AnyChat SDK for iOS

开发流程指南

(版本：V6.0)

广 州 佰 锐 网 络 科 技 有 限 公 司

Guangzhou BaiRui Network Technology Co.,Ltd.

http://www.bairuitech.com http://www.anychat.cn

2016 年 4 月

目录

一、简介

AnyChat SDK(AnyChat 音视频互动开发平台)是一套跨平台的音视频即时通讯解决方案，基于先进的 H.264 视频编码标准、AAC 音频编码标准与 P2P 技术，支持高清视频，整合了佰锐科技在音视频编码、多媒体通讯领域领先的开发技术和丰富的产品经验而设计的高质量、宽适应性、分布式、模块化的网络音视频互动平台。

基于 iOS 的客户端 SDK 应用于 iOS 5.1 以上版本的设备，您可以通过该套 SDK API 接口实现在 iOS 平台快速开发基于音视频通讯交互功能的 App 程序，主要提供的功能如下：

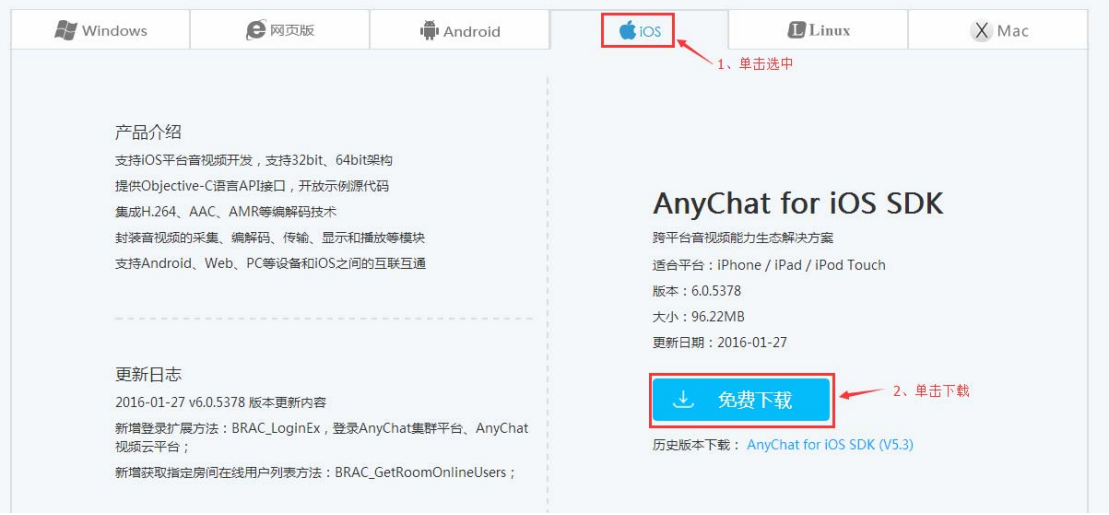
- 音视频即时通讯：提供语音、视频一对一、一对多的实时通讯，支持高清视频和高品质音频效果。
- 录像：支持针对单个人的音视频录制、整个视频通话过程内容的合成音视频录制以及集中服务器保存录制
- 抓拍：可对本地视频和正在视频的对象进行抓拍；
- 文字聊天：支持多用户之间的文字交流；
- 透明通道：提供客户端之间、客户端跟服务器之间的数据通讯能力；
- 文件传输：支持客户端直接、客户端跟服务器之间的文件传输功能，支持断点续传；
- 动态设置音视频参数：提供音视频参数设置的接口，可以根据需要动态设置分辨率、码率、帧率等视频参数，满足各种应用场景的需求；
- 外部音视频输入：支持非标准采集设备以外的音视频源输入，满足更多的应用场景；
- 集成第三方外部音视频编解码器：可集成第三方音视频编解码器，满足特殊环境下面的硬件编解码要求；
- 业务排队：提供自定义营业区域、队列功能，实现客户排队、坐席为队列中客户提供服务的功能；

1.1 面向的读者

《AnyChat SDK for iOS开发流程指南》文档是提供给具有一定的iOS编程经验和了解面向对象概念的读者使用，不要求具备音视频开发方面的经验。您在使用遇到任何问题，都可以通过访问bbs.anychat.cn反馈给我们。

1.2 获取AnyChat SDK for iOS

您可在AnyChat的产品官方网站下载到最新版的AnyChat for iOS SDK，下载地址为：http://sdk.anychat.cn/html/download.html，如下图所示：



AnyChat for iOS SDK 包里面提供了 demo 程序的编译程序、开发指南、demo 程序源码和 SDK 文件，其解压之后的目录结构如下所示：

|----bin AnyChat SDK 演示程序（安装包）

|----doc 客户端开发指南

|----src Demo 程序源代码（XCode 5.1.1 工程）

|----sdk 客户端 SDK 引用文件

1.3 技术支持

在您使用本 SDK 的过程中，遇到任何困难，请与我们联系，我们将热忱为您提供帮助。

您可以通过如下方式与我们取得联系：

1、 在线论坛：http://bbs.anychat.cn/
2、 知识中心：http://www.anychat.cn/faq/
3、 官方网站：http://www.anychat.cn
4、 电子邮件：service@bairuitech.com
5、 24 小时客服电话：+86 （020） 85276986、38109065、38103410

二、编写说明

本指南的编写是为了帮助使用 AnyChat for iOS SDK 的用户快速地搭建 SDK 开发环境、熟悉 SDK 开发流程、掌握 SDK 开发功能接口而编写的。

其中“工程准备”、“基本流程”、“音视频交互”三章的内容是基于 src\HelloAnyChat 目录中提供的 HelloAnyChat 工程来编写说明的，涉及到的开发环境配置、以及相关代码说明可以参考 HelloAnyChat 工程源码。

