia.m		
JXVideoPlayer.h	视频播放器	是
JXVideoPlayer.m		
JXVideoPlayerVC.h	视频播放器 VC	是
JXVideoPlayerVC.m		
MyPlayerLayerView.h		
MyPlayerLayerView.m		
pausevideo@2x.png		是
playvideo@2x.png		是
playvideo@3x.png		
recordVideoViewController.h	录制视频 VC	是
recordVideoViewController.m		
UIImage-Extensions.h		
UIImage-Extensions.m		
GCDAsyncUdpSocket.m	即时通讯 Socket（UDP 协议）	
GCDAsyncSocket.m	即时通讯 Socket（TCP 协议	是
Message.pbobjc.m	ProtoBuf 协议类，通讯传输对象	是

23. 资源文件标注

目录	文件名	标注	是否必需
image.xcassets 资源文件夹			
chat			
	chat_bg_blue_press	聊天气泡 bg	是
	chat_bg_blue		是

	chat_bg_white_press		是
	chat_bg_white		是
	dianhua	通话	是
	dianhua1		是
	hongb	红包	是
	hongbaokuan	红包消息 bg	是
	ic_public_menu	公众号	是
	im_map_button_normal	位置	是
	im_map_button_press		是
	jiangp	键盘输入	是
	lashang		是
	luxiang	录像	是
	luxiang1		是
	public_menu		是
	voice_paly_left_1	语音消息动画	是
	voice_paly_left_2		是
	voice_paly_left_3		是
	voice_paly_right_1		是
	voice_paly_right_2		是
	voice_paly_right_3		是
	white		是
audioRecorder		音频	是
videorecorder		视频	是
emoji		表情 emoji	是
call		通话	
common			
	add	加成员	是
	arrow_black		是

	arrow_right		是
	closeicon	关闭	
	Default_Gray		
	delete		
	downlistSelect		
	graybordButton		
	groupImage	群默认头像	是
	heart_praise		
	ic_has_input		
	little_red_dot	未读消息小红点	是
	lose	删成员	是
	no_data_for_the_time_being	没数据	
	position		
	remind_audio		
	remind_video		
	roomDetail	群组详情	是
	selected_fause		
	selected_true		
	send_case_normal	发送,弃用	
	send_case_press		
live		直播图标	
my		我界面图标	
shareFile		文件分享	
tab_bar		底部 tab 图标	
LaunchImage		启动图	
AppIcon		app 图标	
address		address	
开始		已弃用	

暂停		已弃用	
点赞		已弃用	
返回		已弃用	
camra_beauty_close		关闭美颜	
camra_beauty		美颜	
camra_preview		切换摄像头	
card_message		消息	
card_search		搜索	
close_preview		关闭	
gift		礼物	
me_btn_edit_h_		编辑	
me_harvest_exchange		充值	
me_new_icon_zuanshi		充值	
image			
image_i			
	大红包.png		是
	发送位置.png	发位置,弃用	
	红包.png		是
	口令红包.png		是
	资料.png		
	add_picture@2x.png	blog add	
	add_video@2x.png		
	add_voice@2x.png		
	alert_tick@2x.png		
	avatar_normal@2x.png		
	Check@2x.png	已选择	
	Check@3x.png		
	delete.gif	阅后即焚动画	是
	Didn'tcheck@2x.png	未选择	
	Didn'tcheck@3x.png		
	enlarge@2x.png	+	
	enlarge@3x.png		
	feaBtn_backImg_sel.png		

	Haircircleoffriends2@2x.png	白色下拉菜单背景	
	Haircircleoffriends2@3x.png		
	ic_greeting_checked.png	定位到自己	
	im_003_more_button_normal@2x.png		
	im_video_button_press_@2x.png	聊天面板视频	是
	line_640.png		
	location.png		
	login_input_arrow_click@2x.png	弃用	
	more_flag@2x.png		
	navBarBackground.png	导航条	是
	navBarBackground@3x.png		
	noread@2x.png		
	password@2x.png	密码	
	praise@2x.png		
	praise@3x.png		
	RDUp2.png		
	read@2x.png	已读	是
	search@2x.png	搜索	
	search@3x.png		
	send@2x.png	送达	是
	set_list_next@2x.png		
	setting_white.png		
	shareactivity_delete@2x.png		
	title_back@2x.png	后退	
	title_more@2x.png	更多	
	unableCheck@2x.png	禁止选择	
	unableCheck@3x.png		
	userhead@2x.png	默认头像	
	video_contrl_pause@2x.png		
	video@2x.png	视频播放	
	video@3x.png		
	visio_call.png	无用	
	voice_call.png		

