

云通讯平台 iOS 开发指南

- 1 概述
 - 1.1 介绍
 - 1.2 开发流程
 - 1.3 平台术语
 - 1.4 参考文档
- 2 VoIP快速体验
 - 2.1 申请测试账号
 - 2.2 环境搭建
 - 2.3 Demo介绍
 - 2.4 导入Demo工程
 - 2.5 配置账号信息
- 3 创建自己的VoIP应用
 - 3.1 SDK介绍
 - 3.2 创建工程
 - 3.2.1 新建工程
 - 3.2.2 导入CCP SDK
 - 3.2.3 配置工程信息
 - 3.3 编写代码
 - 3.3.1 CCP SDK初始化
 - 3.3.2 注册VoIP账号
 - 3.3.3 创建VoIP免费通话(或电话直拨)连接
 - 3.3.4 创建VoIP回拨呼叫连接
 - 3.3.5 接收VoIP通话呼入
 - 3.3.6 接听VoIP通话
 - 3.3.7 挂断(或被叫拒接)VoIP通话
 - 3.4 编译运行和测试
 - 3.5 查看日志
 - 3.6 打包

1 概述

云通讯平台旨在为第三方应用开发者提供丰富完善的注册流程、接入机制、安全策略、管理后台以及不同语言的SDK开发包，为开发者在应用内快速、高效、低成本集成语音业务提供了一站式的服务。本文档旨在为第三方应用开发者在iOS平台下集成CCPIOSSDK来打造语音业务提供参考，文档预期的读者为第三方应用开发人员、平台开发人员、相关技术人员等。

1.1 介绍

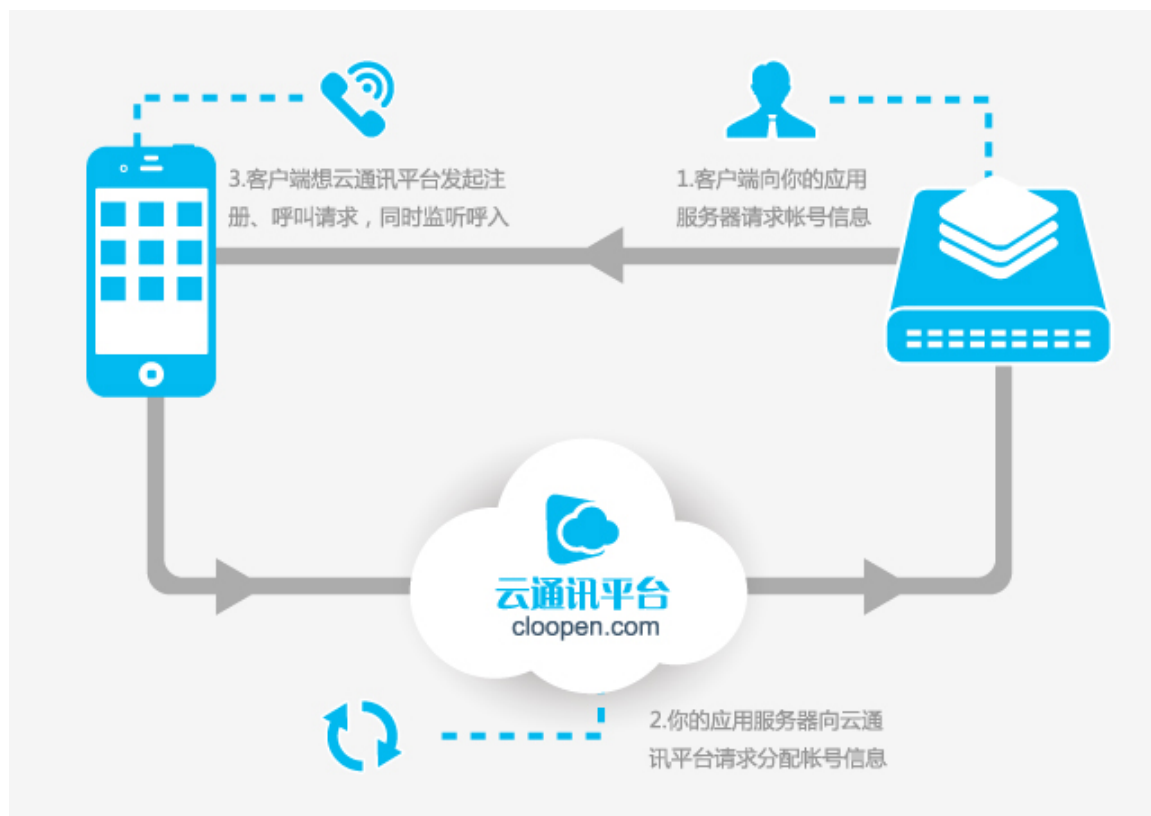
云通讯平台SDK提供了网络通话、视频通话、实时对讲、聊天室、IM等基础能力，REST API除了提供上述功能外，还提供注册账号、创建子账号、营销外呼、语音验证码、各类查询等等。