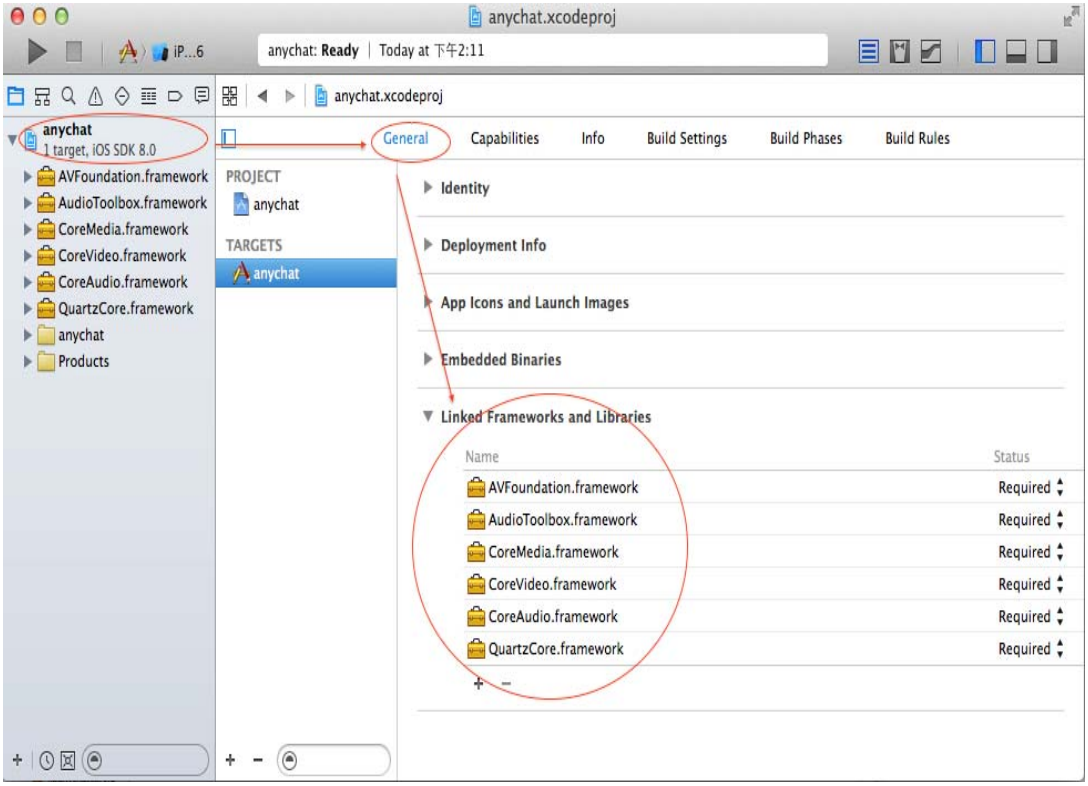
其中“业务排队、视频呼叫”章节内容是基于 src\AnyChatQueue 目录中提供的 AnyChatQueue 工程来编写说明的，涉及到的开发环境配置、以及相关代码说明可以参考 AnyChatQueue 工程源码。

三、工程准备

iOS 开发需要在 MAC OSX 系统环境下进行，开发工具有很多，开发者可根据自己的喜好进行选择。在此，我们推荐开发者使用 XCode 作为自己的开发工具，本套开发指南也是针对 XCode 开发环境下进行编写的。在 XCode 中新建一个 iOS 工程，对工程进行以下配置，搭建 AnyChat 的开发环境。

3.1 添加开发框架

1) 点击在 XCode 工程左侧资源管理器中的工程图标。在右侧菜单 TARGETS—>General—>Link Frameworks and Libraries 的路径里，点击“+”号增加系统框架。如下图所示：

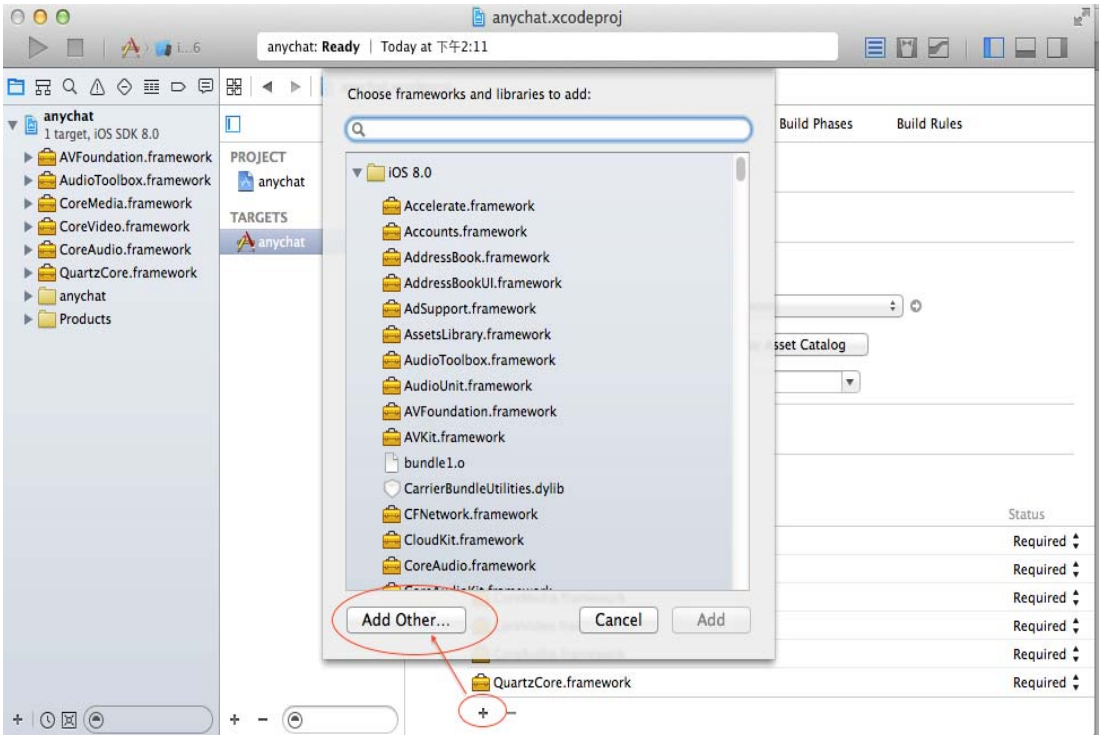


2) 工程需要用到的框架包括有：

AVFoundation.framework AudioToolbox.framework
QuartzCore.framework CoreMedia.framework
CoreVideo.framework CoreAudio.framework