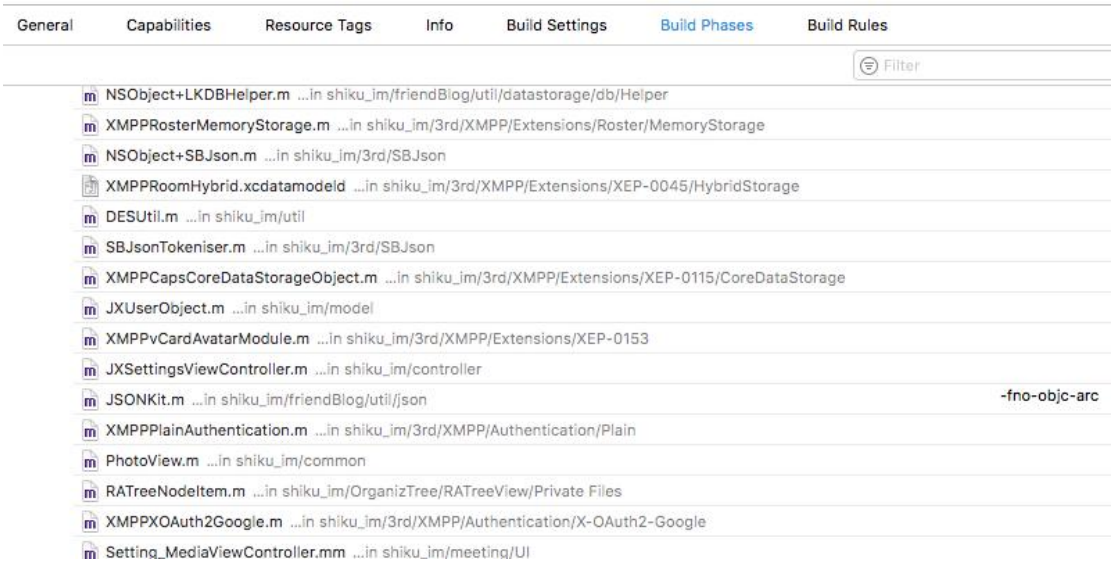
实体文件夹			
image_im			
	down_arrow_black@2x.png		是
	im_10000@2x.png		是
	im_10001@2x.png		是
	im_audio_button_normal@2x.png	聊天面板通话	
	im_audio_button_normal@3x.png		
	im_audio_button_press@2x.png		
	im_audio_button_press@3x.png		
	im_awarda_a_bonus_normal@2x.png	聊天面板红包	是
	im_awarda_a_bonus_normal@3x.png		是
	im_awarda_a_bonus_press@2x.png		是
	im_awarda_a_bonus_press@3x.png		是
	im_card_button_normal@2x.png	聊天面板名片	是
	im_card_button_normal@3x.png		是
	im_card_button_press@2x.png		是
	im_card_button_press@3x.png		是
	im_delete_button_press.png		是
	im_file_button_normal@2x.png	聊天面板文件	是
	im_file_button_normal@3x.png		是
	im_file_button_press@2x.png		是
	im_file_button_press@3x.png		是
	im_file@2x.png		是
	im_input_expression_normal@2x.png	聊天面板表情	是
	im_input_expression_normal@3x.png		是
	im_input_keyboard_normal@2x.png	切换到键盘输入	是
	im_input_keyboard_normal@3x.png		是
	im_input_more_normal@2x.png	下拉聊天面板	是
	im_input_more_normal@3x.png		是

	im_input_ptt_normal@2x.png		是
	im_input_ptt_normal@3x.png		是
	im_map_button_press@2x.png	聊天面板位置	是
	im_map_button_press@3x.png		是
	im_photo_button_normal@2x.png	聊天面板照片	是
	im_photo_button_normal@3x.png		是
	im_photo_button_press@2x.png		是
	im_photo_button_press@3x.png		是
	im_pickup_button_normal@2x.png	聊天面板拍照	是
	im_pickup_button_normal@3x.png		是
	im_pickup_button_press@2x.png		是
	im_pickup_button_press@3x.png		是
	im_video_button_normal@2x.png	聊天面板视频	是
	im_video_button_normal@3x.png		是
	im_video_button_press@2x.png		是
	im_video_button_press@3x.png		是
	new_tips.png	语音红点	是
	title_button_left_press.png		是
	title_button_left.png		是
	title_button_middle_press.png		是
	title_button_middle.png		是
	title_button_right_press.png		是
	title_button_right.png		是
logo			
	320x480.png	启动图,弃用	
	酷聊 57.png	icon,弃用	
papa			
	Accelerate_Audio@2x.png	弃用	
	AlbumRedDelete@2x.png	弃用	
	alert_tick@2x.png	弃用	
	back_page@2x.png	弃用	

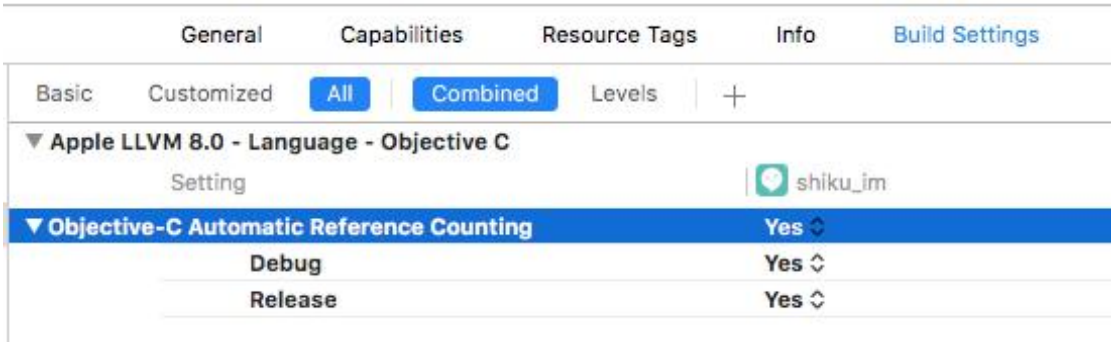
	feeds_play_btn_h_u@2x.png	弃用	
	feeds_play_btn_h@2x.png	弃用	
	feeds_play_btn_u@2x.png	弃用	
	feeds_play_btn@2x.png	弃用	
	navi_arrow_left_white@2x.png	弃用	
	pub_microphone_volume@2x.png	弃用	
	pub_record_button_4i_h@2x.png	弃用	
	pub_record_button_4i@2x.png	弃用	
	pub_record_button@2x.png	弃用	
	pub_recorder@2x.png	弃用	
	publish_toolbar_record_normal@2x.png	弃用	
	rss_post_play@2x.png	弃用	
	shareactivity_delete@2x.png	弃用	

24. 编译设置

本工程已经支持 armv7 arm64 armv7s
本工程 99%采用 ARC，只有一个文件 JSONKit.*采用非 ARC，如要移植到其他工程，请务必对此源文件进行设置，-fno-objc-arc 代表使用非 ARC，您的新工程请参考我们的工程的文件设置一一对应，否则运行中会报各种内存方面的错误：



如果某文件没有设置，我们默认使用非ARC：



25. 语音聊天与视频聊天

1. 说明

音视频开发都是根据加入群组实现音视频通话，群聊是多人加入同一个群组，单聊只是限制两个用户加入同一个群组实现的。

音视频主要文件说明（meeting 文件夹下）：

文件	功能
AskCallViewController.*	拨号等待页面
acceptCallViewController.*	待接听等待页面
JXAVCallViewController.*	音视频通话页面
JXMeetingObject.*	音视频通话逻辑处理
whynotyou.caf / dial.m4a	提示音
images(文件夹)	图片资源