- 云通讯平台Android SDK 以Java libs的方式提供给Android平台开发人员。
- 云通讯平台iOS SDK 以C++静态库的方式提供给iOS平台开发人员。
- REST API 可通过HTTPS GET、POST方式访问。

1.2 开发流程

云通讯平台作为通讯能力的云计算PAAS平台，将传统电信网络的通讯能力、基于IP的通讯能力，通过开放API以及SDK的方式提供给开发者和商家，协助开发者快速、高效、低成本打造融合通讯能力的产品。

云通讯平台能力开发方式，如下图所示：



这是常见的接入方式，通过3个主要步骤来完成：

1. 您的客户端应用集成云通讯平台提供的SDK，同时客户端向您的应用服务器请求分配VoIP账号信息；
2. 您的应用服务器通过调用云通讯平台REST API得到用户账号并返回给您的客户端应用；
3. 客户端应用通过调用SDK API发起呼叫请求或者监听呼入；

1.3 平台术语

- AS: Application Server, 应用服务器, 第三方开发者搭建的服务器, 和云通讯平台交互, 可以查询管理账户, 也可以拨打电话、回拨电话、发送短信等。
- CCP: Cloud Communication Platform, 云通讯平台。
- CCP SDK: CCP Software Development Kit, 云通讯平台软件开发包。
- QML: Quick Markup Language, 快速标记语言, 一组当接收到来电或短信时告诉云通讯平台如何处理的指令。
- Rest: REpresentational State Transfer, 表征状态转移, 是一种针对网络应用的设计和开发方式, 可以降低开发的复杂性, 提高系统的可伸缩性。
- Rest服务器: 为应用服务器提供功能接口的服务器。
- VoIP: Voice over Internet Protocol, 基于网络协议的语音实时传输。
- VoIP帐号: 由VoIP服务器为子帐号分配的帐号。
- 开发者: 特指云通讯平台应用的第三方开发者。
- 主帐号: 第三方开发者在云通讯平台开发者网站上注册后分配得到的账号。
- 子帐号: 第三方开发者可使用主帐号调用REST接口获取的账号。

1.4 参考文档

- [《云通讯平台REST技术文档》](#)
- [《云通讯平台Android技术文档》](#)
- [《云通讯平台IOS技术文档》](#)

2 VoIP快速体验

在[云通讯平台](#)注册账号, 创建Demo账号, 并下载获取CCPVoiDemo程序(具体过程请参考以下内容)。在Demo程序中, 演示了云通讯平台提供的网络通话、视频通话、实时对讲、聊天室等功能。

2.1 申请测试账号

在云通讯平台上获取Demo账号信息, 须注册后创建Demo, 即可获得开发VoIP所需的测试帐号信息。

测试账号信息内容有: 主账号、主账号密码、子账号、子账号密码、VoIP账号、VoIP账号密码, 应用ID

2.2 环境搭建

Mac OS X 10.7 (Lion)及以上版本;

XCode4.2及以上版本, [下载](#)。

2.3 Demo介绍

1. 下载: 在Demo账号信息页面, 提供了Android和IOS平台下的Demo下载, 请选择IOS版下载
2. CCPVoipDemo功能介绍, Demo演示了CCP SDK的API接口调用, 主要实现的功能:
 - 免费电话: 需要对方的VoIP账号, 双方进行的网络P2P通话, 免费通话
 - 电话直拨: 需要对方的手机号, 主叫接入网络电话, 被叫接入普通电话的网络通话
 - 回拨呼叫: 需要对方的手机号, 双方都会接入普通电话网络进行通话
3. CCPVoipDemo工程文件结构说明:
 - Products: 应用生成的app
 - Frameworks: 包含工程需要依赖的资源项, 主要是系统和需要依赖的库资源
 - CCPVoipDemo: 包含整个工程需要完成的代码文件: 视图控制类的实现文件、resource文件夹(账号文件, 图片资源文件等)、ccp_sdk文件夹(SDK库文件)、Supporting Files文件夹(创建工程自动生产的工程相关文件)

