

# AnyChat for iOS SDK

## 开发手册

(版本: V1.9)



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>

2014 年 01 月

# 目 录

<b>1. 系统概述 .....</b>	<b>5</b>
1.1 系统介绍 .....	5
1.2 系统特性 .....	5
1.2.1 视频技术 .....	6
1.2.2 音频技术 .....	6
1.2.3 P2P技术 .....	6
1.3 关于佰锐科技 .....	7
1.4 技术支持 .....	7
1.5 版权申明 .....	8
<b>2. 编程指南 .....</b>	<b>9</b>
2.1. 客户端SDK概述 .....	9
2.2. 函数调用顺序 .....	10
2.3. 编程要点 .....	11
2.3.1. SDK的Objective-C封装 .....	11
2.3.2. 异步事件与Delegate .....	11
2.3.3. UIImageView视频显示 .....	12
2.3.4. 字符编码 .....	12
2.3.5. SDK生命周期 .....	12
2.4. 服务器SDK概述 .....	14
<b>3. 协议(DELEGATE)说明 .....</b>	<b>15</b>
3.1. 基本流程事件协议 .....	15
3.1.1. 接口定义 .....	15
3.1.2. 网络连接事件 .....	16
3.1.3. 用户登录事件 .....	16
3.1.4. 进入房间事件 .....	16
3.1.5. 房间在线用户事件 .....	16
3.1.6. 新用户进入事件 .....	17
3.1.7. 房间用户离开事件 .....	17
3.1.8. 网络连接关闭事件 .....	17
3.2. 用户好友事件接口 .....	17
3.2.1. 接口定义 .....	18
3.2.2. 用户信息更新事件 .....	18
3.2.3. 好友在线状态变化事件 .....	18
3.3. 状态变化事件协议 .....	19
3.3.1. 接口定义 .....	19
3.3.2. 音频设备状态改变事件 .....	19
3.3.3. 视频设备状态改变事件 .....	19
3.3.4. 用户活动状态改变事件 .....	20
3.3.5. 用户P2P连接改变事件 .....	20

3.3.6.	用户视频大小改变事件 .....	20
3.4.	数据传输事件接口 .....	21
3.4.1.	接口定义 .....	21
3.4.2.	收到文件传输数据事件 .....	21
3.4.3.	收到透明通道数据事件 .....	22
3.4.4.	收到扩展透明通道数据事件 .....	22
3.4.5.	收到SDK Filter数据事件 .....	23
3.5.	文字消息事件接口 .....	23
3.5.1.	接口定义 .....	23
3.5.2.	收到文字聊天消息事件 .....	23
3.6.	私聊控制事件接口 .....	24
3.6.1.	接口定义 .....	24
3.6.2.	用户私聊请求事件 .....	24
3.6.3.	用户私聊回复事件 .....	25
3.6.4.	用户私聊退出事件 .....	25
3.6.5.	用户私聊状态改变事件 .....	25
4.	函数说明 .....	26
4.1.	初始化与资源释放 .....	26
4.1.1.	初始化SDK .....	26
4.1.2.	释放SDK资源 .....	26
4.2.	登录流程 .....	27
4.2.1.	设置服务器认证密码 .....	27
4.2.2.	连接服务器 .....	27
4.2.3.	登录系统 .....	28
4.2.4.	进入房间（根据房间编号） .....	28
4.2.5.	进入房间（根据房间名称） .....	29
4.2.6.	离开房间 .....	30
4.2.7.	注销系统 .....	30
4.3.	音视频操作 .....	31
4.3.1.	用户视频控制 .....	31
4.3.2.	用户语音控制 .....	31
4.3.3.	设置视频显示位置 .....	32
4.4.	查询状态 .....	33
4.4.1.	查询摄像头的状态 .....	33
4.4.2.	查询用户音频设备采集状态 .....	33
4.4.3.	查询用户昵称 .....	34
4.4.4.	查询用户IP地址 .....	34
4.4.5.	查询用户视频宽度 .....	34
4.4.6.	查询用户视频高度 .....	35
4.4.7.	查询用户状态（字符串） .....	35
4.4.8.	查询用户状态（数字） .....	36
4.4.9.	查询房间名称 .....	36
4.5.	普通功能 .....	37
4.5.1.	获取SDK版本信息 .....	37

4.5.2.	获取当前房间在线用户列表 .....	37
4.5.3.	传送文本消息 .....	37
4.5.4.	透明通道传送缓冲区 .....	38
4.5.5.	透明通道传送缓冲区扩展 .....	38
4.5.6.	传送文件 .....	39
4.5.7.	查询传输任务相关信息 .....	40
4.5.8.	激活（关闭）SDK调用日志 .....	41
4.5.9.	发送SDK Filter通信数据 .....	41
4.5.10.	音视频录制 .....	42
4.6.	<b>私聊函数</b> .....	42
4.6.1.	更改当前的聊天模式 .....	42
4.6.2.	获取指定用户聊天模式 .....	43
4.6.3.	请求与对方私聊 .....	43
4.6.4.	回复对方的私聊请求 .....	44
4.6.5.	回复对方的私聊请求（扩展） .....	44
4.6.6.	退出与某用户的私聊 .....	45
4.7.	<b>好友列表</b> .....	45
4.7.1.	获取好友列表 .....	46
4.7.2.	获取好友在线状态 .....	46
4.7.3.	获取好友分组列表 .....	46
4.7.4.	获取分组名称 .....	47
4.7.5.	获取分组所对应的用户列表 .....	47
4.7.6.	获取好友用户信息 .....	48
4.7.7.	用户信息控制 .....	48
4.8.	<b>系统设置</b> .....	49
4.8.1.	枚举本地视频采集设备 .....	49
4.8.2.	选择指定的视频采集设备 .....	49
4.8.3.	获取当前视频采集设备 .....	49
4.8.4.	枚举本地音频采集设备 .....	50
4.8.5.	选择指定的音频采集设备 .....	50
4.8.6.	获取当前音频采集设备 .....	50
4.8.7.	枚举本地音频播放设备 .....	51
4.8.8.	选择指定的音频播放设备 .....	51
4.8.9.	获取当前音频播放设备 .....	52
4.8.10.	获取音频设备的当前音量 .....	52
4.8.11.	设置指定音频设备的音量 .....	52
4.8.12.	SDK内核参数设置 .....	53
4.8.13.	SDK内核参数状态查询 .....	54
5.	<b>版本变更记录</b> .....	55
6.	<b>附录一：错误代码参考</b> .....	56

# 1. 系统概述

非常感谢您使用佰锐科技的产品，我们将为您提供最好的服务。

本手册可能包含技术上不准确的地方或排版错误。本手册的内容将做定期的更新，恕不另行通知；更新的内容将会在本手册的新版本中加入。我们随时会改进或更新本手册中描述的产品或程序。

## 1.1 系统介绍

AnyChat for iOS SDK 是一套即时通讯开发平台（SDK），包含了音视频处理模块与 P2P 网络模块两大部分，是 AnyChat Platform Core SDK 的重要组成部分，专为 iOS 平台设计，适用于 iPhone、iPad、iTouch 等 Apple 公司移动终端设备，并针对 ARM 系列 CPU 进行了汇编优化，可以做为 iPhone 平台上的即时通讯内核引擎，也可以做为视频会议、网络教育、即拍即传系统等互动平台的核心库。整个平台由广州佰锐网络科技有限公司独立研发，具有自主知识产权。

AnyChat SDK 分为客户端 SDK 和服务器 SDK 两大部分，其中客户端 SDK 用于实现语音、视频的交互以及其它客户端相关的功能，而服务器 SDK 主要实现业务层逻辑控制，以及与第三方平台的互联等。AnyChat for iOS SDK 提供 Objective-C 语言接口和 C++语言接口。

## 1.2 系统特性

“AnyChat for iOS SDK”采用增强的 H.264 视频编码算法和 AMR 语音编码算法，具有高画质、语音清晰、流畅的特点，支持 P2P 技术进行网络传输，服务器采用完成端口模型的重叠 IO，具有极高的并发处理能力。

服务器支持“SDK Filter Plus”和“AnyChat Server SDK”两种可扩展编程接口，可方便实现与其它系统进行集成，增强 AnyChat 的可扩展性。上层应用也可利用服务器 SDK 来实现更复杂的业务逻辑处理。

## 1.2.1 视频技术

视频制式：PAL-B

分辨率：192×144 —480×360（可调节）

帧 率：1~25（可调节）

视频编码器：H.264

视频流码率：10kbps ~ 1000kbps（VBR）

## 1.2.2 音频技术

采样率：16000 Hz、32000Hz、44100Hz、48000Hz（可设置）

量化值：16 bit

声 道：Mono、Stereo

音频编码器：AMR\_WB、AAC

音频流码率：6kbps ~ 128kbps

音效处理：回音抑制（AES）、噪音抑制（NS）、自动增益控制（AGC）、静音检测（VAD）

## 1.2.3 P2P技术

传输方式：UDP、TCP

支持的 NAT 类型：

Cone NAT <——> Cone NAT

Cone NAT <——> Symmetric NAT

支持 UPNP 协议（wifi 网络中有效）

## 1.3 关于佰锐科技

广州佰锐网络科技有限公司（GuangZhou BaiRui Network Technology Co., Ltd.）是一家从事网络语音视频技术与系统开发的高新技术企业（广州市黄花岗科技园入园企业），企业注册资金 200 万元人民币，拥有独立知识产权。主要从事网络流媒体及相关平台的研发工作。专注于语音视频技术研发与创新。旗下有“中华视频网（<http://www.chinavideo.org>）”、“ffmpeg 工程组（<http://www.ffmpeg.com.cn>）”和“AnyChat 互动平台（<http://www.anychat.com.cn>）”和“Business Eyes（商务眼）（<http://www.shangwuyan.com>）”等多个网站。