3.2 导入库文件

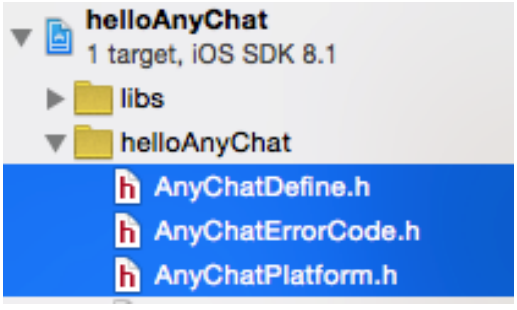
1) 在添加开发框架步骤的同一处，点击“Add other”选项导入 AnyChat SDK 库文件。库文件在 SDK 开发包的“sdk\libs\”和“sdk\libs\audioprocess”目录里。如下图所示：



2) 导入 C++的静态库 libc++.dylib、libstdc++.dylib 和 libstdc++.6.dylib。

3.3 导入SDK API头文件

1) 把开发包“\sdk\include”目录里的 SDK API 头文件添加到工程资源管理器中，操作后的工程目录如下图所示：



2) 在工程中使用到 AnyChat API 的代码里，引用 SDK API 头文件，参考代码如下：

```
#import "AnyChatPlatform.h"
#import "AnyChatDefine.h"
#import "AnyChatErrorCode.h"
```

四、基本流程

在工程准备好了之后，需要实现以下基本流程，才能调用音视频交互等其他功能接口。

4.1 准备一个AnyChatPlatform对象

AnyChatPlatform 类是 SDK 的核心类，提供各种功能接口，如连接服务器、登录、进入房间、操作音视频等，在使用这些功能接口构建应用之前，需要准备一个 AnyChatPlatform 对象。

新建一个 AnyChatPlatform 对象，参考代码如下：

```
//新建核心类对象
AnyChatPlatform *anychat = [AnyChatPlatform new];
```

4.2 监听基本事件

运用通知中心机制，实现监听“连接服务器、用户登录、进入房间、与服务器网络连接”等事件。在需要接收的类中实现以下两步。

(1) 在所需要监听的类的.h 头文件里引用 AnyChat 通知消息代理，参考代码如下：

```
//AnyChat通知消息代理
@interfaceAnyChat : UIViewController<AnyChatNotifyMessageDelegate>
```

使用 AnyChat 通知消息<AnyChatNotifyMessageDelegate>异步消息事件协议时，必须实现处理回调信息的 7 个方法，具体方法如下：

```
// 连接服务器消息
- (void) OnAnyChatConnect:(BOOL) bSuccess;
// 用户登陆消息
- (void) OnAnyChatLogin:(int) dwUserId : (int) dwErrorCode;
// 用户进入房间消息
- (void) OnAnyChatEnterRoom:(int) dwRoomId : (int) dwErrorCode;
// 房间在线用户消息
- (void) OnAnyChatOnlineUser:(int) dwUserNum : (int) dwRoomId;
// 用户进入房间消息
- (void) OnAnyChatUserEnterRoom:(int) dwUserId;
// 用户退出房间消息
- (void) OnAnyChatUserLeaveRoom:(int) dwUserId;
// 网络断开消息
- (void) OnAnyChatLinkClose:(int) dwErrorCode;
```

(2) 在监听回调类的-(void)viewDidLoad 方法里注册通知中心，并实现消息观察者方法和设置消息回调事件接收者，参考代码如下：

```
@property (strong, nonatomic) AnyChatPlatform *anyChat;
- (void)viewDidLoad
{
      [super viewDidLoad];
      //注册通知中心
      [[NSNotificationCenter        defaultCenter]        addObserver:self
selector:@selector(AnyChatNotifyHandler:)            name:@"ANYCHATNOTIFY"
object:nil];
      //初始化SDK
      anyChat = [[AnyChatPlatform alloc] init];
      //AnyChat通知消息代理(回调事件接收者)
      anyChat.notifyMsgDelegate = self;
}
//消息观察者方法
- (void)AnyChatNotifyHandler:(NSNotification*)notify
{
      NSDictionary* dict = notify.userInfo;
      [anyChat OnRecvAnyChatNotify:dict];
}
```

4.3 初始化SDK

加载资源，应用程序中只需要执行一次，其他的功能接口都必须在初始化之后才能正常使用，参考代码如下：

```
//初始化AnyChatSDK
@interfaceAnyChat[AnyChatPlatform InitSDK:0];
```

4.4应用签名

4.4.1 应用 Id

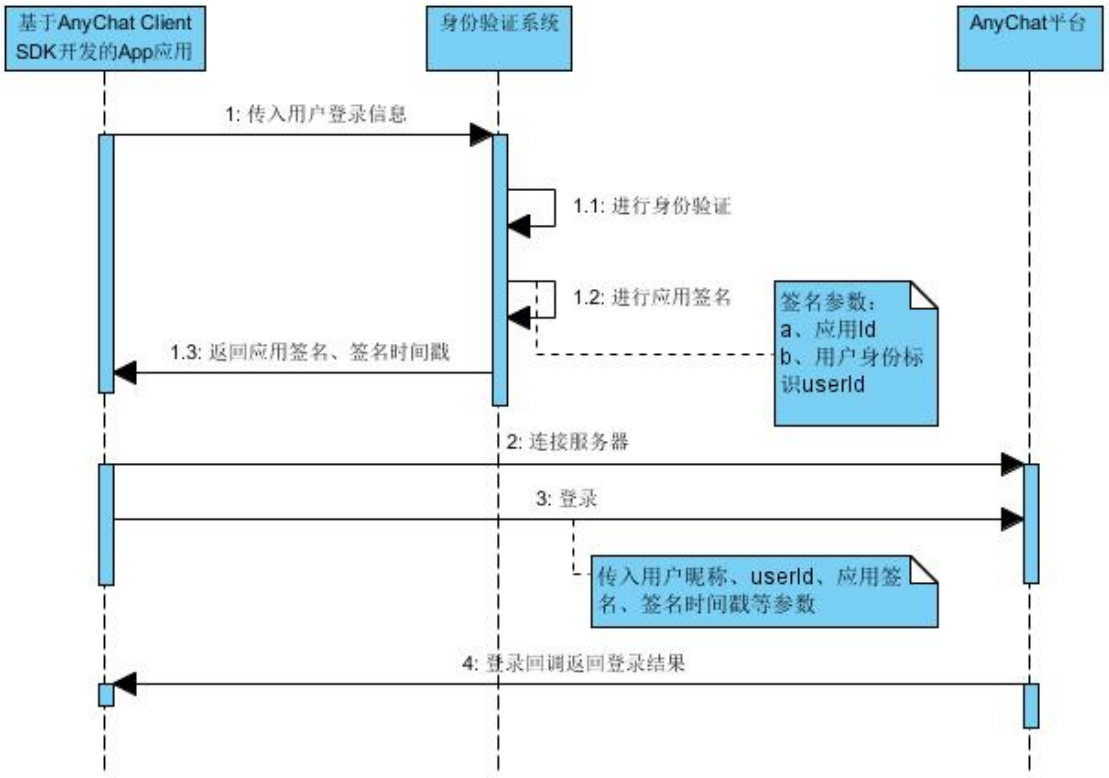
在 AnyChat 视频云平台或 AnyChat 集群 Web 控制台中创建一个音视频应用，每个创建的应用都会有一个 AppGuid 属性值，该属性值即为应用 Id，如："78304AED-7DA7-4E14-92D9-E527AB1EA79A"。各平台的客户端通过该应用 Id 才能正常使用 AnyChat 视频云平台提供的服务或 AnyChat 服务集群。