2.4 导入Demo工程

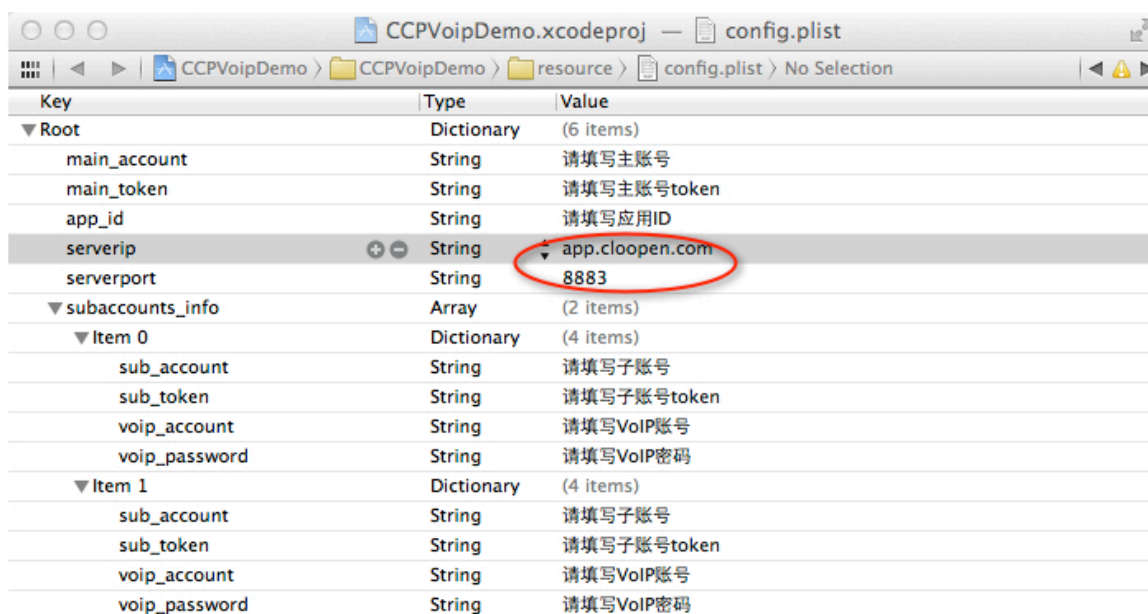
解压下载的CCP_PHONE_DEMO_IOS.rar文件，在解压缩的文件夹中，双击CCPVoipDemo.xcodeproj文件，在XCode中打开工程，即可对Demo进行其他操作。

2.5 配置账号信息

打开resource\config.plist文件，将申请测试账号时获取的Demo账号信息，依次输入配置文件中，

(注：如果想更改设备上应用的测试账号，请在应用的共享文档下更改config.plist文件内容)

如图所示：



The image shows the Xcode interface for editing the config.plist file. The table below represents the data visible in the editor:

Key	Type	Value
Root	Dictionary	(6 items)
main_account	String	请填写主账号
main_token	String	请填写主账号token
app_id	String	请填写应用ID
serverip	String	app.cloopen.com
serverport	String	8883
subaccounts_info	Array	(2 items)
Item 0	Dictionary	(4 items)
sub_account	String	请填写子账号
sub_token	String	请填写子账号token
voip_account	String	请填写VoIP账号
voip_password	String	请填写VoIP密码
Item 1	Dictionary	(4 items)
sub_account	String	请填写子账号
sub_token	String	请填写子账号token
voip_account	String	请填写VoIP账号
voip_password	String	请填写VoIP密码

serverip的配置，如果是沙盒环境，请填写sandboxapp.cloopen.com；正式环境，填写app.cloopen.com。一定不要加前缀https或http等。

账号信息输入完之后，现在可以运行Demo，体验云通讯平台的基础功能。

3 创建自己的VoIP应用

这一节是为了让开发者能够用最少的代码量和时间，来实现基本的VoIP通话功能。


3.1 SDK介绍

1. SDK下载：从云通讯平台下载VoIP的IOS SDK
2. SDK文件说明：SDK文件放在文件夹ccpsdk中，其中包含四个文件：
 - CCPCallService.h为应用调用的函数头文件

- CallEvent.h为SDK的代理函数头文件
- libccpapisdk.a为整个SDK库文件
- CCPSDKBundle.bundle为SDK里面的资源文件

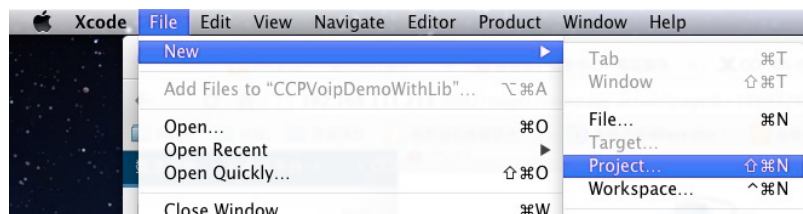
3.2 创建工程

3.2.1 新建工程

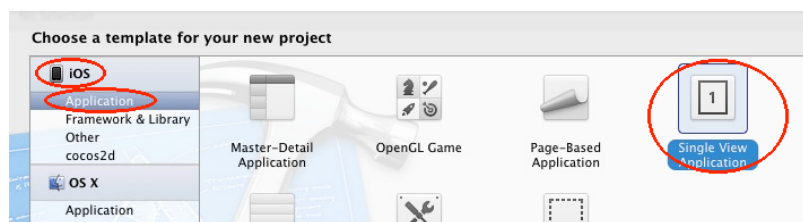
- 创建一个iPhone工程需要在Mac系统下面安装Xcode的软件，Xcode安装完成后可以在桌面的快速启动栏中的"Launchpad"中或者应用程序中找到一个图标，看到后单击或者双击即可打开软件，Xcode版本最小需要4.2。
- 打开Xcode后的界面可以看到几个选项，创建工程需要选择"Create a new Xcode project",如下图所示：