佰锐科技始创于 2005 年 11 月，并于 2006 年启动国内第一家专注于视频技术的技术性网站：中华视频网，2006 年 3 月正式启动 AnyChat 项目。广州佰锐网络科技有限公司于 2006 年申请了两项发明专利（2006101239829、2006101241176）和两项软件著作权（软著登字第 066348 号、软著登字第 092142 号）。

## 1.4 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、知识中心：<http://www.anychat.cn/faq/>
- 2、在线论坛：<http://bbs.anychat.cn/>
- 3、官方网站：<http://www.anychat.cn>
- 4、电子邮件：[service@bairuitech.com](mailto:service@bairuitech.com)
- 5、24 小时客服电话：+86 （020） 85276986、38109065、87691886
- 6、即时通讯：QQ: 75513809

## 1.5 版权申明

“AnyChat for iOS SDK”是由广州佰锐网络科技有限公司开发，拥有自主知识产权（软著登字第 066348 号）的系统平台，

广州佰锐网络科技有限公司拥有与本产品所用技术相关的知识产权。这些知识产权包括但不限于一项或多项发明专利或者正在进行申请的专利（2006101239829、2006101241176）。

本产品发行所依照的许可协议限制其使用、复制分发和反编译。未经广州佰锐网络科技有限公司事先书面授权，不得以任何形式或借助任何手段复制本产品的任何部分。

随本 SDK 一同发布的 Demo 演示程序源代码版权归广州佰锐网络科技有限公司所有。

AnyChat 是广州佰锐网络科技有限公司的商标。

*第三方组件：AnyChat 使用了开源的第三方组件，包括：FFmpeg (LGPL 2.1)、libvpx (BSD)、libspeex (BSD)、WebRTC (BSD)。其中FFmpeg的版权所有者信息、源代码以及其它信息均可在其官方网站：<http://www.ffmpeg.org>找到。*



## 2. 编程指南

### 2.1. 客户端SDK概述

“AnyChat for iOS”属于客户端组件（简称“客户端”），对上层应用提供 Objective-C 语言或是 C++语言的调用接口，内核是由一系列的静态库（相当于 Win32 平台的 LIB）组合而成。

系统采用模块化设计，每个模块都独立完成特定的任务，模块之间采用弱关联设计，今后系统某部分功能的升级，如音频、视频编码算法的改进，只需要替换相关的模块即可，不影响系统的接口。

AnyChat for iOS 与服务器有一系列的交互过程，包括：连接服务器、登录系统、进入房间，交互过程的结果（如连接服务器是否成功）SDK 内部将会采用 Objective-C 接口技术（Windows 平台是采用消息机制）通知上层应用。只有进入同一房间的两个用户之间才能进行语音、视频、文字的交互，当某用户打开了本地设备后，其它用户请求该用户的数据时，便能收到该用户的数据。

AnyChat for iOS 客户端在房间中，收到其它用户的流媒体数据后，上层应用只需要提供一个 UIImageView 控件的句柄，内核便可自动显示视频到该 UIImageView 控件上，并自动播放声音。

AnyChat for iOS 的工作流程与 Windows 平台的 SDK 一致，熟悉 Windows 平台的 SDK 工作机制将更有助于了解 AnyChat for iOS 平台的工作机制。

## 2.2. 函数调用顺序

调用 顺序	函数	功能	备注
A	InitSDK	初始化系统	初始化
	SetSDKOptionInt	设置 SDK 整形值相关参数	
	SetSDKOptionString	设置 SDK 字符串相关参数	
B	Connect	连接服务器	进入系统
	Login	用户登录系统	
	EnterRoom	进入房间	
C	GetOnlineUser	获取当前房间在线用户数	其它功能
	GetCameraState	查询用户摄像头的状态	
	GetSpeakState	查询用户发言状态	
	ShowLVProperty	显示本地视频画面调节对话框	
	EnumVideoCapture	枚举本地视频采集设备	
	SelectVideoCapture	选择指定的视频采集设备	
	GetCurVideoCapture	获取当前使用的视频采集设备	
	EnumAudioCapture	枚举本地音频采集设备	
	SelectAudioCapture	选择指定的音频采集设备	
	GetCurAudioCapture	获取当前使用的音频采集设备	
	AudioGetVolume	获取指定音频设备的当前音量	
	AudioSetVolume	设置指定音频设备的音量	
	SetVideoPos	设置视频显示位置	
	UserCameraControl	操作用户视频	
	UserSpeakControl	操作用户语音	
	SendTextMessage	发送文本消息	
	ChangeChatMode	更改当前的聊天模式	
	GetUserChatMode	获取指定用户当前的聊天模式	
	PrivateChatRequest	请求与对方私聊，发起私聊请求	
	PrivateChatEcho	回复对方的私聊请求	
	PrivateChatExit	退出与某用户的私聊	
	.....	.....	
D	LeaveRoom	离开房间	退出系统
	Logout	用户注销	
	Release	释放资源	

## 2.3. 编程要点

### 2.3.1. SDK的Objective-C封装

AnyChat for iOS 的 Objective-C 接口封装在 “AnyChatPlatform.h” 文件中，接口名称为 “AnyChatPlatform”，所有 API 接口均为静态成员函数，故可以在代码中随时调用 “AnyChatPlatform” 接口中的任何方法。

### 2.3.2. 异步事件与Delegate

AnyChat 的相关操作是一个异步的过程，如调用 “Connect” 方法连接服务器之后就立即返回，此时并不表示连接服务器成功，只有当收到连接服务器的消息（Windows 平台）或是回调时，才能知道连接服务器是否成功，这个异步的过程在 iPhone 平台上被封装成了一系列的协议（Delegate），上层应用根据需要来实现部分协议。

AnyChat for iOS 提供的协议（Delegate）只在 Objective-C 接口中使用，如果是 C++的接口，则可以通过回调来处理异步事件，由于 AnyChat 内核是多线程的，如果使用 C++的回调，然后在回调中需要操作界面元素（主线程），则需要额外的处理，而 Objective-C 接口的协议则不存在该问题，可以在异步事件中直接操作主界面元素，所以建议上层应用使用 Objective-C 接口会相对简洁一些。

要驱动 AnyChat for iOS 的异步事件，需要进行如下两步操作：

1、注册异步通知句柄，示例代码如下：

```
// register our handler
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(AnyChatNotifyHandler:) name:@"ANYCHATNOTIFY"
object:nil];
```

2、处理异步通知，示例代码如下：

```
-(void)AnyChatNotifyHandler:(NSNotification*)notify
{
    NSDictionary* dict = notify.userInfo;
    [anychat OnRecvAnyChatNotify:dict];
}
```

### 2.3.3. UIImageView视频显示

AnyChat for iOS 采用 UIImageView 进行其它用户视频显示，上层应用只需要在界面中创建一个 UIImageView 控件，然后将控件句柄通过 SetVideoPos 方法传递给 SDK，则当有视频数据到达时，内核将会自动将视频显示到该 UIImageView 控件上，不需要上层应用来处理视频的显示。

AnyChat for iOS 采用 AVCaptureVideoPreviewLayer 进行本地视频显示，在初始化本地视频设备之前，通过调用 SetVideoPos 方法将需要显示视频的主窗体 View 对象传递给 SDK，当本地视频初始化成功之后，SDK 内核会调用该 View 的方法 “OnLocalVideoInit”，在该方法中，便可创建和初始化 AVCaptureVideoPreviewLayer 对象，实现本地视频的显示，当本地视频会话关闭，或是对象销毁时，SDK 内核会调用该 View 的方法 “OnLocalVideoRelease”，在该方法中，可释放前面创建的 AVCaptureVideoPreviewLayer 对象。详细信息可参考随 AnyChat for iOS 一同发布的 Demo 程序源代码。

### 2.3.4. 字符编码

iPhone 平台采用 Unicode 编码，AnyChat for iOS 内核在处理 iPhone 与 Windows 平台的通信时，Objective-C 接口会自动将字符串（如文字聊天数据）转换为上层平台所对应的编码，不需要应用层进行转换，但是当上层应用使用透明通道在客户端与服务器，或是客户端之间传输数据，而需要显示时，就需要上层应用手动来处理编码转换的任务，否则显示将会出现乱码。另外 C++ 标准接口还是采用多字节编码，上层应用如需使用 C++ 接口，则需要注意处理字符编码的转换。

### 2.3.5. SDK生命周期

AnyChat for iOS 的生命周期与上层应用程序一致。在系统启动时进行初始化操作，在系统关闭时进行资源释放的操作。

其中需要注意的是 iPhone 应用程序有多种不同的状态，并在这些状态之间进行切换，概括起来主要是两种状态：前台（活动）状态与后台（非活动）状态，

当应用程序进入后台状态之后，需要暂停音、视频的相关操作，如视频流的显示等，可以节省用户的网络流量。

AnyChat 专门为 iPhone 提供了一个应用程序活动状态的内核参数：BRAC\_SO\_CORESDK\_ACTIVESTATE，通过该参数可以设置 AnyChat 内核的活动状态，当设置为后台状态时，AnyChat 内核会暂停请求其它用户的音视频数据，同时也会停止向其它用户传输音视频数据；当设置为前台状态时，AnyChat 内核会恢复之前保存的状态，向其它用户请求音视频数据，同时也可以将本地的音视频数据向其它用户传输。

iPhone 应用程序的状态切换在 AppDelegate 中，典型的代码如下：

```
- (void)applicationWillResignActive:(UIApplication *)application
{
    [AnyChatPlatform SetSDKOptionInt:BRAC_SO_CORESDK_ACTIVESTATE :0];
}
- (void)applicationDidEnterBackground:(UIApplication *)application
{
    [AnyChatPlatform SetSDKOptionInt:BRAC_SO_CORESDK_ACTIVESTATE :0];
}
- (void)applicationWillEnterForeground:(UIApplication *)application
{
    [AnyChatPlatform SetSDKOptionInt:BRAC_SO_CORESDK_ACTIVESTATE :1];
}
- (void)applicationDidBecomeActive:(UIApplication *)application
{
    [AnyChatPlatform SetSDKOptionInt:BRAC_SO_CORESDK_ACTIVESTATE :1];
}
```