4.4.2 应用签名使用场景

在有更高安全级别的需求下，可以采用此非对称算法进行身份验证签名进行登录。在进行签名之前，开发人员可以通过 AnyChat 视频云平台或集群 Web 控制台生成应用的一对私钥、公钥信息，需要开发人员将私钥保存为一个文件；也可以由开发人员通过工具自己生成一对私钥、公钥文件。需要将私钥文件部署在身份验证系统中、公钥文件部署到 AnyChat 平台指定的位置。

当业务系统的使用用户登录系统时，需要在业务系统后端的身份验证系统中对登录的用户信息进行身份验证，验证成功之后传入需要签名的参数数据调用 AnyChat 提供的签名接口进行签名，签名接口会返回应用签名和签名时间戳数据，返回给客户端，在客户端调用 AnyChat 的 BRAC_LoginEx 接口进行登录，AnyChat 平台会对签名数据进行校验，通过登录回调返回登录结果。

应用签名过程如下图所示：



4.5 连接、登录服务器

在AnyChat中，登录系统需要调用BRAC_Connect、BRAC_Login、BRAC_LoginEx等API接口来实现。其中BRAC_Connect为连接服务器的接口；BRAC_Login、BRAC_LoginEx为登录服务器的接口，根据不同的应用场景使用不同的登录接口，其中：BRAC_Login接口实现输入用户名、密码来登录；BRAC_LoginEx接口适用于应用签名登录，应用签名流程见4.4.2节内容介绍。

如果在系统或应用中设置了允许用户以游客的身份进行登录，则登录接口在不传入用户密码或用户身份签名数据的参数情况下，AnyChat 将不验证用户密码或用户身份签名，同样可以允许用户登录系统。

4.5.1 登录 SDK 平台

1) 调用 BRAC_Login 接口登录

单台部署的 SDK 核心服务器只支持一个应用连接到服务器中。应用 Id 可以通过应用常量：BRAC_SO_CLOUD_APPGUID 进行赋值，调用 BRAC_SetSDKOption 接口实现应用 Id 设置。

连接、登录代码如下所示：

```
// 连接服务器,第一个参数为你需要连接的 AnyChat 核心服务器地址，如果您部署 AnyChat 核心服务器的地址为 192.168.1.8，则传入这个地址；第二个参数为端口号
[AnyChatPlatform Connect:@"demo.anychat.cn":@"8906"];
//登录服务器
[AnyChatPlatform Login:userName:@""];
```

我们对外公开测试服务器地址为 demo.anychat.cn。

在客户端调用登录接口后，将会触发业务服务器的回调函数“BRAS_VerifyUser_CallBack”，由业务层服务器完成用户的身份验证工作；客户端根据 OnAnyChatConnect、OnAnyChatLoginSystem 回调函数的返回值来判定是否通过身份验证，在此函数中进行登录成功与否的逻辑处理。

在上述调用登录接口时，也可以放在 OnAnyChatConnect 函数中处理。

```
// 客户端连接服务器，bSuccess表示是否连接成功
- (void) OnAnyChatConnect:(BOOL) bSuccess
{
    if (bSuccess)
    {
        [AnyChatPlatform Login:theUserName.text : @""];
        theStateInfo.text = @"• Success connected to server";
    }
}
```

2) 调用 BRAC_LoginEx 接口登录

在调用BRAC_LoginEx接口登录系统时，需要传入AppId参数，该参数由用户自己根据实际情况设置应用Id，应用签名流程见4.4.2节内容介绍。

```
// 连接服务器
[AnyChatPlatform Connect:@”demo.anychat.cn”:@”8906”];
//登录服务器
 [AnyChatPlatform   LoginEx:@"userName"   :1001   :@""   :   1452580453:@"
TBK5M9FPcDfZ8/bawWDoYW82J9yAw1Sa8GURKf4fgRrOhYGk/ODOHODNih4ox2F4
gXPdq5UK1kRGbu114dNSK2ITLfhupQ5EMxIF9Xy1uR/z7mIWaViMGKMDLNd9b4p6r
9b3vDBmh1OXV9oknaEELLQLiw6Ka3JIj/qTmKKPD6E=" :@""];
```

如果部署了 AnyChat 服务集群，客户端接入服务集群的接口调用方法同上所述，只是服务器地址改为部署的寻址服务器的 IP 地址。如何部署 AnyChat 服务集群请参见《AnyChat 服务集群部署说明文档》。

4.5.2 登录 AnyChat 视频云平台

AnyChat视频云平台支持多个云应用的接入。在AnyChat视频云平台创建的每个云应用都会有一个AppId，见4.4.1节内容介绍。

1) 调用 BRAC_Login 接口登录

需要调用“BRAC_SetSDKOption”接口设置连接的应用 Id。代码如下：

```
if (theGuid.text.length != 0) {
            [AnyChatPlatform SetSDKOptionString:BRAC_SO_CLOUD_APPGUID :theGuid.text];
}
```

我们云平台的接入服务器地址为 cloud.anychat.cn。

连接视频云平台服务器、登录服务器接口调用代码如下所示：

```
// 连接服务器
[AnyChatPlatform Connect:@” cloud.anychat.cn”:@”8906”];
//登录服务器
[AnyChatPlatform Login:userName:@""];
```