2. 进入群音视频通话

先选择是音频通话还是视频通话，先邀请成员加入音视频通话，

JXChatViewController . onInvite

回调：JXChatViewController . meetingAddMember

对被邀请成员依次发送 Socket 消息协议，type：120(语音) & 115(视频)

自己进入音视频聊天群组：JXAVCallViewController

传入值：

rooNum：传入 roomJid （群组音视频通话群组以群组 roomJid 为唯一标识符）

isAudio：是否是音频通话

3. 被邀请加入群音视频通话

JXMeetingObject . newMsgCome 收到 type 为 115(视频) & 120(音频)
过滤：
if (self.isMeeting)：是否正在进行别的通话
if (n <= 30)：是否是 30 秒之内发送的邀请
进入到待接听提示页面：acceptCallViewController
拒绝：acceptCallViewController . onCancel
如果是群组直接退出页面
接听：acceptCallViewController . onAcceptCall
调用代理：JXMeetingObject . doAudioVideoMeeting
如果是群组直接进入通话群组页面：JXAVCallViewController

4. 单聊发起音视频通话
音频通话：JXChatViewController . onChatAudio
视频通话：JXChatViewController . onChatVideo
进入到拨号等待页面：AskCallViewController
发送 Socket 消息，type：100(语音) & 110(视频)
收到对方接听消息 type：102(接听语音) & 112(接听视频),
进入到通话页面：JXAVCallViewController
对方拒绝接听消息 type：103(拒绝语音) & 113(拒绝视频)
退出当前页面

5. 单聊收到音视频通话
JXMeetingObject . newMsgCome 收到 type 为 100(音频) & 110(视频)
过滤：
if(n>30)：是否是 30 秒之内发送的邀请
self.isInCall：防止刚上线就同时收到邀请和取消的 Socket 消息
进入到待接听提示页面：acceptCallViewController
拒绝：acceptCallViewController . onCancel
发送 Socket 消息，type：103(拒绝语音) & 113(拒绝视频)
接听：acceptCallViewController . onAcceptCall
调用代理：JXMeetingObject . doAudioVideoMeeting
如果是群组直接进入通话群组页面：JXAVCallViewController

6. 通话中事件处理
通话中再次接收到音视频邀请，根据 isMeeting 直接过滤，并向来电方发送忙线消息
忙线消息：type：124
通话中接收到结束通话消息：JXAVCallViewController . newMsgCome 退出页面

点击挂断：JXAVCallViewController . conferenceWillLeave
群组：直接退出页面
单聊：发送 Socket 消息，type：104(结束语音) & 114(结束视频)
单聊通话建立后每隔 3 秒发送一次 ping 消息，并检测是否收到对方发送过来的 ping 消息，如果 3 次检测都没发现收到对方发送过来的 ping 消息则表明对方异常断线，然后自己主动退出
单聊 ping 消息：type：123

26. 一对多直播

通过我们库内的控件进行摄像头采集、编码之后向服务器进行 rtmp 推流，在通过 ijkPlayer 进行播流实现简单的直播，在配合 HTTP 协议与 Socket 协议，实现直播内其他功能。

1.直播下的 Socket 协议说明

下图为直播功能所涉及到的 Socket 协议，以下协议全部由服务器代发，我们只需要调用对应的 HTTP 接口即可。ex:当我在该直播间发送弹幕时，我需要调用”liveRoom/barrage”接口，并传入参数，当服务器收到请求后，根据我们传过去的参数对对应的直播间下的每个成员发送一条
Type==910 的 Socket 消息，当我们收到该消息时，自己页面也显示出该弹幕。

```
// 直播协议
#define kRoomRemind_LiveBarrage     910 // 直播弹幕
#define kRoomRemind_LiveGift        911 // 直播礼物
#define kRoomRemind_LivePraise      912 // 直播点赞
#define kRoomRemind_EnterLiveRoom   914 // 加入直播间
#define kRoomRemind_RoomDisable     926 // 禁用直播间
#define kLiveRemind_ExitRoom        927 // 退出、被踢出直播间
#define kLiveRemind_ShatUp          928 // 直播禁言/取消禁言
#define kLiveRemind_SetManager      929 // 直播设置/取消管理员
```

2.发起直播

在 JXLiveViewController 类点击右上角加号按钮，会先调用/liveRoom/getLiveRoom 接口，判断自己是否已存在直播间，并判断此直播间是否被禁用，如存在并没禁用直接进入直播间并重新开启此直播间。如没有直播间跳转到创建直播间 JXCreateLiveRoomVC 类，填写一些直播间信息，开始创建直播间，调用 liveRoom/creat 接口成功，跳转到直播 JXAnchorViewController 类开始直播。

```
    if ([aDownload.action isEqualToString:act_liveRoomGetLiveRoom]) {

        if ([dict objectForKey:@"roomId"]) {

            if ([[dict objectForKey:@"currentState"] integerValue] == 1) {
                [g_App showAlert:@"直播间已被禁用"];
                return;
            }

            _myLiveDict = [dict mutableCopy];
            [g_App showAlert:Localized(@"JXLive_createexistRoom") delegate:self tag:1221 onlyConfirm:NO];
        }else {
            JXCreateLiveRoomVC * createVC = [[JXCreateLiveRoomVC alloc] init];
            createVC.userId = g_myself.userId;
            createVC.delegate = self;
            //    [g_window addSubview:createVC.view];
            [g_navigation pushViewController:createVC animated:YES];
        }
    }
```

```
-(void)enterLiveRoom:(NSDictionary *)dataDict{
    JXLiveRoomViewController * liveVC;
    if ([[dataDict objectForKey:@"userId"] integerValue] == [g_myself.userId integerValue]) {
        liveVC = [[JXAnchorViewController alloc] init];
    }else{
        liveVC = [[JXPlayerViewController alloc] init];
    }
    liveVC.liveUrl = [dataDict objectForKey:@"url"];//房间地址;
    liveVC.userId  = [dataDict objectForKey:@"userId"];
    liveVC.name = dataDict[@"name"];
    liveVC.nickName = dataDict[@"nickName"];
    liveVC.notice = dataDict[@"notice"];
    liveVC.jid = dataDict[@"jid"];
    liveVC.liveRoomId = dataDict[@"roomId"];
    liveVC.count = [dataDict[@"numbers"] longValue];
//    [g_App.window addSubview:liveVC.view];
    [g_navigation pushViewController:liveVC animated:YES];

}
```