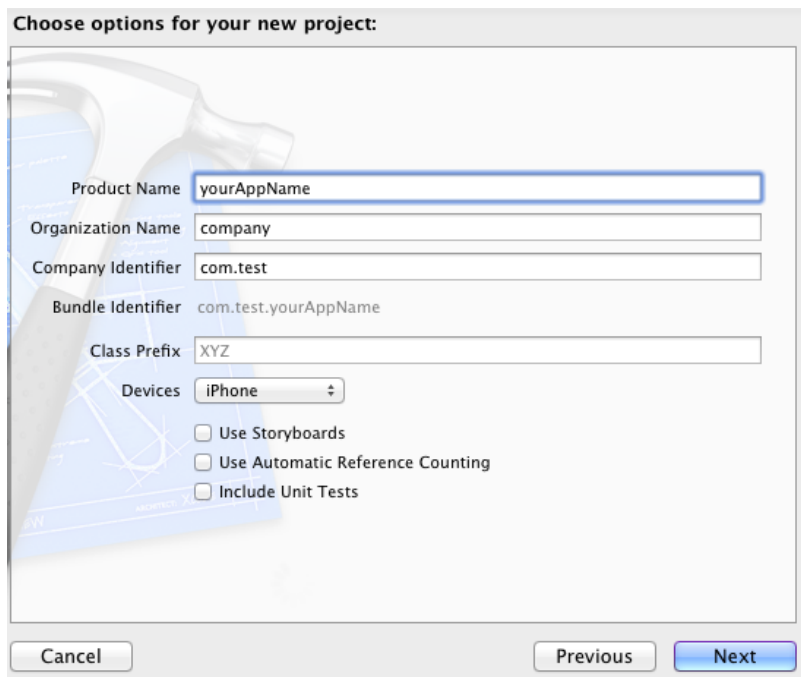
- 也可以根据菜单创建工程，依次选择XCode菜单栏中的File->New->Project，如下图所示：



- 选择完后弹出的界面中选择左边的"iOS"下面的"Application",然后选择右边的"Single View Application"进行双击或者点击右下角"Next",如下图所示：



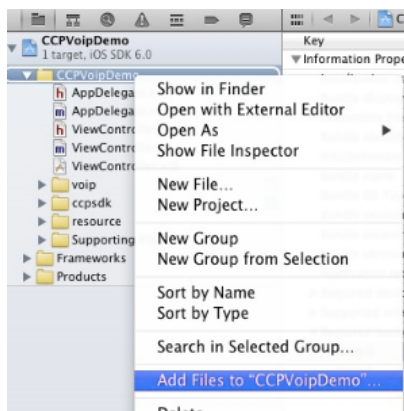
- 在完成后弹出的界面中的"Product Name"中填写你的工程名，如下图所示：



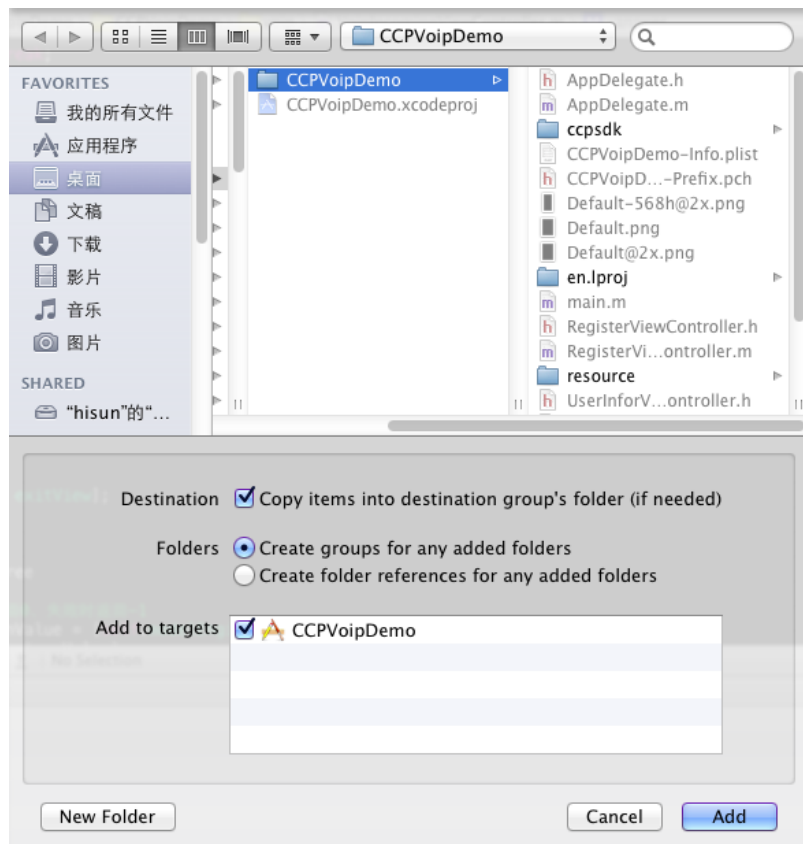
- 添加完成后点击右下角的"Next"，在弹出的对话框中选择你需要把工程放置的位置。这样一个工程就初步的创建完成。