在客户端调用登录接口后，将会触发业务服务器的回调函数“BRAS_VerifyUser_CallBack”，由业务层服务器完成用户的身份验证工作，客户端根据 OnAnyChatLogin 回调函数的返回值来判定是否通过身份验证，在此函数中进行登录成功与否的逻辑处理。

注意：该登录方式需要将业务服务器接入到 AnyChat 视频云平台中。

2) 调用 LoginEx 接口登录

在调用BRAC_LoginEx接口之前需要先进行应用签名，具体应用签名流程见4.4.2节内容介绍。

```
// 连接服务器
[AnyChatPlatform Connect:@” cloud.anychat.cn”:@”8906”];
//登录服务器
[AnyChatPlatform
LoginEx:userName :1001 :@"" :@"C782383B-FC3C-4726-815B-9D6117BF681A" :
1452580453:@ "
TBK5M9FPcDfZ8/bawWDoYW82J9yAw1Sa8GURKf4fgRrOhYGk/ODOHODNih4ox2F
4gXPdq5UK1kRGbu114dNSK2ITLfhupQ5EMxIF9Xy1uR/z7mIWaViMGKMDLNd9b4
p6r9b3vDBmh1OXV9oknaEELLQLiw6Ka3JIj/qTmKKPD6E=" :@""];
```

相关 API 接口详细说明请参考《AnyChat SDK for iOS 开发手册》文档

4.6 进入房间

除了音视频交互功能需要本流程之外，没有特殊说明，其他功能都不需要本流程。应用层将 roomid 传入，进入指定的房间，只有在同一个房间内的用户才能进行音视频交互，参考代码如下：

```
//进入房间
[AnyChatPlatform EnterRoom:1 :@””];
```

此流程操作是一个异步的操作，会依次触发 OnAnyChatEnterRoom、OnAnyChatOnlineUser 消息回调，参考代码如下：

```
//用户进入房间消息(dwRoomId表示房间号，dwErrorCode=0表示进入房间成功)
- (void) OnAnyChatEnterRoom:(int) dwRoomId : (int) dwErrorCode{
}
// 房间在线用户消息(在线用户进入房间成功后调用一次。dwUserNum当前房间总人数,包含自己)
- (void) OnAnyChatOnlineUser:(int) dwUserNum : (int) dwRoomId{
}
```

五、音视频交互

AnyChat for iOS SDK 为开发者提供了便捷的建立音视频通讯的接口，通过以下几步操作，即可在您的应用中集成音视频交互功能。需要注意的是只有在同一个房间内的用户才能进行音视频通讯。

5.1 设置必要的参数

在初始化之后，设置必要的音视频参数，参考代码如下：

```
//设置本地视频采用 Overlay 模式
[AnyChatPlatform SetSDKOptionInt:BRAC_SO_LOCALVIDEO_OVERLAY :1];
//设置本地视频采集随着设备而旋转而处理
[AnyChatPlatform SetSDKOptionInt:BRAC_SO_LOCALVIDEO_ORIENTATION :self.int
erfaceOrientation];
```

5.2 摄像头硬件初始化

这个操作需要在显示视频的类中执行，步骤如下：

(1) 在显示类的.h 头文件中引用 AVFoundationk 框架，参考代码如下：

```
//获取本地视频显示框架
#import <AVFoundation/AVFoundation.h>
```

(2) 在显示类的.m 文件中实现初始化操作方法，代码如下：

```
//创建视频显示层全局变量
AVCaptureVideoPreviewLayer*localVideoSurface;
//AnyChat SDK 自动调用“摄像头硬件初始化”方法
- (void) OnLocalVideoInit:(id)session
{
//通过 session 控制设备的视频数据输入和输出流向
localVideoSurface       =       [AVCaptureVideoPreviewLayerlayerWithSession:
(AVCaptureSession*)session];
//视频显示层 UI 设置
localVideoSurface.frame = CGRectMake(0, 0, 120, 160);
localVideoSurface.videoGravity = AVLayerVideoGravityResizeAspectFill;
//视频显示层添加到自定义的 theLocalView 界面显示视图中。
[self.theLocalView.layeraddSublayer:self.localVideoSurface];
}
```

5.3 打开本地音视频

打开本地音视频数据需要在进入房间成功之后才有效，即在收到 OnAnyChatEnterRoom 回调后（参考 4.5）打开本地音视频，其他客户端才能请求到你的音视频数据。调用 UserCameraControl 打开视频，调用 UserSpeakControl 打开音频。参考代码如下：

```
//打开本地音频(参数“-1”表示本地用户，也可以用本地的真实 userid)
[AnyChatPlatform UserSpeakControl: -1:YES];
//设置本地视频 UI（“0”为默认适配视频显示位置与尺寸大小）
[AnyChatPlatform SetVideoPos:-1 :self :0 :0 :0 :0];
//打开本地视频(参数“-1”表示本地用户，也可以用本地的真实 userid)
[AnyChatPlatform UserCameraControl:-1 : YES];
```

执行此操作之后，会自动调用 OnLocalVideoInit 方法，在此方法中可以进行摄像头的初始化操作（参考 5.2）。

5.4 关闭本地音视频

打开本地音视频后，可以在音视频交互的过程中选择关闭本地音视频。同时，还可以在关闭之后重新打开本地音视频（参考 5.3）；在音视频交互结束之后需要调用该操作，释放本地摄像头和音频采集设备，参考代码如下：

```
//关闭本地音频
[AnyChatPlatform UserSpeakControl: -1 : NO];
//关闭本地视频
[AnyChatPlatform UserCameraControl: -1 : NO];
```

执行此操作之后，AnyChat SDK 自动调用“摄像头硬件资源释放”方法 OnLocalVideoRelease，参考代码如下：

```
- (void) OnLocalVideoRelease:(id)sender
{//“localVideoSurface”表示视频显示层全局变量（参考 4.2.2）
if(localVideoSurface) {
localVideoSurface = nil;
   }
}
```

5.5 请求远程音视频

在触发 OnAnyChatOnlineUser 或者 OnAnyChatEnterRoom 并判断通话目标对象已经进入当前房间之后（参考 4.2），该操作才有效。调用 UserCameraControl 打开目标对象视频，调用 UserSpeakControl 打开目标对象音频；