3.观看直播

在 JXLiveViewController 类中点击正在直播的直播间，先判断此直播间是否被禁用，然后 Socket 加入此直播群组，然后调用加入直播间 liveRoom/enterInto 接口，接口成功后跳转到观看直播间 JXPlayerViewController 类

```
    if (_selected == false) {   //防多次快速点击cell
        _selected = true;
        //点击过一次之后。5秒再让cell点击可以响应
        [self performSelector:@selector(changeDidSelect) withObject:nil afterDelay:0.5];

        _sel = indexPath.row;
        _myLiveDict = array[indexPath.row];
        if ([[_myLiveDict objectForKey:@"currentState"] integerValue] == 1) {
            [g_App showAlert:@"该直播间已被禁用"];
            return;
        }
        [_wait start:Localized(@"JXAlert_AddLiveRoomIng") delay:30];
        NSDictionary *dict = array[indexPath.row];

        _chatRoom = [[JXXMPP sharedInstance].roomPool joinRoom:[_myLiveDict objectForKey:@"jid"] title:
            [_myLiveDict objectForKey:@"name"] isNew:YES];
        _chatRoom.delegate = self;
        [_chatRoom joinRoom:YES];

        [self performSelector:@selector(enterMyLiveRoom) withObject:nil afterDelay:0.6];
//        NSString* roomId = [dict objectForKey:@"roomId"];
//        [g_server enterLiveRoom:roomId toView:self];
        dict = nil;
    }
```

```objc
-(void)enterLiveRoom:(NSDictionary *)dataDict{
    JXLiveRoomViewController * liveVC;
    if ([[dataDict objectForKey:@"userId"] integerValue] == [g_myself.userId integerValue]) {
        liveVC = [[JXAnchorViewController alloc] init];
    }else{
        liveVC = [[JXPlayerViewController alloc] init];
    }
    liveVC.liveUrl = [dataDict objectForKey:@"url"];//房间地址;
    liveVC.userId  = [dataDict objectForKey:@"userId"];
    liveVC.name = dataDict[@"name"];
    liveVC.nickName = dataDict[@"nickName"];
    liveVC.notice = dataDict[@"notice"];
    liveVC.jid = dataDict[@"jid"];
    liveVC.liveRoomId = dataDict[@"roomId"];
    liveVC.count = [dataDict[@"numbers"] longValue];
//    [g_App.window addSubview:liveVC.view];
    [g_navigation pushViewController:liveVC animated:YES];

}
```

4.直播间成员列表的实时更新

当我们加入或退出直播间时，会调用’liveRoom/enter’或’liveRoom/quit’接口，当服务器收到我们的请求后，会给该直播间内的每个用户发送一条 type 为 914 或 9027 的 Socket 消息，当我收到该消息后，会发送特定的广播，在直播间收到该广播后我们便去刷新直播间成员列表.

```objc
            case kRoomRemind_EnterLiveRoom:{
                [self newUserComeInRemind:remind];
                break;
            }
            case kLiveRemind_ExitRoom:{
                [self userQuitRemind:remind];
                break;
            }
```

5.聊天

当我们每创建一个直播间，我们都需要创建一个对应的群组，以及在服务器的数据库内创建一个对应的直播间，该群组不会存储在服务端的群组表内，也不会被我们当做朋友存入朋友表，他是依附在该直播间下的，在直播间下有一个 roomJid 即代表该群组，我们加入和退出该直播间，都需要调用 Socket 的加入与退出群组方法，实现聊天功能。

```objc
-(void)createRoom:(NSString *)roomName desc:(NSString *)desc{
    NSString* jidStr = [XMPPStream generateUUID];
    jidStr = [[jidStr stringByReplacingOccurrencesOfString:@"-" withString:@""] lowercaseString];
    _room = [[roomData alloc] init];
    _room.roomJid= jidStr;
    _room.name   = roomName;
    _room.desc   = desc;
    _room.userId = [g_myself.userId longLongValue];
    _room.userNickName = g_server.myself.userNickname;

    _chatRoom = [[JXXMPP sharedInstance].roomPool createRoom:jidStr title:roomName];
    _chatRoom.delegate = self;

    [_wait start:Localized(@"JXAlert_CreatRoomIng") delay:30];

    [g_server createLiveRoom:MY_USER_ID nickName:g_server.myself.userNickname roomName:_room.name
        notice:_room.desc jid:_room.roomJid toView:self];
}
```

```objc
-(void)quitLiveRoom{
    [g_server quitLiveRoom:_liveRoomId toView:nil];
    _chatRoom.delegate = self;
    [_chatRoom.xmppRoom leaveRoom];
    memberData * member = [[memberData alloc] init];
    member.userId = [g_myself.userId integerValue];
    [_chatRoom removeUser:member];
    _chatRoom.delegate = nil;
    [_chatTextView resignFirstResponder];
    [self performSelector:@selector(afterDelaySetLiveJidNil:) withObject:_jid afterDelay:30.0f];
    [self actionQuit];
}
```

6.弹幕

当我们发送一条弹幕时，调用”liveRoom/barrage”接口，服务端收到请求后，给直播间内每个人发送一条 910 消息，收到消息后，我们也发送一条特定的通知到直播界面进行显示。

```objc
-(void)sendBarrage:(NSString *)textStr{
    if (g_App.myMoney >= BARRAGE_PRICE){
        [g_server liveRoomBarrage:textStr roomId:_liveRoomId toView:self];
    }else{
        [g_App showAlert:Localized(@"JX_NotEnough") delegate:self tag:2000 onlyConfirm:NO];
    }
}
```

收到弹幕时处理：

```objc
//直播间通知消息
if ([remind.objectId isEqualToString:self.jid]) {
    switch ([remind.type intValue]) {
        case kRoomRemind_LiveBarrage:{
            //显示到聊天列表里
            //[self showOneMsg:remind];
            //弹幕区滚动
            [_barrageView showNewBarrage:remind.content];
            break;
        }
```

