云通讯 IM——iOS 快速开发文档

2015 年 1 月

云通讯

1.集成流程

(1)注册成为云通讯平台用户。

(2)登陆云通讯平台，进入管理控制台，创建自己的应用。

(3)下载云通讯 iOS SDK，根据开发指南进行编码实现。

2.前期准备

(1)注册成为云通讯的用户，详见新手指引

(2)登陆云通讯平台，进入管理控制台，创建自己的应用。在创建 IM 类应用的时候，只需填写应用名称和选择应用对应的行业，不必选择“启用应用回调地址”、“启用 IVR”、“启用 TTS”、“服务器白名单”等选项。

(3)应用创建成功后，在“应用列表”里面里面可以看见自己刚刚创建的应用，可以对该应用进行编辑和删除操作。当该应用集成完云通讯 IM 能力后准备正式对外发布时，需要点击“上线”申请，此时应用才可以在云通讯的生产环境使用，否则只能在沙盒环境使用。

3.核心概念

(1)生产环境：应用上线后的正式生产环境，对应的接入域名为：https://app.cloopen.com:8883

(2)沙盒环境：应用开发测试时使用的调试环境，对应的接入域名为：https://sandboxapp.cloopen.com:8883

(3)开发者主账号：每个开发者在云通讯平台对应一个主账号

(a)ACCOUNT SID：主账号 id。

(b)AUTH TOKEN：主账号验证 token。

(4)AppID：应用标识，每个创建的应用都对应唯一的 id 标识。

(5)子账号:每个 App 用户在云通讯对应一个子账号,包括账号 id 和验证 token，用于安全性验证。

(6)voip 账号：voip 账号主要用于云通讯平台的 voip 通讯能力，每个子账号都对应一个 voip 账号，由系统自动生成。

说明：每一个注册开发者对应一个主账号。

每一个主账号下可以创建多个 App。

每一个 App 下面可以创建亿级子账号。

每一个子账号对应一个 voip 账号。

4.子账号创建

(1)通过控制台创建

当初期开发测试时，可以通过控制台的“创建子账号”功能创建少量子账号进行调试。如图所以

点击右上角“创建新的子账号”按钮，进入创建子账号窗口

选择归属应用和创建数量，提交后即可创建子账号。

(2)接口创建

通过创建子账号的 REST 接口，可以动态创建子账号。对于通过接口创建子账号的逻辑通常有两种。

① 新用户注册

第一步：App 用户通过 App 注册页面发送到 App 后台。

第二步：App 后台创建用户账号成功，同时调用云通讯创建子账号接口创建云通讯子账号和 voip 账号。

第三步：云通讯返回创建的子账号和 voip 账号后，在 App 后台和用户的账号做绑定入库。

② 老用户注册

对于 App 已经存在的老用户，可以在 App 后台写一个简单的程序，为已经存在的账号创建云通讯子账号和 voip 账号，并和原账号绑定入库。

5.下载 sdk 和 demo

iOS SDK 的下载地址为: http://xxxxx

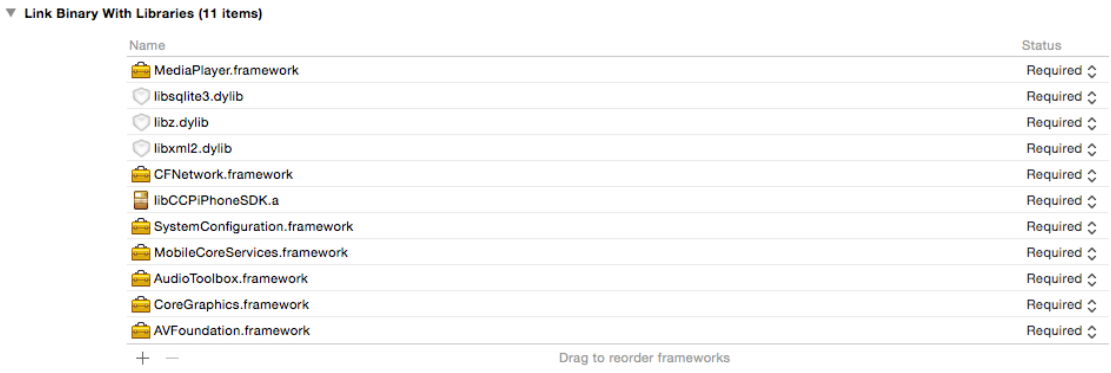
iOS Demo 的下载地址为：http://yyyyy.