SetVideoUser 绑定指定的 UIImageView 用于显示视频。（如要显示多个人的视频，侧要为每一位目标对象用户绑定一个不同指针地址的 UIImageView 即可），参考代码如下：

```
//打开当前房间在线目标对象的音视频，需要传入它的 userid
[AnyChatPlatform UserSpeakControl:userid:YES];
//绑定目标对象视频显示在自定义的 remoteVideoSurface
UIImageView*remoteVideoSurface;
//“0”参数：目标对象视频显示位置与尺寸大小
[AnyChatPlatform SetVideoPos:userid:remoteVideoSurface:0:0:0:0];
//打开目标用户视频
[AnyChatPlatform UserCameraControl:userid :YES];
```

5.6 关闭远程音视频

请求远程音视频后，可以在音视频交互的过程中选择关闭远程音视频。同时，还可以在关闭之后重新请求远程音视频（参考 5.5）；在音视频交互结束之后需要调用该操作，释放远程音视频资源，参考代码如下：

```
//关闭远程音频，userid 为远程目标用户 userid
[AnyChatPlatform UserSpeakControl: userid : NO];
//关闭远程视频
[AnyChatPlatform UserCameraControl: userid: NO];
```

六、业务排队

AnyChat for iOS SDK 为开发者提供了实现业务队列及用户排队、座席为队列内的用户进行服务的接口，通过以下几步操作，即可在您的应用中集成业务排队功能。座席从队列中取出用户后，通过调用视频呼叫的接口实现音视频交互。

AnyChat排队业务解决方案的介绍可以点击 “AnyChat提供业务排队整体解决方案” 链接查看。

6.1 初始化业务对象身份

在进行排队业务之初，需要对登录用户的身份进行初始化，登录的用户是坐席还是客户，调用设置对象属性接口对属性进行赋值。参考代码如下：

```
// 业务对象身份初始化
 [AnyChatPlatform SetSDKOptionInt:BRAC_SO_OBJECT_INITFLAGS : dwAgentFlags];
// 0 普通用户 2 坐席
// 客户端用户对象优先级
 [AnyChatPlatform
ObjectSetIntValue:ANYCHAT_OBJECT_TYPE_CLIENTUSER :mSelfUserId :ANYCHAT_OB
JECT_INFO_PRIORITY :10];
// 对象属性赋值
 [AnyChatPlatform
ObjectSetIntValue:ANYCHAT_OBJECT_TYPE_CLIENTUSER :mSelfUserId :ANYCHAT_OB
JECT_INFO_ATTRIBUTE :-1];
```

6.2 显示/进入/离开营业厅

在业务对象身份初始化之后，需要调用 ObjectControl 接口发送服务区域数据同步请求指令同步已存在的营业厅数据，接口第二个参数为-1 表示同步所有的营业厅。参考代码如下：

```
// 向服务器发送数据同步请求指令
 [AnyChatPlatform                                                ObjectControl:
ANYCHAT_OBJECT_TYPE_AREA  :ANYCHAT_INVALID_OBJECT_ID  :ANYCHAT_OBJECT_CTRL_
SYNCDATA :mSelfUserId :0 :0 :0 :nil];
```

调用同步数据方法之后会接收到 ANYCHAT_OBJECT_EVENT_UPDATE 事件（业务对象更新事件），需要响应此事件去处理营业厅显示的逻辑功能，每个营业厅对象都会触发一次。参考代码如下：

```
- (void) OnAnyChatObjectEventCallBack: (int) dwObjectType : (int) dwObjectId :
(int) dwEventType : (int) dwParam1 : (int) dwParam2 : (int) dwParam3 : (int)
dwParam4 : (NSString*) lpStrParam {
    switch (dwEventType) {
        case ANYCHAT_OBJECT_EVENT_UPDATE:// 1.对象数据更新
            [self AnyChatObjectUpdate:dwObjectType :dwObjectId];
            break;
        default:
            break;
    }
}
```

将营业厅显示在界面之后，可以调用 ObjectControl 接口，执行 ANYCHAT_AREA_CTRL_USERENTER 方法进入某个营业厅。参考代码如下：

```
// 进营业厅
[AnyChatPlatform
ObjectControl:ANYCHAT_OBJECT_TYPE_AREA  :(int)bHall.hallId  :ANYCHAT_AREA_C
TRL_USERENTER :0 :0 :0 :0 :nil];
```

在执行成功后，会触发 ANYCHAT_AREA_EVENT_ENTERRESULT 事件（用户进入营业厅事件）。

如果要退出某个营业厅，也可以调用 ObjectControl 接口，执行 ANYCHAT_AREA_CTRL_USER LEAVE 方法进入某个营业厅。参考代码如下：

```
// 退出营业厅
[AnyChatPlatform
ObjectControl:ANYCHAT_OBJECT_TYPE_AREA :self.businessHallId :ANYCHAT_AREA
_CTRL_USERLEAVE :0 :0 :0 :nil];
```

6.3 显示/进入/离开队列

身份为“客户”的用户在进入营业厅后，在 ANYCHAT_AREA_EVENT_ENTERRESULT 事件（用户进入营业厅事件）中可以实现显示营业厅中的队列功能，会调用到 ObjectGetIdList、ObjectGetIntValue、ObjectGetStringValue 等接口获取队列的信息，如队列 ID 列表、队列名称、队列描述信息、队列中当前人数等。参考代码如下：

```
// 获取营业厅内的队列信息
NSMutableArray *queuesArray= [AnyChatPlatform
ObjectGetIdList:ANYCHAT_OBJECT_TYPE_QUEUE];
// 获取营业厅名称
NSString *areaName = [AnyChatPlatform
ObjectGetStringValue:dwObjectType :dwObjectId :ANYCHAT_OBJECT_INFO_NAME];
```

在已显示的队列中，通过调用 ObjectControl 接口，执行 ANYCHAT_QUEUE_CTRL_USERENTER 方法进入某个队列。参考代码如下：

```
// 进入队列
   [AnyChatPlatform
ObjectControl:ANYCHAT_OBJECT_TYPE_QUEUE :[self.businessListIdArray[indexP
ath.row] intValue] :ANYCHAT_QUEUE_CTRL_USERENTER :0 :0 :0 :nil];
```