## 2.4. 服务器SDK概述

“AnyChat Server SDK”与“SDK Filter Plus”均是服务器扩展编程接口，均为动态连接库（DLL）形式，两者的主要区别是：（1）、“SDK Filter Plus”的 DLL 被 AnyChat 核心服务器程序（AnyChatCoreServer.exe）调用，与 AnyChat 核心服务器程序属同一个进程；（2）、“AnyChat Server SDK”被业务层服务器程序（需要用户编写）调用，与 AnyChat 核心服务器程序属不同的进程，与 AnyChat 核心服务器采用 IPC 的方式进行通信。

“AnyChat Server SDK”与“SDK Filter Plus”两者可以实现相同的功能，通常来说，“SDK Filter Plus”适合业务逻辑较简单的应用，而“AnyChat Server SDK”则适合业务逻辑较复杂的应用，实现独立的业务层服务器，有对应的界面显示。

有关“SDK Filter Plus”的详细介绍可参考相关的开发文档（《AnyChat SDK Filter Plus 开发指南》）及 SDK 包中所附带的相关源代码。

有关“AnyChat Server SDK”的详细介绍可参考相关的开发文档（《AnyChat Server SDK 开发指南》）及 SDK 包中所附带的相关源代码。

有关AnyChat平台用户身份验证与第三方平台集成的问题可参考技术论坛相关介绍：<http://bbs.anychat.cn/forum.php?mod=viewthread&tid=12&extra=page%3D1>

服务器与客户端之间可以传输缓冲区、文件等数据，详情可参考在线文档：<http://www.anychat.cn/faq/index.php?action=artikel&cat=2&id=206&artlang=zh>

## 3. 协议(Delegate)说明

AnyChat for iOS SDK 通过接口（类似于 C++ 的回调函数）实现与上层应用的状态更新和数据交互。SDK 的很多调用都是异步的，如登录操作，调用函数完成之后，需要等待对应接口的函数来触发是否登录成功，总体原则是：需要异步操作的地方，都采用接口来实现。

根据不同的类型，接口也分为几大类，在实际的开发过程中，可根据具体情况实现这些接口。

### 3.1. 基本流程事件协议

AnyChat 与服务器交互是一个异步的过程，上层应用调用 API 的方法后将立即返回，上层应用需要根据异步事件（如 Win32 的消息通知、iOS 的 Delegate 协议调用）等来确定相关的操作是否成功。

#### 3.1.1. 接口定义

```
@protocol AnyChatNotifyMessageDelegate <NSObject>
// 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;
@end
```

### 3.1.2. 网络连接事件

- (void) OnAnyChatConnect:(BOOL) bSuccess

**参数：** bSuccess      表示是否连接成功，BOOLEAN 类型；

**说明：**当客户端连接服务器时被触发，等同于 WIN32 平台的 WM\_GV\_CONNECT 消息。

### 3.1.3. 用户登录事件

- (void) OnAnyChatLogin:(int) dwUserId : (int) dwErrorCode

**参数：**

dwUserId      表示自己的用户 ID 号,当 dwErrorCode 为 0 时有效

dwErrorCode    出错代码，可判断登录是否成功

**说明：**当客户端登录服务器时被触发，等同于 WIN32 平台的 WM\_GV\_LOGINSYSTEM 消息。

### 3.1.4. 进入房间事件

- (void) OnAnyChatEnterRoom:(int) dwRoomId : (int) dwErrorCode

**参数：**

dwRoomId      表示进入的房间 ID 号

dwErrorCode    出错代码，可判断进入房间是否成功

**说明：**当客户端请求进入房间时被触发，等同于 WIN32 平台的 WM\_GV\_ENTERROOM 消息。

### 3.1.5. 房间在线用户事件

- (void) OnAnyChatOnlineUser:(int) dwUserNum : (int) dwRoomId

**参数：**

dwUserNum      表示当前房间的在线用户数（包含自己）

dwRoomId      房间编号



**说明：**房间在线用户消息，进入房间后触发一次，等同于 WIN32 平台的 WM\_GV\_ONLINEUSER 消息。收到该消息后，便可对房间中的用户进行音视频的相关操作，如请求音频、请求视频等。

### 3.1.6. 新用户进入事件

- (void) OnAnyChatUserEnterRoom:(int) dwUserId

**参数：**

dwUserId          表示进入当前房间的用户 ID 号

**说明：**当成功进入房间之后，有新的用户进入房间会触发该接口，等同于 WIN32 平台的 WM\_GV\_USERATROOM 消息。

### 3.1.7. 房间用户离开事件

- (void) OnAnyChatUserLeaveRoom:(int) dwUserId

**参数：**

dwUserId          表示从当前房间离开的用户 ID 号

**说明：**当成功进入房间之后，有其它的用户离开房间会触发该接口，等同于 WIN32 平台的 WM\_GV\_USERATROOM 消息。

### 3.1.8. 网络连接关闭事件

- (void) OnAnyChatLinkClose:(int) dwErrorCode

**参数：**

dwErrorCode          表示连接被关闭的原因

**说明：**当连接服务器成功之后，网络连接关闭时触发该接口，等同于 WIN32 平台的 WM\_GV\_LINKCLOSE 消息。

## 3.2. 用户好友事件接口

用户好友事件回调函数包含了好友信息相关的所有事件，包括信息更新事

件、好友上下线事件等。

### 3.2.1. 接口定义

```
@protocol AnyChatUserInfoDelegate <NSObject>
// 用户信息更新通知, wParam (INT) 表示用户 ID 号, lParam (INT) 表示更新类别
- (void) OnAnyChatUserInfoUpdate:(int) dwUserId: (int) dwType;
// 好友在线状态变化, wParam (INT) 表示好友用户 ID 号, lParam (INT) 表示用户的当前活动状态: 0 离线, 1 上线
- (void) OnAnyChatFriendStatus:(int) dwUserId: (int) dwStatus;
@end
```

### 3.2.2. 用户信息更新事件

- (void) OnAnyChatUserInfoUpdate:(int) dwUserId : ( int) dwType;

参数:

dwUserId 用户 ID 号;  
dwType 表示用户信息更新类别;

说明:

当服务器更新用户信息之后, 将会触发该消息, 通知客户端某个用户的信息已被更新, 客户端可以通过 API 接口获取更新后的信息显示在界面上; 等同于 WIN32 平台的 WM\_GV\_USERINFOUPDATE 消息。

### 3.2.3. 好友在线状态变化事件

- (void) OnAnyChatFriendStatus: (int) dwUserId : ( int) dwStatus;

参数:

dwUserId 用户 ID 号;  
dwStatus 表示用户的当前活动状态: 0 离线, 1 上线;

说明:

由服务器通知客户端, 当好友上线, 或是离线时将会收到通知。在客户端登录成功之后, 客户端针对每一个在线好友均会触发一次用户上线消息; 等同于 WIN32 平台的 WM\_GV\_FRIENDSTATUS 消息。

## 3.3. 状态变化事件协议

状态变化事件接口包含了 SDK 的用户视频设备状态、音频设备状态、P2P 连接状态变化等异步事件的通知。

### 3.3.1. 接口定义

```
@protocol AnyChatStateChangeDelegate <NSObject>
// 用户得到/释放 mic 消息
- (void) OnAnyChatMicStateChg:(int) dwUserId: (BOOL) bGetMic;
// 用户摄像头状态改变消息
- (void) OnAnyChatCameraStateChg:(int) dwUserId: (int) dwState;
// 用户活动状态发生变化消息
- (void) OnAnyChatActiveStateChg:(int) dwUserId: (int) dwState;
// P2P 连接状态变化消息
- (void) OnAnyChatP2PConnectState:(int) dwUserId: (int) dwState;
// 用户视频分辨率改变消息
- (void) OnAnyChatVideoSizeChg:(int) dwUserId: (int) dwWidth: (int)
dwHeight;
@end
```

### 3.3.2. 音频设备状态改变事件

- (void) OnAnyChatMicStateChg:(int) dwUserId: (BOOL) bOpenMic

参数:

dwUserId          表示状态变化的用户 ID

bOpenMic          表示该用户是否已打开音频采集设备

说明: 当进入房间成功之后, 当用户使用 API: UserSpeakControl 操作本地音频设备时将会触发该接口, 等同于 WIN32 平台的 WM\_GV\_MICSTATECHANGE 消息。

### 3.3.3. 视频设备状态改变事件

- (void) OnAnyChatCameraStateChg:(int) dwUserId: (int) dwState

参数:

dwUserId	表示状态变化的用户 ID
dwState	表示该用户当前的视频设备状态：
0	没有摄像头设备
1	有摄像头设备，但没有打开
2	已打开摄像头设备

**说明：**当进入房间成功之后，当用户使用 API: UserCameraControl 操作本地视频设备时将会触发该接口，等同于 WIN32 平台的 WM\_GV\_CAMERASTATE 消息。

### 3.3.4. 用户活动状态改变事件

- (void) OnAnyChatActiveStateChg:(int) dwUserId: (int) dwState

**参数：**

dwUserId	表示状态变化的用户 ID
dwState	表示该用户当前的活动状态：

**说明：**当进入房间成功之后，当用户改变活动状态时将会触发该接口，等同于 WIN32 平台的 WM\_GV\_ACTIVESTATECHG 消息。

### 3.3.5. 用户P2P连接改变事件

- (void) OnAnyChatP2PConnectState:(int) dwUserId: (int) dwState

**参数：**

dwUserId	表示其它用户 ID 号
dwState	表示本地用户与其它用户的当前 P2P 网络连接状态：
0	没有任何连接
1	P2P 连接成功，TCP 连接
2	P2P 连接成功，UDP 连接
3	P2P 连接成功，TCP 与 UDP 连接