6.开发准备

(1)解压缩 SDK 到自己的工程目录下

(2)导入头文件 ECDeviceHeaders.h 和 libCCPiPhoneSDK.a 到工程中。

(3)设置依赖库



7.开发指南

通过云通讯 IM 进行即时通讯开发，通常遵循如下步骤

（1） 初始化 SDK

（2） 实现通知回调的 delegate。

（3） 连接云通讯

（4） 实现具体的功能。

具体的开发示例如下：

●获取 SDK 实例

```
//获取 ECDevice 单例
[ECDevice sharedInstance];
//设置代理
[ECDevice sharedInstance].delegate = self
```

●实现 delegate

```
//监听与服务器的连接状态ECDelegateBase.h
-(void)onConnected:(ECError *)error
    {
        if (error.errorCode == ECErrorType_KickedOff) {
              NSLog(@"账号已在其它地方登录");
         }
         else if (error.errorCode == ECErrorType_NoError) {
                        NSLog(@"连接成功");
              }
         Else
              NSLog(@"连接失败");
    }

//监听与服务器注销结果
-(void)onDisconnect:(ECError*)error
 {
       NSLog(@"注销登录");
 }

//接收群组通知消息 ECGroupDelegate.h
-(void)onReceiveGroupNoticeMessage:(ECGroupNoticeMessage *)groupMsg
   {
        //对 groupMsg 进行类型判断
         NSLog(@"解散群组、收到邀请、申请加入、退出群组、有人加入、
        移除成员");
 }

//用户接收即时消息和消息的回执报告 ECChatDelegate.h
-(void)onReceiveMessage:(ECMessage*)message
{
       if (message.messageBody.messageBodyType ==
             MessageBodyType_Text){
              NSLog(@"接收文本消息");
       }if (message.messageBody.messageBodyType ==
             MessageBodyType_Media){
              NSLog(@"接收多媒体消息");
       }
        NSLog(@"the messageId is: %@", message.messageId);
        NSLog(@"the sender is: %@", message.from);
   }
   //消息的回执
   -(void)onReceivedReport:(ECReport*)report
   {
        NSLog(@"接收消息回执: %@", report);
   }
```

●连接云通讯

```
// ECLoginInfo登录所需要的信息
ECLoginInfo *loginInfo = [[ECLoginInfo alloc]    initWithAccount: @"VoIP账号"
Password: @"VoIP密码"];
 loginInfo.subAccount = @"子帐号";
 loginInfo.subToken =@"子帐号鉴权";
 loginInfo.serviceUrl =@"sandboxapp.cloopen.com:8883";
 //登录操作
  [[ECDevice sharedInstance] login:loginInfo completion:^(ECError *error) {
        if (error.errorCode == ECErrorType_NoError){
                    NSLog(@"登录成功");
              }}];
```

●单聊、群聊

◆发送文本消息

```
//ECTextMessageBody 为文本消息体（具体属性请参"ECMessageBody.h"）
ECTextMessageBody *messageBody = [[ECTextMessageBody alloc] initWithText:
@"要发送的文本消息"];
//ECMessage消息类，包含发送者，接收者等消息信息（具体属性请参考
  "ECMessage.h"）
ECMessage *message = [[ECMessage alloc] initWithReceiver: @"要发送的人的id"
body:messageBody];
[[ECDevice sharedInstance].messageManager sendMessage:message progress:nil
completion:^(ECError*error, ECMessage *amessage) {
      if (error.errorCode == ECErrorType_NoError) {
                NSLog(@"发送成功");
       }
}];
```

◆发送多媒体消息（包括图片、语音、附件消息）

```
//ECMediaMessageBody 附件消息体,具体属性请参考  ("ECMediaMessageBody.h")
   ECMediaMessageBody * mediaBody = [[ECMediaMessageBody
   alloc]initWithFile:@"文件路径" displayName:@"文件名字"];
   ECMessage *message = [[ECMessage alloc] initWithReceiver:to body:mediaBody];
    //发送消息
   [[ECDevice sharedInstance].messageManager sendMessage:message progress:nil
   completion:^(ECError *error, ECMessage *amessage) {
            if (error.errorCode == ECErrorType_NoError) {
                    NSLog(@"发送成功");
             }
   }];
```