在进入队列后就开始排队，等待坐席的服务，在排队期间可以通过 ObjectGetValue 接口获取队列的属性信息，如：获取当前队列人数、获取排在自己前面的用户数、自己在队列中的等待时间等。参考代码如下：

```
// 获取排在你前面有多少人
self.beforeNum                    =                    [AnyChatPlatform
ObjectGetIntValue:ANYCHAT_OBJECT_TYPE_QUEUE  :self.businessId  :ANYCHAT_QUE
UE_INFO_BEFOREUSERNUM];
// 队列名称
NSString            *businessName            =            [AnyChatPlatform
ObjectGetStringValue:ANYCHAT_OBJECT_TYPE_QUEUE  :(int)self.businessId  :ANY
CHAT_OBJECT_INFO_NAME];
// 队列中等待时间
int              waitingTime              =              [AnyChatPlatform
ObjectGetIntValue:ANYCHAT_OBJECT_TYPE_QUEUE  :self.businessId  :ANYCHAT_QUE
```

在等待的过程中，通过调用 ObjectControl 接口，执行 ANYCHAT_QUEUE_CTRL_USERLEAVE 方法实现客户退出队列功能。参考代码如下：

```
// 离开队列
[AnyChatPlatform
ObjectControl:ANYCHAT_OBJECT_TYPE_QUEUE  :self.businessId  :ANYCHAT_QUEUE_C
TRL_USERLEAVE :0 :0 :0 :0 :nil];
```

6.4 坐席服务

身份为“坐席”的用户在进入营业厅后，在 ANYCHAT_AREA_EVENT_ENTERRESULT 事件（用户进入营业厅事件）中可以进入到坐席服务窗口，会调用到 ObjectGetIntValue、ObjectGetStringValue 等接口获取营业厅、队列等信息，如队列数量、队列总用户数、累计服务的用户数等。参考代码如下：

```
// 队列数量
[AnyChatPlatform  ObjectControl:ANYCHAT_OBJECT_TYPE_AREA  :self.areaId  :
ANYCHAT_AREA_INFO_QUEUECOUNT :0 :0 :0 :0 :nil];
// 队列总用户数
[AnyChatPlatform  ObjectControl:ANYCHAT_OBJECT_TYPE_AREA  :self.areaId  :
ANYCHAT_AREA_INFO_QUEUEUSERCOUNT :0 :0 :0 :0 :nil];
// 累计服务的用户数
[AnyChatPlatform  ObjectControl:ANYCHAT_OBJECT_TYPE_AGENT  :self.userId  :
ANYCHAT_AGENT_INFO_SERVICETOTALNUM :0 :0 :0 :0 :nil];
```

坐席端通过调用 ObjectControl 接口，执行 ANYCHAT_AGENT_CTRL_SERVICEREQUEST 方法开始服务，从队列中获取一个用户（根据优先级、优先队列、先进先出等策略由系统自动分配）进行服务。

```
// 开始服务
[AnyChatPlatform
ObjectControl:ANYCHAT_OBJECT_TYPE_AGENT :self.selfUserId :ANYCHAT_AGENT_CTRL_SERVICEREQUEST :0 :0 :0 :nil];
```

成功后会触发 ANYCHAT_AGENT_EVENT_SERVICENOTIFY 事件（坐席服务通知（哪个用户到哪个坐席办理业务）事件），同时被选择的用户会离开队列。在此事件中将进行坐席与客户之间的视频呼叫请求处理，参考代码如下：

```
if ([self.role.text isEqualToString:@"坐席"] && self.selfUserId == dwAgentId)
{
       self.customerId = clientId;
       // 呼叫用户
       [AnyChatPlatform
VideoCallControl:BRAC_VIDEOCALL_EVENT_REQUEST :clientId :0 :0 :nil];
       ServerQueueViewController                *serverQVC                =
[self.navigationController.viewControllers lastObject];
       serverQVC.waitingAlertView = [[UIAlertView alloc] initWithTitle:@"呼叫中，等待顾客确定" message:@"请稍等..." delegate:serverQVC
cancelButtonTitle:@"取消" otherButtonTitles:nil, nil];
       [serverQVC.waitingAlertView show];
   }
```

后续的坐席与客户之间视频呼叫功能见下面“七、视频呼叫”章节描述。

七、视频呼叫

AnyChat for iOS SDK 为开发者提供了视频呼叫的功能，实现两个用户之间（如客户与坐席）的视频呼叫请求（Request）、视频呼叫回复（Reply）、视频呼叫开始（Start）以及视频呼叫结束（Finish）等过程，可以形象理解为打电话的流程：拨号、等待、通话、挂断。

AnyChat视频呼叫详细技术实现可以点击“AnyChat视频呼叫业务逻辑详解”链接查看。

7.1 视频呼叫请求

视频呼叫请求由请求方（如坐席）向被服务的一方（如客户）发起，通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 BRAC_VIDEOCALL_EVENT_REQUEST 事件实现发送视频呼叫请求，参考代码如下：

```
// 呼叫用户
[AnyChatPlatform
VideoCallControl:BRAC_VIDEOCALL_EVENT_REQUEST :clientId :0 :0 :0 :nil];
```

被服务的一方（如客户）接收到 BRAC_VIDEOCALL_EVENT_REQUEST 事件后要实现是否接收请求的业务逻辑处理，参考代码如下：

```
- (void) OnAnyChatVideoCallEventCallBack:(int) dwEventType : (int) dwUserId :
(int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr{
    self.remoteUserId = dwUserId;
    self.loginVC.remoteUserId = dwUserId;
    switch (dwEventType) {

        case BRAC_VIDEOCALL_EVENT_REQUEST://呼叫请求 1
        {
            self.requestAlertView = [[UIAlertView alloc] initWithTitle:@"客服
人 员 请 求 与 您 通 话 "  message:nil  delegate:self  cancelButtonTitle:nil
otherButtonTitles:@"确定",@"取消" ,nil];
            self.requestAlertView.delegate = self;
            [self.requestAlertView show];

            break;
        }

    }
```