3.2.2 导入CCP SDK

- 创建完成工程后，把SDK包里面的ccpsdk文件夹拷贝到新创建的工程路径下面，然后在工程目录结构中，右键选择Add Files to "(工程名)"，在弹出的对话框中选择新创建的工程文件夹下的ccpsdk这个文件夹(也就是前面拷贝的文件夹)。或者将这个文件夹拖入XCode工程目录结构中，



- 在弹出的界面中勾选Copy items into destination group's folder(if needed), 并确保Add To Targets勾选相应的target。相应的操作请看下图中所示：



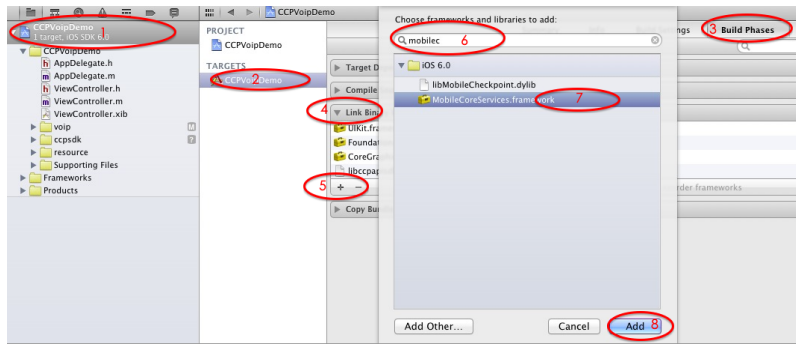
3.2.3 配置工程信息

- 添加依赖框架(Frameworks)

CCP SDK的实现, 依赖了一些系统框架, 在开发应用时, 要在工程里加入这些框架。开发者首先点击工程右边的工程名,然后在工程名右边依次选择TARGETS->Build Phases->Link Binary With Libraries, 展开Link Binary With Libraries后点击展开后下面的"+"来添加下面的依赖项:

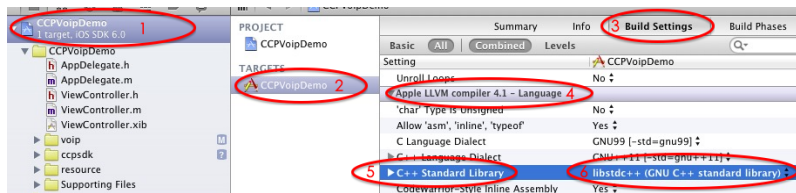
```
libz.dylib
libxml2.dylib
MobileCoreServices.framework
SystemConfiguration.framework
AvFoundation.framework
CFNetwork.framework
AudioToolbox.framework
Foundation.framework
CoreGraphics.framework
CoreTelephony.framework
CoreMedia.framework
QuartzCore.framework
OpenGL.framework
```

添加步骤如图所示:



• 编译器设置

在xcode4.5环境下，首先点击工程右边的工程名,然后在工程名右边依次选择TARGETS、Build Settings，在下面的窗口中找到"Apple LLVM compiler 4.1 - Language"中的"C++ Standard Library",在后面的选择框中选择"libstdc++(GNU C++ standard library)"选项，如果默认是则不需要修改。步骤如下图所示：



• 工程属性设置

一般的iOS程序进入后台后会被系统挂起，就会停止执行，不能执行任何操作。

1. 从iOS4开始，苹果增加了特性，很好的支持了VoIP功能：

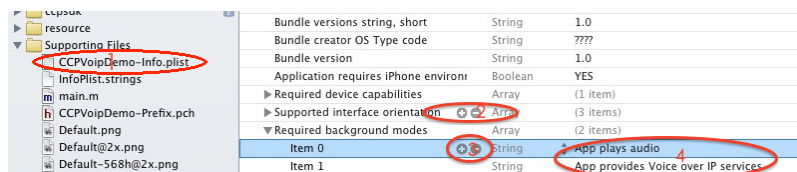
- 苹果支持应用可以在后台播放和录制声音；
- 苹果支持网络托管，保证应用在后台时，还能保持网络连接，能接收到来电；
- 应用可以设置一个超时处理，程序在后台运行时，周期性地唤醒应用，保证客户端和服务端有长连接，使网络不断开。