**说明：**当进入房间成功之后，与其它用户建立 P2P 连接，或是 P2P 连接被断开时触发该接口，等同于 WIN32 平台的 WM\_GV\_P2PCONNECTSTATE 消息。

### 3.3.6. 用户视频大小改变事件

- (void) OnAnyChatVideoSizeChg:(int) dwUserId: (int) dwWidth: (int) dwHeight

参数:

dwUserId	表示状态变化的用户 ID
dwWidth	表示该用户当前的视频宽度
dwHeight	表示该用户当前的视频高度

**说明:** 当进入房间成功之后, 成功打开本地视频设备, 或是修改视频设备采集分辨率之后将触发该接口, 等同于 WIN32 平台的 WM\_GV\_VIDEOSIZECHG 消息。

## 3.4. 数据传输事件接口

### 3.4.1. 接口定义

```
@protocol AnyChatTransDataDelegate <NSObject>
// 透明通道回调函数
- (void) OnAnyChatTransBufferCallBack:(int) dwUserId: (NSData*) lpBuf;
// 透明通道数据扩展回调函数
- (void) OnAnyChatTransBufferExCallBack:(int) dwUserId: (NSData*) lpBuf:
(int) wParam: (int) lParam: (int) dwTaskId;
// 文件传输回调函数
- (void) OnAnyChatTransFileCallBack:(int) dwUserId: (NSString*)
lpFileName: (NSString*) lpTempFilePath: (int) dwFileLength: (int) wParam:
(int) lParam: (int) dwTaskId;
// SDK Filter 通信数据回调函数
- (void) OnAnyChatSDKFilterDataCallBack:(NSData*) lpBuf;
@end
```

### 3.4.2. 收到文件传输数据事件

```
- (void) OnAnyChatTransFileCallBack:(int) dwUserId: (NSString*) lpFileName:
(NSString*) lpTempFilePath: (int) dwFileLength: (int) wParam: (int) lParam: (int)
dwTaskId;
```

参数:

dwUserId:	用户 ID, 指示发送用户
lpFileName:	文件名 (含扩展名, 不含路径)

**lpTempFilePath:** 接收完成后，SDK 保存在本地的临时文件（包含完整路径）

**dwFileLength:** 文件总长度

**wParam:** 附带参数 1

**lParam:** 附带参数 2

**dwTaskId:** 该文件所对应的任务编号

#### 说明:

当收到其它用户使用“TransFile”方法发送的文件时，将会触发该接口，等同于回调函数：**BRAC\_TransFile\_CallBack**。

特别提示：本 SDK 不会删除“lpTempFilePath”所指示的临时文件，上层应用在处理完毕后，需要主动删除该临时文件。

### 3.4.3. 收到透明通道数据事件

- (void) OnAnyChatTransBufferCallBack:(int) dwUserid: (NSData\*) lpBuf

#### 参数:

**dwUserid:** 用户 ID，指示发送用户

**lpBuf:** 缓冲区地址

#### 说明:

当收到其它用户使用“TransBuffer”方法发送的缓冲区数据时，将会触发该接口，等同于回调函数：**BRAC\_TransBuffer\_CallBack**。

由于该函数传递的数据是一个与本 SDK 无关的缓冲区（由上层应用自己填充内容），相对于本 SDK 来说是透明的，故称为透明通道，利用该通道，可以向当前房间内的任何用户传输上层应用自定义的数据。

### 3.4.4. 收到扩展透明通道数据事件

- (void) OnAnyChatTransBufferExCallBack:(int) dwUserid: (NSData\*) lpBuf: (int) wParam: (int) lParam: (int) dwTaskId

#### 参数:

**dwUserid:** 用户 ID，指示发送用户

lpBuf:	缓冲区地址
wParam:	缓冲区附带参数（由发送者设置，上层应用可自定义用途）
lParam:	缓冲区附带参数 2
dwTaskId:	该缓冲区所对应的传输任务编号

**说明：**

当收到其它用户使用“TransBufferEx”方法发送的缓冲区数据时，将会触发该接口，等同于回调函数：BRAC\_TransBufferEx\_CallBack。

### 3.4.5. 收到SDK Filter数据事件

- (void) OnAnyChatSDKFilterDataCallBack:(NSData\*) lpBuf

**参数：**

lpBuf:	缓冲区地址
--------	-------

**说明：**

当收到服务器“SDK Filter”或是“Server SDK”相关接口发送的缓冲区数据时，将会触发该接口，等同于回调函数：BRAC\_SDKFilterData\_CallBack。

## 3.5. 文字消息事件接口

### 3.5.1. 接口定义

```
@protocol AnyChatTextMsgDelegate <NSObject>
// 发送文字的回调函数
- (void) OnAnyChatTextMsgCallBack:(int) dwFromUserId: (int) dwToUserId:
(BOOL) bSecret: (NSString*) lpMsgBuf;
@end
```

### 3.5.2. 收到文字聊天消息事件

- (void) OnAnyChatTextMsgCallBack:(int) dwFromUserId: (int) dwToUserId:
(BOOL) bSecret: (NSString\*) lpMsgBuf;

**参数：**

dwFromUserid	消息发送者用户 ID
dwToUserid	目标用户, -1 表示发送给大家, 即房间所有人
bSecret	是否为悄悄话, 当目标用户不为-1 时有效
lpMsgBuf	消息字符串

**说明:** 当进入房间成功之后, 收到其他用户发送的文字聊天信息时将触发该接口, 等同于 WIN32 平台的回调函数: BRAC\_TextMessage\_CallBack。本地用户向其它用户发送文字消息时, 将不会触发该接口。

## 3.6. 私聊控制事件接口

### 3.6.1. 接口定义

```
@protocol AnyChatPrivateChatDelegate <NSObject>
// 用户聊天模式改变消息
- (void) OnAnyChatChatModeChg:(int) dwUserId: (BOOL) bPublicChat;
// 用户私聊请求消息
- (void) OnAnyChatPrivateRequest:(int) dwUserId: (int) dwRequestId;
// 用户私聊请求回复消息
- (void) OnAnyChatPrivateEcho:(int) dwUserId: (int) dwErrorCode;
// 用户退出私聊消息
- (void) OnAnyChatPrivateExit:(int) dwUserId: (int) dwErrorCode;
@end
```

### 3.6.2. 用户私聊请求事件

- (void) OnAnyChatPrivateRequest:(int) dwUserId: (int) dwRequestId

**参数:**

dwUserId	表示发起者的用户 ID 号, 私聊请求者
dwRequestId	表示私聊请求编号, 标识该请求

**说明:** 当进入房间成功之后, 当用户发起私聊请求时, 在被请求者端将触发该接口, 等同于 WIN32 平台的 WM\_GV\_PRIVATEREQUEST 消息。



### 3.6.3. 用户私聊回复事件

- (void) OnAnyChatPrivateEcho:(int) dwUserId: (int) dwErrorCode

参数:

dwUserId           表示回复者的用户 ID 号

dwErrorCode       返回代码，表示对方是否同意自己的邀请

**说明:** 当进入房间成功之后，当用户发起私聊请求时，其它用户回复该请求时将触发该接口，等同于 WIN32 平台的 WM\_GV\_PRIVATEECHO 消息。

### 3.6.4. 用户私聊退出事件

- (void) OnAnyChatPrivateExit:(int) dwUserId: (int) dwErrorCode

参数:

dwUserId           表示退出者的用户 ID 号

dwErrorCode       返回代码，表示退出私聊的原因

**说明:** 当进入房间成功之后，在自己私聊列表中的用户退出，或网络连接断开时将触发该接口，等同于 WIN32 平台的 WM\_GV\_PRIVATEEXIT 消息。

### 3.6.5. 用户私聊状态改变事件

- (void) OnAnyChatChatModeChg:(int) dwUserId: (BOOL) bPublicChat

参数:

dwUserId           表示状态改变的用户 ID 号

bPublicChat       当前是否为公聊模式

**说明:** 当进入房间成功之后，有用户在公聊状态和私聊状态之间切换时将触发该接口，等同于 WIN32 平台的 WM\_GV\_CHATMODECHG 消息。

## 4. 函数说明

### 4.1. 初始化与资源释放

#### 4.1.1. 初始化SDK

**+ (int) InitSDK: (int) dwFuncMode**

**功能：**初始化 SDK

**返回值：**0 表示成功，否则为出错代码

**参数：**

**flags**    功能模式组合，默认为 0，参考 WIN32 平台 SDK 的 InitSDK 的参数 dwFuncMode 定义

**备注：**

    功能模式组合可根据实际的需求灵活定义，如果在后续的方法调用中失败，则很有可能是某一项功能没有被定义，默认为 0，SDK 内部会自动设置常用的标志。

    该方法必须第一个被调用（SetSDKOptionString 方法除外），否则后续的其它方法调用将会返回没有初始化错误。

#### 4.1.2. 释放SDK资源

**+ (int) Release**

**功能：**释放 SDK 占用的所有资源

**返回值：**0 表示成功，否则为出错代码

**参数：**

    无。

**备注：**

    该方法必须最后一个被调用，调用该方法后，SDK 内部所占用的资源将被释放，如果在其后面再调用其它的方法，将会返回没有初始化的错误。

该方法通常在上层应用退出系统时被调用。

## 4.2. 登录流程

### 4.2.1. 设置服务器认证密码

**+ (int) SetServerAuthPass: (NSString\*) lpPassword**

**功能：**设置服务器连接认证密码，确保 SDK 能正常连接到服务器。

**返回值：**0 表示成功，否则为出错代码

**参数：**

lpPassword 认证密码（大小写敏感）；

**备注：**

为了防止未授权 SDK 连接服务器，在服务器配置文件（AnyChatCoreServer.ini）中可设置“SDKAuthPass”，如果该配置项被设置，当 SDK 连接服务器时，会将该方法所传入的密码加密后传输到服务器，服务器再比较是否合法，如果密码不正确，则连接将被断开。如果该配置项未被设置（配置文件默认），则无论该方法是否被调用，SDK 均可正常连接到服务器。