7.2 视频呼叫回复

视频呼叫回复由被服务的一方（如客户）发起，通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 BRAC_VIDEOCALL_EVENT_REPLY 事件实现发送视频呼叫回复；另外被服务的一方（如客户）也可以选择拒绝请求方（如坐席）的视频呼叫请求，参考代码如下：

```
//发送视频呼叫回复指令，dwErrcode=0
[AnyChatPlatform
VideoCallControl:BRAC_VIDEOCALL_EVENT_REPLY :self.remoteUserId :0 :0 :0 :
nil];
```

传入 AC_ERROR_VIDEOCALL_REJECT 参数实现拒绝呼叫请求，参考代码如下：

```
//发送视频呼叫回复指令，拒绝请求，dwErrcode=100104
[AnyChatPlatform
VideoCallControl:BRAC_VIDEOCALL_EVENT_REPLY  :self.remoteUserId  :GV_ERR_VI
DEOCALL_REJECT :0 :0 :nil];
```

请求方（如坐席）在收到 BRAC_VIDEOCALL_EVENT_REPLY 事件后要根据被服务的一方（如客户）的回复情况实现不同的业务逻辑处理，参考代码如下：

```
- (void) OnAnyChatVideoCallEventCallBack:(int) dwEventType : (int) dwUserId :
(int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr{
    switch (dwEventType) {
        case BRAC_VIDEOCALL_EVENT_REPLY:// 呼叫请求回复 2{
            switch (dwErrorCode){
                case GV_ERR_VIDEOCALL_CANCEL: // 源用户主动放弃会话
                    break;
                case GV_ERR_VIDEOCALL_REJECT: // 目标用户拒绝会话
                    break;
                case GV_ERR_VIDEOCALL_OFFLINE:// 目标用户不在线
                    break;
                case GV_ERR_VIDEOCALL_BUSY:// 目标用户忙
                    break;
                case GV_ERR_VIDEOCALL_TIMEOUT:// 会话请求超时
                    break;
                case GV_ERR_VIDEOCALL_DISCONNECT:// 网络断线
                    break;
                case GV_ERR_VIDEOCALL_NOTINCALL:// 用户不在呼叫状态
                    break;
            }
            break;
        }
    }

}
```

7.3 视频呼叫开始

被服务的一方（如客户）通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 BRAC_VIDEOCALL_EVENT_REQUEST 事件同意了请求方（如坐席）的视频呼叫请求，则系统会触发 BRAC_VIDEOCALL_EVENT_START 事件，系统会自动的分配一个房间号，双方都需要进入该房间号，程序需要响应此事件实现开始视频呼叫的业务逻辑功能。参考代码如下：

```
- (void) OnAnyChatVideoCallEventCallBack:(int) dwEventType : (int) dwUserId :
(int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr{
    switch (dwEventType) {
        case BRAC_VIDEOCALL_EVENT_START://用户同意，视频开始 3
        {
            if (self.waitingAlertView != nil) {
                [self.waitingAlertView
dismissWithClickedButtonIndex:self.waitingAlertView.cancelButtonIndex
animated:YES];
            }
            //进入房间
            [AnyChatPlatform EnterRoom:dwParam :nil];
            break;
        }
    }
}
```

在进入房间后，客户端 SDK 会触发 ENTERROOM 回调方法。参考代码如下：

```
//进入房间回调
- (void) OnAnyChatEnterRoom:(int) dwRoomId : (int) dwErrorCode {
    NSLog(@"用户进入房间");
    if (dwErrorCode == 0) {
        VideoViewController *videoVC = [[VideoViewController alloc] init];
        if ([self.role.text isEqualToString:@"普通用户"]) {
            videoVC.remoteUserId = self.remoteUserId;
        }else if([self.role.text isEqualToString:@"坐席"]) {
            videoVC.remoteUserId = self.customerId;
        }
        [self.navigationController pushViewController:videoVC animated:YES];
    }
```

7.4 视频呼叫结束

在整个视频通话服务完成后，任一方都可以通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 BRAC_VIDEOCALL_EVENT_FINISH 事件结束当前的视频呼叫。参考代码如下：

```
- (void) OnAnyChatVideoCallEventCallBack:(int) dwEventType : (int) dwUserId :
(int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr{
    switch (dwEventType) {
        case BRAC_VIDEOCALL_EVENT_FINISH://视频结束 4
        {
            // 关闭设备
            [AnyChatPlatform UserSpeakControl: -1 : NO];
            [AnyChatPlatform UserCameraControl: -1 : NO];
            [AnyChatPlatform UserSpeakControl: dwUserId : NO];
            [AnyChatPlatform UserCameraControl: dwUserId : NO];
            // 离开房间
            [AnyChatPlatform LeaveRoom:-1];
            [MBProgressHUD showSuccess:@"视频通话结束"];
            [self.navigationController popViewControllerAnimated:YES];

            break;
        }
    }
}}
```

八、资源释放

(1) 离开房间

释放当前房间内的音视频资源。参考代码如下：

```
//“-1”表示离开当前房间
 [AnyChatPlatform LeaveRoom:-1];
```

在音视频交互结束后，可调用该操作。离开当前房间之后，可再次选择进入指定房间。

(2) 退出

断开与 AnyChat 通讯服务器连接。参考代码如下：

```
 //断开与服务器的连接
[AnyChatPlatform Logout];
```

在需要断开跟 AnyChat 服务器通讯连接的时候，可调用该操作。退出之后，可以再次调用连接、登录服务器。

(3) 释放 SDK

释放整个 SDK 资源。参考代码如下：

```
//释放资源
 [AnyChatPlatform Release];
```

建议在退出整个应用程序的时候调用该操作。释放 SDK 之后，需要重新初始化 SDK 之后才能进行连接、登录、进入房间等操作。

九、附录

本附录中包括了开发流程指南中所用到的示例程序的运行截图。

9.1 HelloAnyChat界面

AnyChat for iOS SDK 包里提供的 HelloAnyChat 程序（源码在“src\helloAnyChat”目录下）运行效果如下图所示：

登录界面　　音视频通讯界面　　视频参数设置界面

图 9-1 HelloAnyChat App 界面

9.2 AnyChatQueue界面

AnyChat for iOS SDK 包里提供 AnyChatQueue 程序（源码在“src \AnyChatQueue”目录下）运行效果如下图所示：

图 9-2 登录界

图 9-3 队列列表

图 9-4 普通用户排队等待

图 9-5 视频通话中