2. CCP SDK封装了这些特性，保证了在iOS平台上，有很好的VoIP体验。

3. 开发者需要修改配置文件，这样iOS工程才能支持这些特性。

- 在工程名的文件夹下面的Supporting Files文件夹中找到并且选择(工程名)Info.plist
- 在右边出现的窗口中添加Key: Required background modes，在下面添加两个项: App plays audio和App provides Voice over IP services。

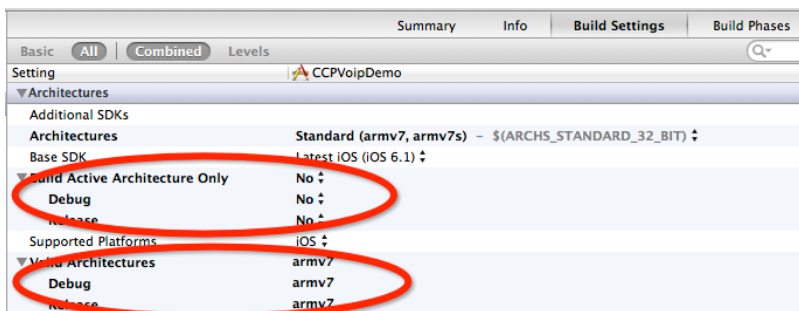
如图所示：



4. 3.3版本以后为了精简库文件，armv7s的编译需要进行设置，使用iPhone5及以上机型编译时，需要修改工程配置文件：

- 修改Build Active Architecture Only选项为NO。
- 修改Valid Architectures为armv7 (双击该项打开，然后删除掉多余的选项，仅保留armv7)。

如图所示：



3.3 编写代码

介绍代码的实现过程，也可参考Demo的代码实现

注意事项:

- 调用SDK文件要以.mm为后缀
- 要包含头文件CCPCallService.h和CallEvent.h

3.3.1 CCP SDK初始化

1. 初始化SDK的代码

```
{
    // 初始化CCP SDK，并传入代理实现的类实例
    CCPCallService *ccpServer = [[CCPCallService alloc] initWithDelegate:self];

    // 另一个方法
    // CCPCallService *ccpServer = [[CCPCallService alloc] init];
    // [ccpServer setDelegate:self];
}
```

注意事项: 必须要设置代理

3.3.2 注册VoIP账号

1. 注册VoIP

```
{
    [ccpServer connectToCCP:rest_addr onPort:rest_port
    withAccount:accountStr withPsw:passwordStr
    withAccountSid:accountSid withAuthToken:authToken];
}
```

2. 注册事件相关代理函数

- 注册成功:

```
// 与云通讯平台连接成功
-(void)onConnected
{
    // 注册成功后的处理代码
    // TODO
}
```

- 注册失败:

```
// 与云通讯平台连接失败或连接断开
-(void)onConnectError:(NSInteger)reason withReasonMessage:(NSString *)reasonMessage
{
    // 注册失败后的处理代码
    // TODO
}
```

3.3.3 创建VoIP免费通话(或电话直拨)连接

1. 创建呼出代码

```
{
    // 拨打免费通话(对方VoIP账号) 或 电话直拨(对方电话号码)
    NSString *callid = [cppService makeCallWithType:EVoipCallType_Voice
andCalled:@"对方电话或VoIP账号"];
    if(callid)
    {
        // 创建成功
        // TODO
    }
    else
    {
        // 创建失败处理代码
        // TODO
    }
}
```

注意事项: callid不能为nil, 必须存在; 因为callid是作为某一个的电话id, 在后续的操作中继续使用。

2. 创建成功并连接被叫过程中代理函数

```
-(void)onCallProceeding:(NSString *)callid
{
    // 创建成功并在连接对方过程
    // TODO
}
```

3. 连接被叫成功

```
-(void)onCallAlerting:(NSString *)callid
{
    // 连接被叫成功
    // TODO
}
```

4. 对方接听的代理函数

```
-(void)onCallAnswered:(NSString *)callid
{
    // 对方已接听
    // TODO
}
```

5. 呼叫失败(被叫拒接, 被叫忙等原因)的代理函数, 可参考[错误码](#)查找失败原因

```
-(void)onMakeCallFailed:(NSString *)callid withReason:(int)reason
{
    // 呼叫失败, 可根据reason查找错误原因
    // TODO
}
```