### 4.2.2. 连接服务器

**+ (int) Connect: (NSString\*) lpServerAddr : (int) dwPort**

**功能：**用于与服务器建立连接。

**返回值：**0 表示成功，否则为出错代码

**参数：**

lpServerAddr 服务器 IP 地址，或是网站域名（URL）地址；

dwPort 服务端口号（默认为 8906）

**备注：**

返回值为 0 并不表示连接服务器成功，仅表示 SDK 已成功收到连接服务器的指令，如果连接成功，或是失败，都将会通过相应的接口通知上层应用，这里是一个异步的过程。

### 4.2.3. 登录系统

**+ (int) Login: (NSString\*) lpUserName : (NSString\*) lpPassword**

**功能：**登录服务器，请求身份认证。

**返回值：**0 表示成功，否则为出错代码

**参数：**

lpUserName      注册用户名；

lpPassword      登录密码（为空表示游客）；

**备注：**

该方法可以连接系统之后立即调用，而不用关心连接系统是否成功，当 SDK 连接系统成功之后，如果之前调用过该方法，则 SDK 将会自动向服务器发出登录系统的申请。

返回值为 0 并不表示登录服务器成功，仅表示 SDK 已成功收到登录服务器的指令，如果登录成功，或是失败，都将会通过相应的接口通知上层应用，这里是一个异步的过程。

如果服务器配置了“SDK Filter Plus”插件，则客户端调用该方法后，将会触发其 API 接口：BRFP\_VerifyUser，用户名、密码参数将会作为参数传递给该 API 函数，由“SDK Filter Plus”完成用户的身份验证工作，服务器根据该 API 接口的返回值来判定是否通过身份验证，详细信息可参考文档《AnyChat SDK Filter Plus 开发指南》。

如果在服务器端使用“AnyChat Server SDK”开发了业务层服务器，则客户端调用该方法后，将会触发业务层服务器的回调函数“BRAS\_VerifyUser\_CallBack”，由业务层服务器完成用户的身份验证工作，服务器根据回调函数的返回值来判定是否通过身份验证，详细信息可参考文档《AnyChat Server SDK 开发指南》。

### 4.2.4. 进入房间（根据房间编号）

**+ (int) EnterRoom: (int) dwRoomid : (NSString\*) lpRoomPass**

**功能：**根据房间编号进入房间

**返回值：**0 表示成功，否则为出错代码

**参数：**

**dwRoomid**                      房间编号，系统唯一；

**lpRoomPass**                  房间密码（当房间需要密码时有效，如果没有可为空）；

**备注：**

该方法可以登录系统之后立即调用，而不用关心登录系统是否成功，当 SDK 登录系统成功之后，如果之前调用过该方法，则 SDK 将会自动向服务器发出进入房间的申请。

返回值为 0 并不表示进入房间成功，仅表示 SDK 已成功收到进入房间的指令，不论成功，或是失败，都将会通过相应的接口通知上层应用，这里是一个异步的过程。

用户必须进入一个房间，否则无法进行相关的操作，后续用户所有的操作都是在房间内操作，针对游戏，房间可以理解为游戏桌（一桌游戏对应一个房间），针对视频会议，房间可以理解为会议室。

如果服务器配置了“SDK Filter Plus”插件，则客户端调用该方法后，将会触发其 API 接口：BRFP\_PrepereEnterRoom，用户 ID、房间 ID、房间密码将会作为参数传递给该 API 函数，由“SDK Filter Plus”完成用户进入房间的验证工作，服务器根据该 API 接口的返回值来判定是否允许进入房间，详细信息可参考文档《AnyChat SDK Filter Plus 开发指南》。

如果在服务器端使用“AnyChat Server SDK”开发了业务层服务器，则客户端调用该方法后，将会触发业务层服务器的回调函数“BRAS\_PrepereEnterRoom\_CallBack”，由业务层服务器完成用户进入房间的验证工作，服务器根据该 API 接口的返回值来判定是否允许进入房间，详细信息可参考文档《AnyChat Server SDK 开发指南》。

#### 4.2.5. 进入房间（根据房间名称）

**+ (int) EnterRoomEx: (NSString\*) lpRoomName : (NSString\*) lpRoomPass**

**功能：**根据房间名称进入房间

**返回值：**0 表示成功，否则为出错代码

**参数：**

lpRoomName          房间名称；

lpRoomPass          房间密码（当房间需要密码时有效，如果没有可为空）；

**备注：**

该方法与“EnterRoom”功能相同，区别在于房间的标识不同，其中“EnterRoom”是用房间 ID 进入房间，而该方法是用房间名称进入房间，如果房间不存在，而且系统配置为自动创建房间时，将会由系统分配一个唯一的房间编号，通过异步消息事件接口（AnyChatNotifyMessageDelegate）返回给上层应用，上层应用可以通过方法“GetRoomName”来获取房间名称。

## 4.2.6. 离开房间

**+ (int) LeaveRoom: (int) dwRoomid**

**功能：**离开房间。

**返回值：**0 表示成功，否则为出错代码

**参数：**

Roomid 房间编号，为-1 表示退出当前房间

**备注：**

在用户变换房间之前，需要调用该方法离开房间，然后才能进入新的房间。

## 4.2.7. 注销系统

**+ (int) Logout**

**功能：**将用户从系统中注销。

**返回值：**0 表示成功，否则为出错代码

**参数：**

无

**备注：**

在切换用户（如用户换用其它的用户名登录系统）时需要先调用该方法，或是在退出系统前需要调用该方法，注销系统后，网络连接将被断开。

## 4.3. 音视频操作

### 4.3.1. 用户视频控制

**+ (int) UserCameraControl: (int) dwUserid : (BOOL) bOpen**

**功能：**用户视频控制，打开或关闭本地摄像头，或请求对方的视频

**返回值：**0 表示成功，否则为出错代码

**参数：**

dwUserid: 用户编号，为-1 表示对本地视频进行控制

bOpen 是否打开视频

**备注：**

对于本地用户，该方法是直接操作用户的摄像头，而对于其它用户，该方法只是向对方发送一个请求（取消）视频流的申请，并不会直接操作对方的摄像头。

### 4.3.2. 用户语音控制

**+ (int) UserSpeakControl: (int) dwUserid : (BOOL) bOpen**

**功能：**用户发言控制

**返回值：**0 表示成功，否则为出错代码

**参数：**

dwUserid 用户编号，为-1 表示对本地发言进行控制

bOpen 是否允许用户发言，当 dwUserid=-1 时，1 表示请求发言（拿 Mic），0 表示停止发言（放 Mic）

**备注：**

对于本地用户，该方法是直接操作用户的 Mic，而对于其它用户，该方法只

是向对方发送一个请求（取消）音频流的申请，并不会直接操作对方的 Mic。

### 4.3.3. 设置视频显示位置

**+ (int) SetVideoPos: (int) dwUserId : (NSObject\*) surface : (int) left : (int) top : (int) width : (int) height**

**功能：**设置视频显示位置，或是刷新视频显示

**返回值：**0 表示成功，否则为出错代码

**参数：**

userid      用户编号，为-1 表示操作自己的视频显示位置

surface     视频显示对象

dwLeft、dwTop、dwRight、dwBottom   位置信息，默认为 0

**备注：**

该方法在打开用户视频（UserCameraControl）之前调用。

对于其它用户的视频，surface 参数为 UIImageView 控件，将控件句柄通过 SetVideoPos 方法传递给 SDK，则当有视频数据到达时，内核将会自动将视频显示到该 UIImageView 控件上，不需要上层应用来处理视频的显示。

对于本地视频，采用 AVCaptureVideoPreviewLayer 进行本地视频显示，在初始化本地视频设备之前，通过调用 SetVideoPos 方法将需要显示视频的主窗体 View 对象传递给 SDK，当本地视频初始化成功之后，SDK 内核会调用该 View 的方法 “ OnLocalVideoInit ”，在该方法中，便可创建和初始化 AVCaptureVideoPreviewLayer 对象，实现本地视频的显示，当本地视频会话关闭，或是对象销毁时，SDK 内核会调用该 View 的方法 “OnLocalVideoRelease”，在该方法中，可释放前面创建的 AVCaptureVideoPreviewLayer 对象。详细信息可参考随 AnyChat for iOS 一同发布的 Demo 程序源代码。



## 4.4. 查询状态

### 4.4.1. 查询摄像头的状态

**+ (int) GetCameraState: (int) dwUserid**

**功能：**查询用户摄像头的状态

**返回值：**返回指定用户的摄像头状态，定义为：

- 0 没有摄像头
- 1 有摄像头但没有打开
- 2 摄像头已打开

**参数：**

userid 用户编号，为-1时表示获取自己的摄像头状态；

**备注：**

该方法必须在登录系统之后调用方才有效，根据返回参数的不同，可以判别用户当前摄像头的状态，以及判断用户是否有摄像头。

### 4.4.2. 查询用户音频设备采集状态

**+ (int) GetSpeakState: (int) dwUserid**

**功能：**查询用户音频设备采集状态

**返回值：**返回指定用户的音频设备状态，定义为：

- 0 音频采集关闭
- 1 音频采集开启

**参数：**

userid 用户编号，为-1时表示获取自己的音频设备状态；

**备注：**

这里所说的“音频设备采集状态”是指在 SDK 内部是否已开始音频采集，当返回值为 1 时，表示 SDK 已经开始采集，当有其它用户请求时，才对外传输。

关于实际应用中的“公麦”、“麦序”等属于业务逻辑范畴，具体的实现方式

可参考《AnyChat Server SDK 开发指南》中“常用业务处理逻辑”的章节。

### 4.4.3. 查询用户昵称

**+ (NSString\*) GetUserName: (int) dwUserId**

**功能：**查询用户昵称

**返回值：**用户昵称字符串

**参数：**

userid 用户编号，为-1时表示获取自己的昵称；

**备注：**

这里所查询到的用户昵称，是用户在身份验证时，服务器端调用 SDK Filter 的“BRGS\_VerifyUser”方法时，由 SDK Filter 返回给服务器的 lpNickName 参数值，如果 lpNickName 为空，则默认采用登录用户名替代用户昵称。