7.送礼物

送礼物时，调用’liveRoom/give’接口，服务端收到请求会发送 type==911 的消息，当我们收到消息时，发送特定的通知至直播间界面

```objc
-(void)chooseGiftViewDelegateGift:(JXLiveGiftObject *)giftObj count:(NSUInteger)count{
    double giftPrice = [giftObj.price doubleValue];
    if (g_App.myMoney >= giftPrice){
        [g_server liveRoomGiveGift:self.liveRoomId anchorUserId:self.userId giftId:giftObj.giftId
            price:giftObj.price count:1 toView:self];
//        g_App.myMoney -= giftPrice;
    }else{
        [g_App showAlert:Localized(@"JX_NotEnough") delegate:self tag:2000 onlyConfirm:NO];
    }
}
```

收到礼物时处理：

```objc
-(void)showGiftRemind:(JXRoomRemind *)remind{
    if ([remind.userId integerValue] != [g_myself.userId integerValue]) {
        [self showGiftAnimate:remind.userId giftId:remind.content fromUserName:remind.fromUserName];
    }
}
-(void)showGiftAnimate:(NSString *)userId giftId:(NSString *)giftId fromUserName:(NSString *)fromUserName{
```

8.点赞

点赞时，调用’liveRoom/praise’ 接口，服务端收到请求会发送 type==912 的消息,当我们收到消息时，发送特定的通知至直播间界面

```
-(void)sendPraise{
    [g_server liveRoomPraise:self.liveRoomId toView:self];
}
```

收到点赞时处理：

```
case kRoomRemind_LivePraise:{
    [self showLove];
    break;
}
```

9.设置管理员

主播拥有设置管理员的功能，当设置管理员时，调用’liveRoom/setmanager’接口，服务端收到请求会发送 type==929 的，当我们收到消息时，发送特定的通知到直播间界面，收到通知后，我们判断被设置的对象是不是自己，是自己则将自己的 type 改为管理员类型，这样就可以对其他人进行权限管理，不是自己也需要更新他的 type，因为管理员不可以对另外的管理员进行操作

```
case JXLiveMemDetailActionTypeSetManager:
    [g_server liveRoomSetManager:memData.userId liveRoomId:_liveRoomId toView:self];
    break;
```

10.禁言/取消禁言

主播和管理员拥有对直播间其他成员禁言的权限，当对某人禁言获取消息禁言时，我们都调用’liveRoom/shutup’接口，只不是禁言’state’参数传的为 1 取消禁言’state’传的为 0。当服务器收到请求后，也只会推送 type==928 的消息给我们，我们通过判断 ChatMessage 的 conten 字段，如果为 0 则代表取消禁言，否则为禁言，我们在发送特定的广播到直播界面

```
case JXLiveMemDetailActionTypeShutUp:{
    NSInteger shutUpState = [memData.state integerValue];
    shutUpState = shutUpState == 1 ? 0 : 1;

    [g_server liveRoomShutUPMember:memData.userId liveRoomId:_liveRoomId state:shutUpState
        toView:self];
    break;
}
```

11.踢人

主播和管理员拥有踢出其他成员的权限，当踢出某人时，回调用’liveRoom/kick’接口，服务端收到该请求时，会发送一条 type==927 的消息给直播间成员，但是我们主动离开直播间服务端也是发送 type 为 927 的消息，所以我们也是通过某些特性判断该操作是主动退出还是被踢出，在发出特定的通知

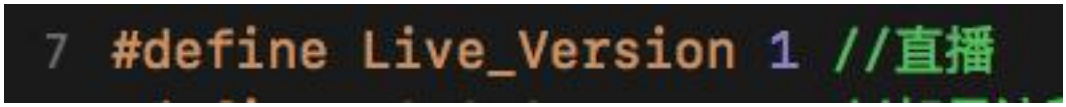
```
case JXLiveMemDetailActionTypeKick:
    [g_server liveRoomKickMember:memData.userId liveRoomId:_liveRoomId toView:self];
    break;
```

收到被踢时处理：

```objc
-(void)userQuitRemind:(JXRoomRemind *)remind{

    if ([remind.toUserId intValue] == [self.userId intValue]) {
        NSLog(@"主播走了");
        [g_App showAlert:@"主播已停止直播" delegate:self tag:2457 onlyConfirm:YES];
        return;
    }
    if([remind.toUserId intValue] == [_myMember.userId intValue]){
        //        //自己被踢出房间
        [g_App showAlert:Localized(@"JXLiveVC_AlreadyKickOutRoom")];
        [self performSelector:@selector(quitLiveRoom) withObject:nil afterDelay:1.5f];
        return;
    }

    [_membersSet removeObject:remind.toUserId];
    JXLiveMemObject * mem = [self memArrayObjectWithUserId:remind.toUserId];
    if (mem == nil)
        return;
    NSUInteger index = [_membersArray indexOfObject:mem];
    if (index == NSNotFound)
        return;

    [_membersArray removeObject:mem];
    NSIndexPath * indexpath = [NSIndexPath indexPathForItem:index inSection:0];

    [_membListCollection deleteItemsAtIndexPaths:@[indexpath]];
    [self refreshCountLabel];
}
```