◆下载多媒体消息（包括图片、语音、附件消息）

```
   //下载多媒体消息
   [[ECDevice sharedInstance].messageManager downloadMediaMessage:message
   progress:nil completion:^(ECError *error, ECMessage *message){
        if (error.errorCode == ECErrorType_NoError) {
             NSLog(@"下载成功");
        }}];
```

●群组管理

◆创建群组

```
  //ECGroup 为群组信息类，具体属性请参考 ECGroup.h
  ECGroup * group = [[ECGroup alloc] init];
  group.name = @"群组名字";
  group.declared = @"群公告";
  group.groupId = @"服务器分配的群 id";
   //创建群组
[[ECDevice sharedInstance].messageManager createGroup:group completion:^(ECError
*error, ECGroup*group) {
                if (error.errorCode == ECErrorType_NoError) {
                        NSLog(@"创建成功");
                 }else{
                          NSLog(@"创建群失败");
                 }
}];
```

◆邀请加入群组

```
  //邀请加入群组
[[ECDevice sharedInstance].messageManager inviteJoinGroup: @"当前群组id"
reason:@"邀请理由"members: @"邀请成员" confirm: @"是否需要对方验证"
completion:^(ECError *error, NSString *groupId,NSArray *members) {
                if(error.errorCode ==ECErrorType_NoError) {
                        NSLog(@"邀请成功");
                }
        }];
```

◆主动加入群组

```
  //主动加入群组
  [[ECDevice sharedInstance].messageManager joinGroup: @"申请加入的群组id"
reason: @"申请理由" completion:^(ECError *error, NSString *groupId) {
            if (error.errorCode == ECErrorType_NoError){
                    NSLog(@"申请加入群组成功");
            }
  }];
```

◆解散群组

```
 //解散群组
[[ECDevice sharedInstance].messageManager deleteGroup: @"将
要解散的群组id" completion:^(ECError *error, NSString
*groupId){
     if (error.errorCode == ECErrorType_NoError) {
          NSLog(@"解散群组成功");
      }
 }];
```

◆踢出群组

```
//踢出群组
[[ECDevice sharedInstance].messageManager deleteGroupMembers: @"将成员踢出的
群组id" members: @"被踢出的成员" completion:^(ECError *error, NSString *groupId,
NSArray *members){
            if (error.errorCode ==ECErrorType_NoError){
                    NSLog(@"踢出成功");
            }
        }];
```

8.Demo 介绍

(1)实现功能说明

登　录：输入云通讯的用户名和密码，选择 VoIP 子账号进行登录。

注　册：给出注册引导流程

通讯录：显示子账号联系人

沟　通：显示最近联系人列表，以及消息未读条数

单　聊：实现了点对点收发接收文本消息和多媒体消息。

实现了离线消息的接收

实现消息的发送回执

群　聊：实现了群组里面收发接受文本消息和多媒体消息。

实现了离线消息的接收

群　组：支持创建群组，踢出群组成员，邀请联系人加入，解散群组。

（2）类说明

文件名称	所在位置	文件说明
DeviceDelegateHelper.m	Class/Model	监听程序中所有的回调方法，如接收消息、网络变化监听等
DemoGlobalClass.m	Class/Model	程序中全局变量管理
DeviceChatHelper.m	Class/Model	对发送消息，下载消息等操作进行封装
DeviceDBHelper.m	Class/Model	对数据库进行相关操作，例如：从数据库中获得会话消息
SessionViewController.m	Class/View/Chat	沟通界面的实现
ChatViewController.m	Class/View/Chat/ ChatViewCell	聊天的实现
ContactListViewController.m	Class/View/ Contact	通讯录页面布局的实现
ContactDetailViewController.m	Class/View/ Contact	通讯录二级页面布局的实现
GroupListViewController.m	Class/View/ Group	获取群组列表

CreateGroupViewController.m	Class/View/Group	创建群组
InviteJoinViewController.m	Class/View/Group	邀请联系人加入群组
ApplyJoinGroupViewController.m	Class/View/Group	申请加入群组
SelectViewController.m	Class/View/Group	选择联系人发起会话
DetailsViewController.m	Class/View/Group	群组详情
GroupNoticeViewController.m	Class/View/Group	群通知消息
LoginSelectViewController.m	Class/View/LoginAndMain	登录选择子账号界面
LoginViewController.m	Class/View/LoginAndMain	登录界面
MainViewController.m	Class/View/LoginAndMain	登录成功后主页面
SettingViewController.m	Class/View/Personal	设置页面