当用户离开房间之后（包括在 WM\_GV\_USERATROOM 消息中，状态为用户离开时）将会查询失败。

### 4.4.4. 查询用户IP地址

**+ (NSString\*) GetUserIPAddr: (int) dwUserId**

**功能：**查询用户互联网 IP 地址

**返回值：**用户 IP 字符串

**参数：**

userid 用户编号，为-1时表示获取自己的 IP 地址；

**备注：**

这里所查询到的用户 IP 为互联网 IP 地址，可能是用户本机的真实 IP，也可能是用户接入互联网的网关 IP 地址。

### 4.4.5. 查询用户视频宽度

**+ (int) GetUserVideoWidth: (int) dwUserId**

**功能：**查询用户视频分辨率的宽度值

**返回值：**返回指定用户的视频宽度，如果用户视频没有打开，或是打开视频设备失败，则获取的值为 0

**参数：**

userid 用户编号，为-1 时表示获取自己的视频宽度；

**备注：**

如查询本地的视频宽度值，则必须在打开本地视频设备成功之后方能查询成功；如查询其它用户的视频宽度值，则必须在对方打开视频设备成功，且状态同步到本地后，才能查询成功，故上层应用可用一个定时器间隔查询。

## 4.4.6. 查询用户视频高度

**+ (int) GetUserVideoHeight: (int) dwUserId**

**功能：**查询用户视频分辨率的高度值

**返回值：**返回指定用户的视频高度，如果用户视频没有打开，或是打开视频设备失败，则获取的值为 0

**参数：**

userid 用户编号，为-1 时表示获取自己的视频高度；

**备注：**

如查询本地的视频高度值，则必须在打开本地视频设备成功之后方能查询成功；如查询其它用户的视频高度值，则必须在对方打开视频设备成功，且状态同步到本地后，才能查询成功，故上层应用可用一个定时器间隔查询。

## 4.4.7. 查询用户状态（字符串）

**+ (NSString\*) QueryUserStateString: (int) dwUserId : (int) infoname**

**功能：**查询指定用户状态（字符串类型）

**返回值：**相关状态的字符串

**参数：**

dwUserId 用户编号，可用-1 代表本地用户（自己）；

infoname 需要查询的信息代码（见 WIN32 平台 SDK 相关定义）

**备注：**

通过调用该方法，可以查询指定用户的相关状态值。

## 4.4.8. 查询用户状态（数字）

**+ (int) QueryUserStateInt: (int) dwUserId : (int) infoname**

**功能：** 查询指定用户状态（字符串类型）

**返回值：** 相关状态值

**参数：**

dwUserId	用户编号，可用-1 代表本地用户（自己）；
infoname	需要查询的信息代码（见 WIN32 平台 SDK 相关定义）

**备注：**

通过调用该方法，可以查询指定用户的相关状态值。

## 4.4.9. 查询房间名称

**+ (NSString\*) GetRoomName: (int) dwRoomId**

**功能：** 查询根据房间 ID 获取房间名称

**返回值：** 返回房间名称，如查询失败，则返回空字符串

**参数：**

roomid	房间编号；
--------	-------

**备注：**

目前只能查询当前所在房间的房间名称，当用户离开房间后，查询将会失败。

## 4.5. 普通功能

### 4.5.1. 获取SDK版本信息

**+ (NSString\*) GetSDKVersion**

**功能：**获取 SDK 主版本号。

**返回值：**版本字符串，包含 SDK 的编译日期时间等信息

**参数：**无

### 4.5.2. 获取当前房间在线用户列表

**+ (NSMutableArray\*) GetOnlineUser**

**功能：**获取当前房间在线用户列表（不包含自己）

**返回值：**在线用户 ID 数组

**参数：**

无。

**备注：**

获取在线用户列表，并不包含当前用户自己的 ID，自己的 ID 在登录事件（AnyChatNotifyMessageDelegate）中已通知给上层应用。

### 4.5.3. 传送文本消息

**+ (int) SendMessage: (int) dwUserId: (BOOL) bSecret: (NSString\*) lpMsgBuf**

**功能：**向指定的用户传送文本消息

**返回值：**0 表示成功，否则为出错代码

**参数：**

**dwUserId:** 目标用户编号，-1 表示大家（所有人）

**secret:** 是否为密语，只在 dwUserId 不为-1 时有效，选择密语时，其它用户看不到发送的消息

**lpMsgBuf:** 消息字符串

**备注:**

可以利用该消息实现文字交流的功能，发送消息的对象可以是大家，也可以是指定的对象，如果是对指定的对象发送文字消息，可以选择密语。

对方收到该消息后，会触发 AnyChatTextMsgDelegate 接口的接口函数的调用。

## 4.5.4. 透明通道传送缓冲区

**+ (int) TransBuffer: (int) dwUserid : (NSData\*) lpBuf**

**功能:** 透明通道传送缓冲区

**返回值:** 0 表示成功，否则为出错代码

**参数:**

**dwUserid:** 目标用户编号，-1 表示大家（用户当前房间所有人）

**lpBuf:** 缓冲区

**备注:**

可以利用该方法实现自定义功能，缓冲区采用透明传输，目标对象可以是大家，也可以是指定的对象。

该方法将会触发对方的 AnyChatTransDataDelegate 接口的对应接口函数。

该 API 方法支持跨房间传输缓冲区数据，目标用户为指定用户时，目标用户可以与自己在不同的房间，或是目标用户没有进入任何房间，或是源用户（自己）没有进入任何房间，只要双方都登录服务器成功，则可利用该方法传输缓冲区，当目标用户编号为-1 时，则源用户（自己）必须已经在房间中，表示向该房间的其他用户广播数据（注：自己不能发送给自己）。

## 4.5.5. 透明通道传送缓冲区扩展

**+ (int) TransBufferEx: (int) dwUserid : (NSData\*) lpBuf: (int) wParam: (int) lParam: (int)**

**dwFlags**

**功能:** 透明通道传送缓冲区

**返回值:** >0 表示任务 ID 号（可利用该 ID 查询该任务的传输进度），否则表示出错。

**参数:**

**userid:** 目标用户编号，只针对某一个用户，不能为-1（所有人）

**lpBuf:** 缓冲区，≤1024KB（1MB），内部会自动分包处理

**wParam:** 附带参数，由上层应用自定义

**lParam:** 附带参数 2，由上层应用自定义

**dwFlags:** 特殊功能标志，当对该缓冲区有特殊要求时，可通过使用相关的功能标志，通知 SDK 进行特殊的处理，默认为 0，SDK 将自动根据网络状态选择合适的传输途径（TCP、UDP or P2P）

**备注:**

该方法与“TransBuffer”功能相同，都是传输上层应用自定义（透明通道）数据，区别在于该方法通过设置相应的功能标识，如可选择采用 UDP 通道传输，但是只针对指定的用户传输，而“TransBuffer”方法则固定采用 TCP 通道传输，且缓冲区大小不能超过 1024 个字节，但可以针对所有用户传输。

从应用来看：

（1）、TransBuffer 适合数据量小、要求实时传输的缓冲区传递，如控制指令等；

（2）、TransBufferEx 适合数据量大、对实时性要求不高的需求；

## 4.5.6. 传送文件

**+ (int) TransFile: (int) dwUserId: (NSString\*) lpLocalPathName: (int) wParam: (int) lParam: (int) dwFlags**

**功能:** 传送文件给指定用户

**返回值:** >0 表示任务 ID 号（可利用该 ID 查询该任务的传输进度），否则表示出错。

**参数:**

**userid:** 目标用户编号，只针对某一个用户，不能为-1（所有人）

**filepath:** 本地文件名，含路径

**wParam:** 附带参数 1，便于上层应用扩展

**lParam:** 附带参数 2

**dwFlags:** 特殊功能标志，参考：“TransBufferEx”方法

**备注：**

该方法传输效率与“TransBufferEx”方法相同，只是在 SDK 内部封装了文件的分组传输功能，实现对上层应用的透明，简化上层应用的开发难度。

## 4.5.7. 查询传输任务相关信息

**+ (int) QueryTransTaskInfoInt: (int) dwUserId: (int) dwTaskId: (int) infoname**

**功能：** 查询与传输任务相关的信息，如传输进度、传输状态、传输码率等

**返回值：** 状态值

**参数：**

**dwUserId:** 任务发起者用户编号（并非传输目标用户编号）

**dwTaskId:** 需要查询的任务编号

**infoname** 需要查询的信息代码（见备注附表）

**infoval** 查询信息值的保存参数地址

**infofen** 保存查询信息值参数类型所占内存大小

**备注：**

通过调用该方法，可以查询指定传输任务编号的缓冲区传输情况。用户编号与任务编号组合才具有唯一性，不同的用户可能存在相同的任务编号。

目前提供的查询的信息代码见下表：

信息代码定义	参数类型	用途		备注
TRANSTASK_PROGRESS	DOUBLE	传输任务进度查询		0.0 ~ 100.0
TRANSTASK_BITRATE	INT	传输任务当前码率		单位：bps
TRANSTASK_STATUS	INT	传输任务当前状态：		
		1	准备状态	
		2	传输状态	