6. 通话过程中, 对方挂断的代理函数

```
-(void)onCallReleased:(NSString *)callid
{
    // 通话过程中, 对方挂断电话
    // TODO
}
```

3.3.4 创建VoIP回拨呼叫连接

1. 创建VoIP回拨呼叫

```
{  
    // 创建fromPhone回拨呼叫到toPhone  
    [cppService callback:src withTOCall:dest];  
}
```

说明：回拨呼叫成功后，src首先接到VoIP落地电话，接听后，dest才接到VoIP落地电话。

2. 回拨结果代理函数

- 回拨成功，等待系统回拨电话

```
-(void)onCallback  
{  
    // 回拨成功  
    // TODO  
}
```

- 回拨失败，原因可参考[错误码](#)

```
-(void)onCallbackError:(NSInteger)reason withReasonMessge:(NSString *)reasonMessage  
{  
    // 回拨失败  
    // TODO  
}
```

3.3.5 接收VoIP通话呼入

实现代理函数

```
-(void)onIncomingCallReceived:(NSString*)callid withCallerAccount:(NSString *)caller withCallerPhone:(NSString  
*)callerphone withCallerName:(NSString *)callername withCallType:(NSInteger)calltype  
{  
    // 有VoIP电话呼入处理  
    // TODO  
}
```

说明：在该代理函数中，能够得到来电者的电话，昵称等信息

3.3.6 接听VoIP通话

1. 接听代码

```
{
    // 接听VoIP电话, 使用callid参数标识接听某个具体的VoIP电话
    NSInteger ret = [cppService acceptCall:callid withType:callType];
    if (ret == 0)
    {
        // 接听过程中
        // TODO
        if (callType == 0)
        {
            //语音通话处理过程
        }
        else if (callType == 1)
        {
            //视频通话处理过程
        }
    }
    else
    {
        // 接听失败
        // TODO
    }
}
```

2. 接听成功的代理函数

```
-(void)onCallAnswered:(NSString *)callid
{
    // 接听成功
    // TODO
}
```

3.3.7 挂断(或被叫拒接)VoIP通话

来电拒接, 通话过程中的挂断函数:

```
{
    [cppService releaseCall:callid];
}
```

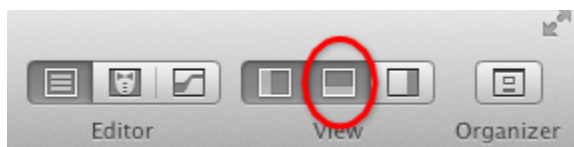
3.4 编译运行和测试

在用Xcode打开工程后, 看是需要用真机还是用模拟器运行应用, 默认是模拟器下编译运行的, 单击左上角的三角符号就可以进行应用运行, 如果是真机则需要你插上真机连接电脑, 然后在最上面有单击"iPhone 6.0 Simulator"(6.0为模拟器的版本)后可以看到显示你iPhone的机器名字, 并且选择后点击左上角的三角符号进行运行即可。如下图所示:

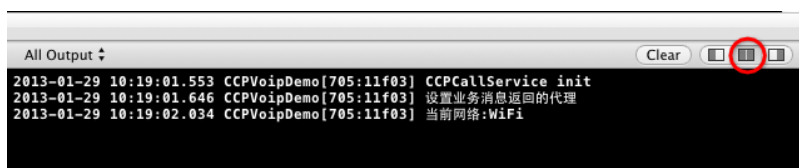


3.5 查看日志

查看编译器日志
选择xcode右上角区的的按钮

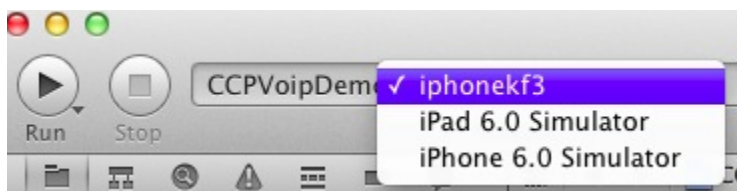


在xcode下方区域看见日志，如果没有可以点击下方区域中按钮

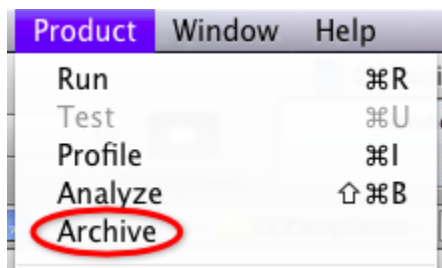


3.6 打包

首先要从弹出的列表中选择第一项

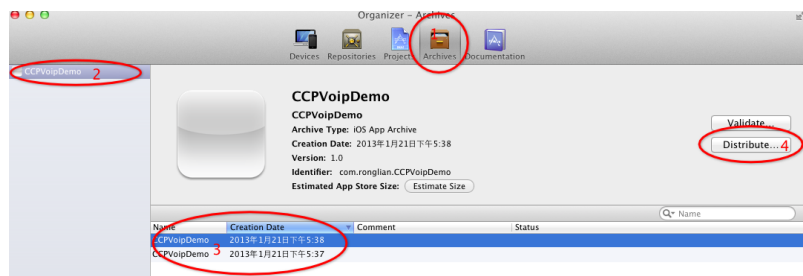


然后选择xcode的菜单Product->Archive项

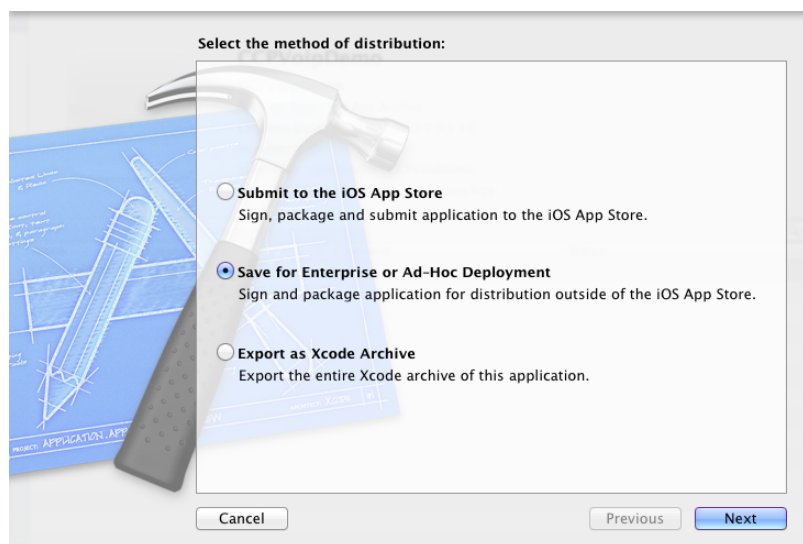


在弹出的窗口中如果没有显示下列窗口依次选择上方Archives，在左侧列表选中需要打包的项目名，在右下窗口中根据创建日期的选择包，

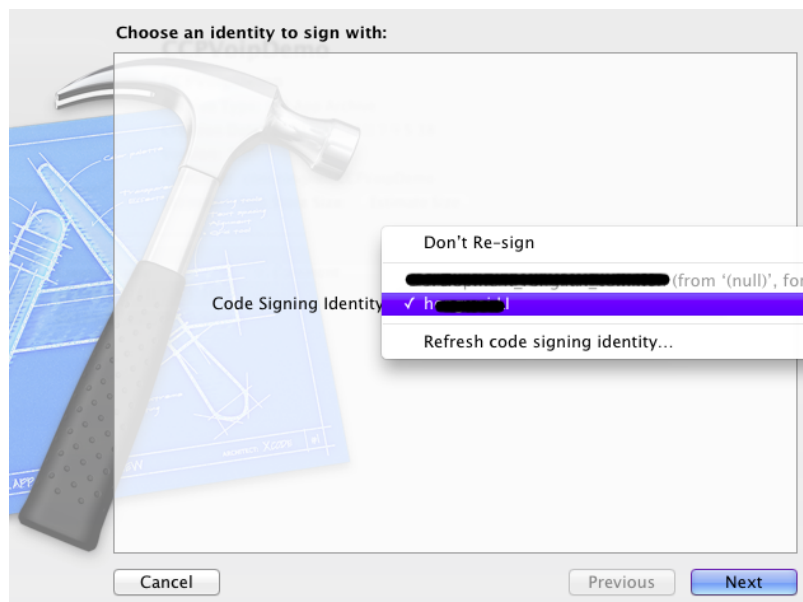
单击Distribute...按钮



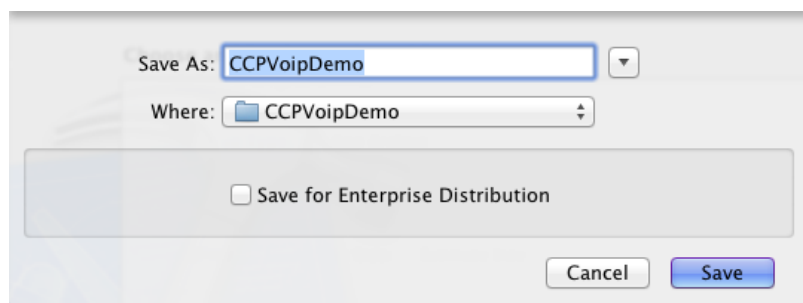
根据需要选择，测试请选择第二项



在下列窗口中请选择程序的签名证书



重命名包名和选择要保持的路径



点击"Save"按钮打包结束