12.移植说明

pch 文件里打开注释 Live_Version

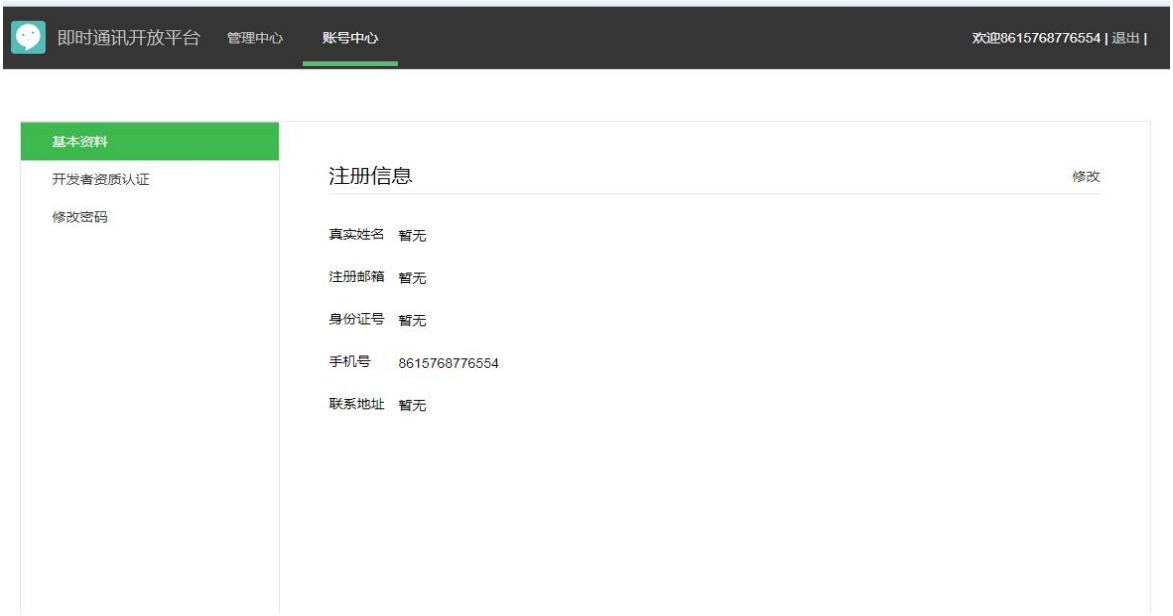


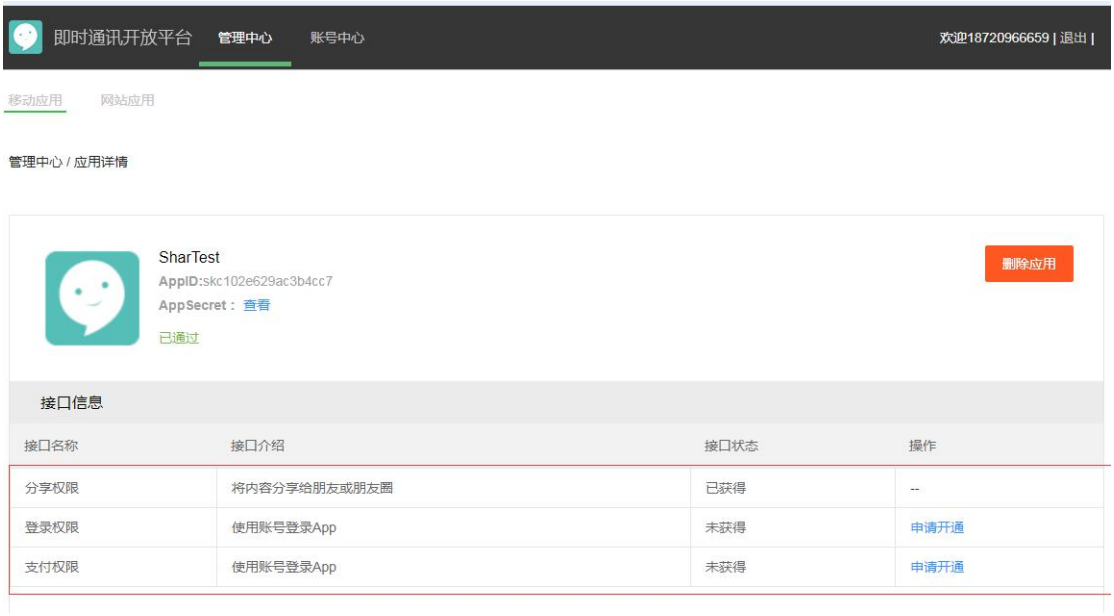
shiku_im 下 Live 文件夹

shiku_im 下 3rd 文件夹 下 LiveVideo 文件夹

27. 集成登录分享 SDK

1. 访问 http://imapi.shiku.co/open/login ， 申请成为开发者
2. 将需要集成 SDK 的应用注册到我们的开发平台内，获取对应的 appId 与 appSecret
3. 根据自己的需求开启对应的功能(目前只支持登录与分享)





4.以上步骤完成后，将 libShikuSDK 文件夹导入到自己项目中

libShikuSDK 文件夹下文件说明：

文件	功能
ShikuConfig.*	配置文件，导入头文件
SKApiManager.*	Api 管理，登录分享回调
SKApi.*	主要事件执行类
SKBaseResp.*	模型类
SKSendAuthResp.*	登录模型类
SKShareResp.*	分享模型类

在 AppDelegate 类中导入头文件 #import "ShikuConfig.h"

并实现：

```
// NOTE: 9.0以后使用新API接口
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:
    (NSDictionary<NSString*, id> *)options
{

    [SKApi handleOpenURL:url delegate:[SKApiManager sharedManager]];
    return YES;
}
```

5.实现登录功能

SKApi 调用方法：

+ (BOOL)AuthAppId:(NSString *)appId AppSecret:(NSString *)appSecret urlSchemes:(NSString *)urlSchemes;

```
/*! @brief 发送登录请求
 * appId: 视酷开发者平台注册app时生成的appId
 * appSecret: 视酷开发者平台注册app时生成的appSecret
 * urlSchemes: 当前app的URL Schemes
 */

+ (BOOL)AuthAppId:(NSString *)appId AppSecret:(NSString
    *)appSecret urlSchemes:(NSString *)urlSchemes;
```

6. 实现分享功能

SKApi 调用方法：

+ (BOOL)share:(NSString *)url title:(NSString *)title subTitle:(NSString *)subTitle imageUrl:(NSString *)imageUrl AppName:(NSString *)appName AppIcon:(NSString *)appIcon AppId:(NSString *)appId AppSecret:(NSString *)appSecret urlSchemes:(NSString *)urlSchemes;

```
/*! @brief 发送分享请求
 * url: 分享点击跳转的url
 * title: 分享显示的标题
 * subTitle: 分享显示的二级标题
 * imageUrl: 分享显示的图片url
 * appName: 分享显示本软件的app名
 * appIcon: 分享显示本软件的appIcon
 * appId: 视酷开发者平台注册app时生成的appId
 * appSecret: 视酷开发者平台注册app时生成的appSecret
 * urlSchemes: 当前app的URL Schemes
 */
+ (BOOL)share:(NSString *)url title:(NSString *)title subTitle:
    (NSString *)subTitle imageUrl:(NSString *)imageUrl AppName:
    (NSString *)appName AppIcon:(NSString *)appIcon AppId:
    (NSString *)appId AppSecret:(NSString *)appSecret urlSchemes:
    (NSString *)urlSchemes;
```