		3	完成状态
		4	任务被发送者取消
		5	任务被接收方取消

### 4.5.8. 激活（关闭）SDK调用日志

**+ (int) ActiveCallLog: (BOOL) bActive**

**功能：**打开，或是关闭 SDK 调用期间所产生的日志

**返回值：**0 表示成功，否则为出错代码

**参数：**

**bActive:** 是否打开日志功能

**备注：**

通过调用该方法，可以记录应用程序调用该 SDK 的方法顺序及参数，对于分析利用该 SDK 进行开发所产生的异常现象有一定的帮助，建议在开发过程中打开该选项，在发布的版本中关闭该选项，所产生的日志文件名为：BRAnyChatCore.log，与调用者程序在同一级目录。

SDK 默认已打开日志记录功能，该方法必须在 SDK 初始化之前调用。

### 4.5.9. 发送SDK Filter通信数据

**+ (int) SendSDKFilterData: (NSData\*) lpBuf**

**功能：**向服务器发送 SDK Filter 通信数据

**返回值：**0 表示成功，否则为出错代码

**参数：**

**buf:** 缓冲区

**备注：**

服务器收到数据后，会将该缓冲区数据全部提交给 SDK Filter，由 SDK Filter 来解析，该缓冲区的内容对于本 SDK 和服务器来说，都是透明的。

## 4.5.10. 音视频录制

**+(int) StreamRecordCtrl:(int)dwUserId : (BOOL)bStartRecord : (int)dwFlags : (int) dwParam**

**功能：**对指定用户的音视频流进行录制，在中心录像服务器上保存音视频文件

**返回值：**0 表示录制指令被 SDK 成功接收，否则为出错代码

**参数：**

**dwUserId:** 需要录制视频的用户编号，可用-1 表示本地用户（自己）；

**bStartRecord:** 指示当前指令是启动录像，或是停止录像；

**dwFlags:** 录制功能标志，参考备注；

**dwParam:** 录制指令附带参数，录像任务结束时，该参数将通过回调函数返回给上层应用。

**备注：**

该方法只是向 SDK 下达（停止）录像任务，当指令（bStartRecord）为停止录像时，而且已经录制到了数据时，SDK 将产生一次回调，通知上层应用录像文件名。

录像功能标志指示 SDK 在录制时，进行特殊的处理，0 表示默认（音视频同步录制），目前支持如下标志组合：

BRAC\_RECORD\_FLAGS\_VIDEO 录制视频

BRAC\_RECORD\_FLAGS\_AUDIO 录制音频

BRAC\_RECORD\_FLAGS\_SERVER 服务器端录制

目前 AnyChat for iOS SDK 暂时不支持客户端录像。

在服务器端录制音视频，需要单独部署中心录像服务器，参考论坛相关介绍：

<http://bbs.anychat.cn/forum.php?mod=viewthread&tid=20&extra=page%3D1>。

## 4.6. 私聊函数

### 4.6.1. 更改当前的聊天模式

**+(int) ChangeChatMode: (int) dwChatMode**

**功能：**更改自己的聊天模式（默认为公聊模式）

**返回值：**0 表示成功，否则为出错代码

**参数：**

**dwChatMod:** 聊天模式，定义为：

**GV\_CHATMODE\_PUBLIC**      0    公聊模式

**GV\_CHATMODE\_PRIVATE**    1    私聊模式

**备注：**

不同的聊天模式下对语音视频数据的发送方式采取了不同的策略，公聊模式下可以接收所有的视频和语音信息，而私聊模式下只接收与自己私聊的用户的语音和视频数据，在私聊模式下，自己的语音视频数据也只向私聊对象传送。

该方法必须在进入房间成功之后调用方才有效，在私聊模式下，接收不到其它用户的语音和视频数据，仅能接收到自己私聊对象的语音和视频数据，文字消息不受模式影响，关于私聊模式与公聊模式之间的关系，请参考“关于私聊的实现思路”部分。

## 4.6.2. 获取指定用户聊天模式

**+ (int) GetUserChatMode: (int) dwUserId**

**功能：**获取指定用户当前的聊天模式。

**返回值：**0 表示成功，否则为出错代码

**参数：**

**dwUserId:** 用户 ID 号，-1 表示获取自己的聊天模式；

**dwChatMod:** 聊天模式

**备注：**

该方法必须在进入房间成功之后调用方才有效。

## 4.6.3. 请求与对方私聊

**+ (int) PrivateChatRequest: (int) dwUserId**

**功能：**请求与对方私聊，向对方发起私聊请求。

**返回值：**0 表示成功，否则为出错代码

**参数：**

dwUserid: 对方的用户 ID 号

**备注：**

该方法必须在进入房间成功之后调用方才有效。如果对方接受请求，则系统将会自动切换到私聊模式，调用该方法后，对方将会收到消息：WM\_GV\_PRIVATEREQUEST。

#### 4.6.4. 回复对方的私聊请求

**+ (int) PrivateChatEcho: (int) dwUserid: (int) dwRequestid : (BOOL) bAccept**

**功能：**回复对方的私聊请求。

**返回值：**0 表示成功，否则为出错代码

**参数：**

dwUserid: 对方的用户 ID 号

dwRequestid: 请求 ID 号，在请求消息中传过来，标示是哪一个请求

bAccept: 是否同意对方的私聊请求

**备注：**

该方法必须在进入房间成功之后调用方才有效。当收到其它用户的私聊请求时，可以调用该方法同意，或是拒绝对方的请求，调用该方法后，对方将会收到消息：WM\_GV\_PRIVATEECHO。

#### 4.6.5. 回复对方的私聊请求（扩展）

**+ (int) PrivateChatEchoEx: (int) dwUserid: (int) dwRequestid : (int) dwErrorCode**

**功能：**回复对方的私聊请求。

**返回值：**0 表示成功，否则为出错代码

**参数:**

**dwUserId:** 对方的用户 ID 号

**dwRequestid:** 请求 ID 号, 在请求消息中传过来, 标示是哪一个请求

**dwErrorCode:** 是否同意对方的私聊请求(**errorCode=GV\_ERR\_SUCCESS** 表示同意与对方私聊, 否则表示拒绝与对方私聊)

**备注:**

该方法与“PrivateChatEcho”功能相同, 但是该方法可以让上层传入相应的出错代码, 指示拒绝的原因, 便于上层扩展, 传入的 **errorCode** 将会附带在对方收到的 **WM\_GV\_PRIVATEECHO** 消息中。

## 4.6.6. 退出与某用户的私聊

**+ (int) PrivateChatExit: (int) dwUserId**

**功能:** 退出与某用户的私聊, 或者将某用户从自己的私聊列表中清除。

**返回值:** 0 表示成功, 否则为出错代码

**参数:**

**dwUserId:** 对方的用户 ID 号

**备注:**

该方法必须在进入房间成功之后调用方才有效。因为同时可以和多个用户之间进行私聊, 所以断开与该用户的私聊连接后, 不影响与其它用户的私聊, 调用该方法后, 自己和对方都将会收到消息: **WM\_GV\_PRIVATEEXIT**。

## 4.7. 好友列表

自 AnyChat for iOS SDK V1.9 版本开始, AnyChat 提供了一个轻量级的用户好友解决方案, 可以实现大厅好友列表、好友分组以及好友在线状态同步等功能。

好友列表需要业务服务器的支持, 详情请参考《AnyChat Server SDK 开发指南》以及示例程序 (AnyChatCallCenter) 源代码。

### 4.7.1. 获取好友列表

**+ (NSMutableArray\*) GetUserFriends;**

**功能：**获取本地用户的好友列表；

**返回值：**返回好友用户 ID 列表数组；

**参数：**无

**备注：**

当客户端登录系统成功之后，会触发异步消息：WM\_GV\_LOGINSYSTEM，同时服务器会向客户端发送用户好友信息，当客户端接收完成之后，会触发客户端的异步消息：WM\_GV\_USERINFOUPDATE，且该消息的 lParam 为 0，表示好友列表有更新。所以该 API 通常在接收到 WM\_GV\_USERINFOUPDATE 异步消息之后调用。

### 4.7.2. 获取好友在线状态

**+ (int) GetFriendStatus: (int) dwFriendUserId;**

**功能：**获取本地用户的好友在线状态，根据状态可以知道好友是否在线。

**返回值：**返回该好友的在线状态：0 离线， 1 在线

**参数：**

dwFriendUserId      好友的用户 ID；

**备注：**

登录成功之后调用有效。

### 4.7.3. 获取好友分组列表

**+ (NSMutableArray\*) GetUserGroups;**

**功能：**获取本地用户的好友分组列表，返回好友分组 ID 列表数组。

**返回值：**用户 ID 数组

**参数：**无

**备注：**

登录成功之后调用有效。好友分组是指将好友归纳到某一个组别下，如“家人”、“大学同学”以及“老师”等。每一个分组对应一个分组 ID，通过分组 ID 可以获取分组的名称。

#### 4.7.4. 获取分组名称

**+ (NSString\*) GetGroupName: (int) dwGroupId;**

**功能：**根据分组 ID 获取分组名称。

**返回值：**分组名称字符串

**参数：**

dwGroupId                  分组 ID;

**备注：**

登录成功之后调用有效。分组名称由业务服务器设置。

#### 4.7.5. 获取分组所对应的用户列表

**+ (NSMutableArray\*) GetGroupFriends: (int) dwGroupId;**

**功能：**获取分组所对应的用户列表，即该分组下有多少用户。

**返回值：**用户 ID 数组

**参数：**

dwGroupId                  分组 ID;

**备注：**

登录成功之后调用有效。通过该 API 接口，可以获得每一个分组下面的用户列表，进而可以获得该分组下每一个用户的详细信息。

## 4.7.6. 获取好友用户信息

**+ (NSString\*) GetUserInfo: (int) dwUserId : (int) dwInfoId;**

**功能：**获取好友用户的详细信息。

**返回值：**用户信息字符串

**参数：**

dwUserId            好友用户 ID;

dwInfoId            用户信息类型 ID，业务层可自定义;

**备注：**

登录成功之后调用有效。当业务服务器调用 API: BRAS\_SetUserInfo 设置了用户的信息之后，客户端便可以通过该 API 获取业务服务器所设置的信息，其中 dwInfoId 由业务层（上层应用）自己定义。