7. 登录分享回调

回调类：SKApiManager

```
// 登录认证回调
- (void)onAuthResp:(SKBaseResp *)resp {

    [[NSNotificationCenter defaultCenter]
        postNotificationName:@"kOpenUrl" object:resp];
}

// 分享回调
- (void)onShareResp:(SKBaseResp *)resp {

}
```

登录回调参数说明:

```
@property (nonatomic, copy) NSString *userId;    // 视酷用户Id
@property (nonatomic, copy) NSString *userName; // 视酷用户名
@property (nonatomic, assign) BOOL authSuccess; // 是否登录成功
@property (nonatomic, copy) NSString *errorMsg; // 登录错误信息
```

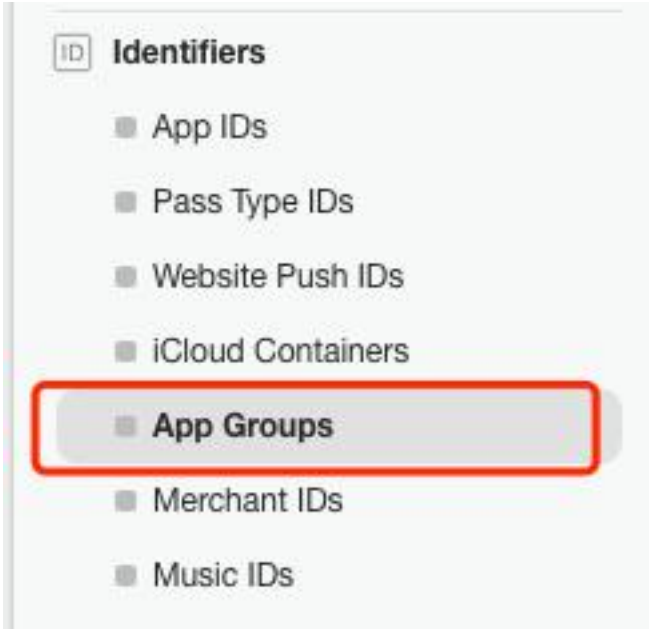
分享回调参数说明:

```
@property (nonatomic, assign) BOOL shareSuccess;    // 是否分享成功
@property (nonatomic, copy) NSString *errorMsg;     // 分享错误信息
```

28. 系统分享

1. 登录 https://developer.apple.com/account/ios/identifier/bundle
2. 注册系统分享专用的包名， 取名格式:******.share (******指的是 App 包名),

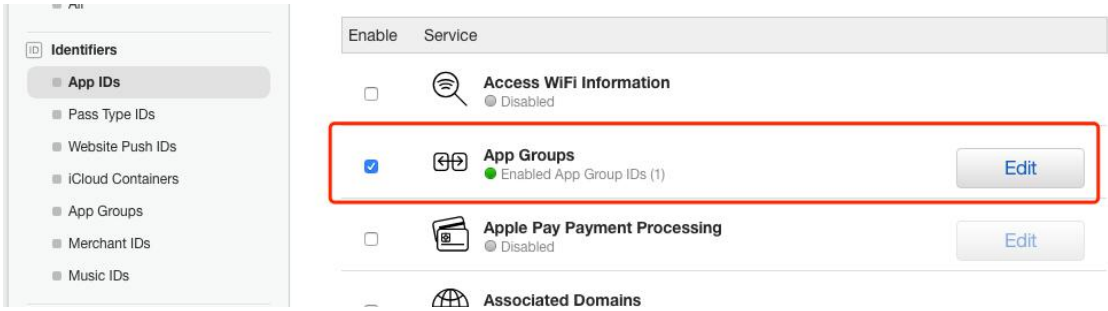
   注册包名和普通包名一样
3. 注册 App Groups ,选定图中 App Group ， 接下来和普通包名注册一样



4. 包名绑定 App Groups：

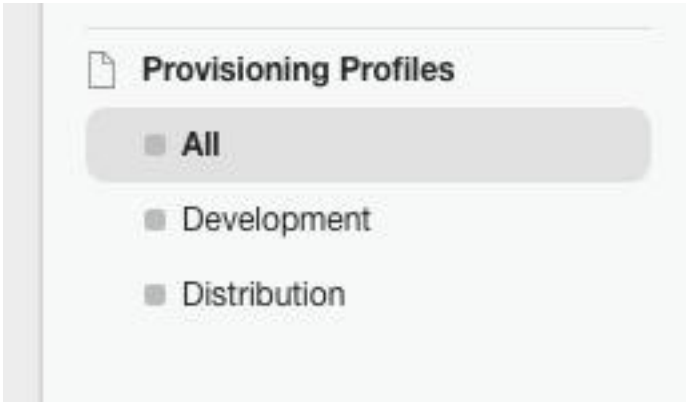
   1、编辑原 App 包名 找到图中部分， 编辑并绑定已注册的 App Groups

2、编辑新注册系统分享专用包名，编辑并绑定已注册的 App Groups



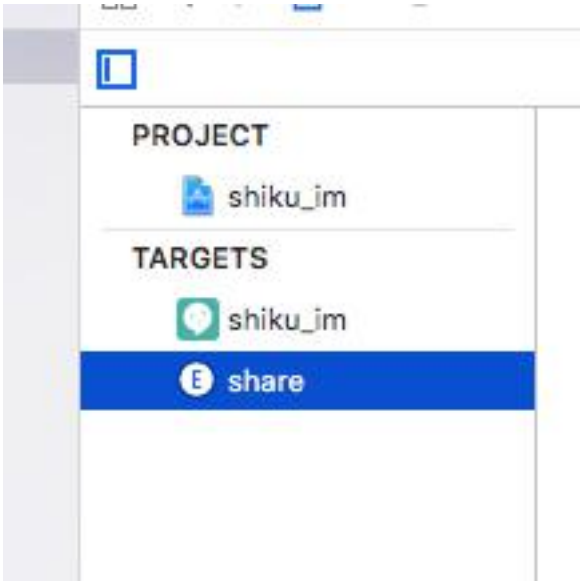
5. 配置文件：

创建系统分享专用的包名后，根据该包名创建配置文件并安装到系统



6. 配置系统分享证书

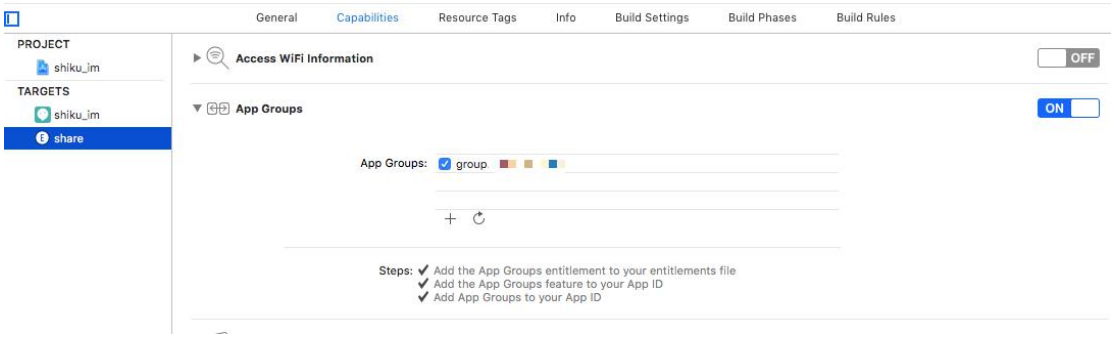
找到图中 Targets 配置包名和证书



7. XCode 中绑定 App Groups

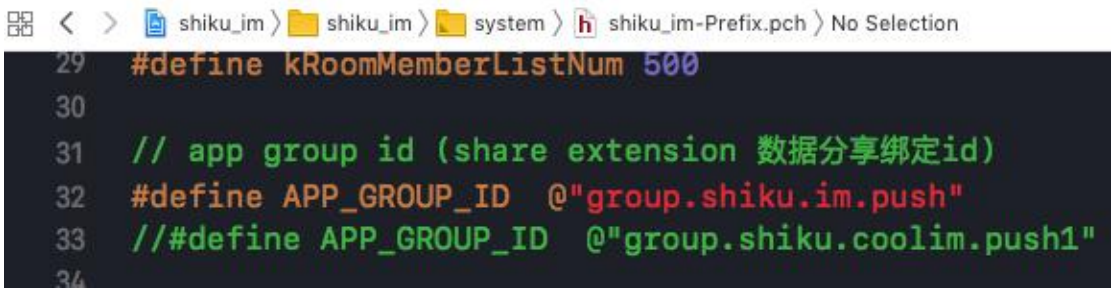
打开图中的 App Groups，并勾选已注册的 App Groups

此处注意：原 App 和系统分享（share）均要做此操作并绑定同一个 App Groups



8. 更改项目中设计到的 App Groups

找到图中.pch 文件，把 APP_GROUP_ID 替换成已注册的 App Groups 即可



Share 文件夹下文件说明:

文件	功能
JXCustomShareVC.*	主界面
JXSelectFriendVC.*	选择好友列表
JXShareViewController.*	分享生活圈界面
JXShareUser.*	好友模型类
JXNetwork.*	网络接口和回调处理类
JXHttpRequet.*	网络请求类

JXCustomShareVC.m

获取系统图片、视频、链接 数据方法

shiku_im 〉 share 〉 JXCustomShareVC.m 〉 -getDataWithItem:

```
250
251 - (void)getDataWithItem:(NSExtensionItem *)extensionItem {
252     //可以从NSExtensionItem项中的attachments属性中获得附件数据，如音频、视频、图片等，NSItemProvide就是实例的表示
253     NSItemProvider *itemProvider = [[extensionItem attachments] firstObject];
254
```

跳转到好友列表

```
121 #pragma mark - 发送给朋友
122 - (void)sendToFriend {
123     JXSelectFriendVC *selVC = [[JXSelectFriendVC alloc] init];
124     selVC.delegate = self;
125     [self presentViewController:selVC animated:NO completion:nil];
126 }
```

选择好友后的回调（其中处理发送逻辑）

```
128 - (void)sendToFriendSuccess:(JXSelectFriendVC *)selectVC user:(JXShareUser *)user {
129     self.user = user;
130     self.proInt = 0;
131     self.isSendToFriend = YES;
132     [self setProgressView];
133     [self updateViewsIsComplete:NO title:nil content:nil];
134     if (self.url.length > 0) {
```

跳转到生活圈编辑界面

```
157 #pragma mark - 分享给生活圈
158 - (void)shareLifeCircle {
159     if (self.url.length > 0) {
160         //分享图片到生活圈
161         [self setProgressView];
162         [self updateViewsIsComplete:NO title:nil content:nil];
163         [[JXHttpRequet shareInstance] addMessage:self.url type:1 data:nil flag:3 toView:self];
164         return;
165     }
166
167     JXShareViewController *shareVC = [[JXShareViewController alloc] init];
168     shareVC.delegate = self;
169     shareVC.image = self.imgV.image;
170     [self presentViewController:shareVC animated:NO completion:nil];
171
172 }
```

发送给生活圈回调

```
174 - (void)sendToLifeCircleSucces:(JXShareViewController *)shareVC {
175     self.text = shareVC.textView.text;
176     self.proInt = 0;
177     self.isSendToFriend = NO;
178     [self setProgressView];
179     [self updateViewsIsComplete:NO title:nil content:nil];
180
181     NSString *path = [[JXHttpRequet shareInstance] getDataUrlWithImage:shareVC.image];
182     [[JXHttpRequet shareInstance] uploadFile:path validTime:nil messageId:nil toView:self];
183 }
```