关于好友用户信息这一部分，对于 AnyChat 来说是透明的，业务服务器设置了什么样的信息，客户端便可以获取到什么样的信息，AnyChat 只是提供了一个信息传输的中间通道，业务层可以自由扩展。

## 4.7.7. 用户信息控制

**+ (int) UserInfoControl: (int) dwUserId : (int) dwCtrlCode : (int) wParam : (int) lParam : (NSString\*) lpStrValue;**

**功能：**对用户信息进行控制。

**返回值：**0 表示成功，否则为出错代码

**参数：**

dwUserId            用户 ID;

dwCtrlCode          控制代码，业务层自定义，其中<100 的值为系统保留，业务层使用时，其值必须>100

wParam              附带参数，业务层自定义;

lParam               附带参数，业务层自定义;

lpStrValue           附带参数（字符串类型），业务层自定义，可为空;



**备注：**

登录成功之后调用有效。该 API 调用之后，会向业务服务器发送信息控制指令，将会触发业务服务器对应的回调函数。

## 4.8. 系统设置

### 4.8.1. 枚举本地视频采集设备

**+ (NSMutableArray\*) EnumVideoCapture**

**功能：**枚举本地视频采集设备

**返回值：**返回设备列表

**参数：**

无

### 4.8.2. 选择指定的视频采集设备

**+ (int) SelectVideoCapture: (NSString\*) szCaptureName**

**功能：**选择指定的视频采集设备

**返回值：**0 表示成功，否则为出错代码

**参数：**

szCaptureName      所获取设备的名称；

**备注**

当用户有多个视频采集设备时，可以通过该方法选用指定的视频采集设备。  
采集设备名称必须是调用“EnumVideoCapture”方法枚举得到。

### 4.8.3. 获取当前视频采集设备

**+ (NSString\*) GetCurVideoCapture**

**功能：**获取当前使用的视频采集设备名称

**返回值：**当前已打开的视频采集设备名称

**参数：**

无；

**备注**

当设备没有打开时，返回空值。

## 4.8.4. 枚举本地音频采集设备

**+ (NSMutableArray\*) EnumAudioCapture**

**功能：**枚举本地音频采集设备

**返回值：**返回设备列表

**参数：**

无

## 4.8.5. 选择指定的音频采集设备

**+ (int) SelectAudioCapture: (NSString\*) szCaptureName**

**功能：**选择指定的音频采集设备

**返回值：**0 表示成功，否则为出错代码

**参数：**

szCaptureName      所获取设备的名称；

**备注**

当用户有多个音频采集设备时，可以通过该方法选用指定的音频采集设备。  
采集设备名称必须是调用“EnumAudioCapture”方法枚举得到。

## 4.8.6. 获取当前音频采集设备

**+ (NSString\*) GetCurAudioCapture**

**功能：**获取当前使用的音频采集设备

**返回值：**当前已打开的音频采集设备名称

**参数：**

无

**备注：**

当设备没有打开时，返回空值。

## 4.8.7. 枚举本地音频播放设备

**+ (NSMutableArray\*) EnumAudioPlayback**

**功能：**枚举本地音频采集设备

**返回值：**返回设备列表

**参数：**

无

**备注：**

iPhone 设备有两个播放设备：Receiver（听筒）、Speaker（喇叭）。AnyChat 默认使用 Speaker 播放声音，如需转到 Receiver 则需要进行选择。

## 4.8.8. 选择指定的音频播放设备

**+ (int) SelectAudioPlayback: (NSString\*) szDeviceName**

**功能：**选择指定的音频播放设备

**返回值：**0 表示成功，否则为出错代码

**参数：**

szDeviceName          播放设备名称；

**备注**

可以通过该方法选用指定的音频播放设备，在 Receiver 和 Speaker 之间切换。  
播放设备名称必须是调用“EnumAudioPlayback”方法枚举得到。

### 4.8.9. 获取当前音频播放设备

**+ (NSString\*) GetCurAudioPlayback**

**功能：**获取当前使用的音频播放设备

**返回值：**当前已打开的音频播放设备名称

**参数：**

无

**备注：**

当设备没有打开时，返回空值。

### 4.8.10. 获取音频设备的当前音量

**+ (int) AudioGetVolume:(int) device**

**功能：**获取指定音频设备的当前音量

**返回值：**0 表示成功，否则为出错代码

**参数：**

**device** 设备类型，定义为：

AD\_WAVEIN = 0, ///< 输入设备: Mic

AD\_WAVEOUT = 1, ///< 输出设备: Wave

**dwVolume** 保存该设备的当前音量，取值范围：0~100；

**备注**

根据设备类型（**device**）参数的不同，可以获取放音设备（**WaveOut**）和录音设备（**WaveIn**）的当前音量大小。

### 4.8.11. 设置指定音频设备的音量

**+ (int) AudioSetVolume:(int) device : (int) dwVolume**

**功能：**设置指定音频设备的音量

**返回值：**0 表示成功，否则为出错代码

**参数：**

device        设备类型，定义为：

AD\_WAVEIN    = 0,                    ///< 输入设备：Mic

AD\_WAVEOUT   = 1,                    ///< 输出设备：Wave

dwVolume    需要设置的音量，取值范围：0~100，值越大，音量越大；

**备注**

根据设备类型（device）参数的不同，可以调节放音设备（WaveOut）和录音设备（WaveIn）的音量大小。

## 4.8.12. SDK内核参数设置

### **+ (int) SetSDKOptionInt:(int) optname: (int) value**

**功能：**SDK 内核参数设置（整形值参数）

**返回值：**0 表示成功，否则为出错代码

**参数：**

optname       内核参数名称；

value         设置的参数值

**备注**

可以通过该方法对 AnyChat Core SDK 内部的参数进行设置，实现特殊的功能要求。

目前提供的可设置内核参数名称代码见 WIN32 平台 SDK 相关定义。

### **+ (int) SetSDKOptionString:(int) optname: (NSString\*) value**

**功能：**SDK 内核参数设置（字符串值参数）

**返回值：**0 表示成功，否则为出错代码

**参数：**

optname       内核参数名称；

value         设置的参数值

**备注**

可以通过该方法对 AnyChat Core SDK 内部的参数进行设置，实现特殊的功能要求。

目前提供的可设置内核参数名称代码见 WIN32 平台 SDK 相关定义。

### 4.8.13. SDK内核参数状态查询

**+ (int) GetSDKOptionInt:(int) optname**

**功能：**SDK 内核参数状态查询（整形值）

**返回值：**返回查询结果

**参数：**

optname     内核参数名称；

**备注**

可以通过该方法对 AnyChat Core SDK 内部的参数进行状态查询，获取当前的设置。

**+ (NSString\*) GetSDKOptionString:(int) optname**

**功能：**SDK 内核参数状态查询（字符串）

**返回值：**返回查询结果

**参数：**

optname     内核参数名称；

**备注**

可以通过该方法对 AnyChat Core SDK 内部的参数进行状态查询，获取当前的设置。

## 5. 版本变更记录

2011-09-05 V1.0

初始版本，采用 AnyChat Platform Core V4.1 内核

2011-11-15 V1.2

基于 AnyChat Platform Core SDK V4.2 版本编译；

2012-02-20 V1.3

基于 AnyChat Platform Core SDK V4.3 版本编译；

2012-05-11 V1.4

基于 AnyChat Platform Core SDK V4.4 版本编译；

2012-09-10 V1.5

基于 AnyChat Platform Core SDK V4.5 版本编译；

2012-11-22 V1.6

基于 AnyChat Platform Core SDK V4.6 版本编译；

2013-03-20 V1.7

基于 AnyChat Platform Core SDK V4.7 版本编译；

2013-07-28 V1.8

基于 AnyChat Platform Core SDK V4.8 版本编译；

2014-01-01 V1.9

基于 AnyChat Platform Core SDK V4.9 版本编译；

Android、iOS、Web 等平台支持中心服务器录像；

iOS 平台支持视频分辨率参数设置；

修正 iOS 设备水平放置时视频会自动旋转的问题；

增加 VP8 编码器

提供完整的视频呼叫解决方案；

提供完整的大厅好友解决方案；

修正 iOS 平台下文件传输事件回调中路径为空的 Bug

优化回声消除算法，提高通话过程中的用户体验；

支持最新的 iOS 7 平台；

## 6.附录一：错误代码参考

#define GV_ERR_SUCCESS	0	///< 成功
#define GV_ERR_DB_ERROR	1	///< 数据库错误
#define GV_ERR_NOTINIT	2	///< 系统没有初始化
#define GV_ERR_NOTINROOM	3	///< 还未进入房间
#define GV_ERR_FUNCNOTALLOW	20	///< 函数功能不允许（初始化时没有指定该功能）
//连接部分		
#define GV_ERR_CONNECT_TIMEOUT	100	///< 连接服务器超时
#define GV_ERR_CONNECT_ABORT	101	///< 与服务器的连接中断
#define GV_ERR_CONNECT_AUTHFAIL	102	///< 未能通过服务器的认证，属于非法连接
//登录部分		
#define GV_ERR_CERTIFY_FAIL	200	///< 认证失败，用户名或密码有误
#define GV_ERR_ALREADY_LOGIN	201	///< 该用户已登录
#define GV_ERR_ACCOUNT_LOCK	202	///< 帐户已被暂时锁定
#define GV_ERR_IPADDR_LOCK	203	///< IP 地址已被暂时锁定
#define GV_ERR_VISITOR_DENY	204	///< 游客登录被禁止（登录时没有输入密码）
#define GV_ERR_INVALID_USERID	205	///< 无效的用户 ID（用户不存在）
//进入房间		
#define GV_ERR_ROOM_LOCK	300	///< 房间已被锁住，禁止进入
#define GV_ERR_ROOM_PASSERR	301	///< 房间密码错误，禁止进入
#define GV_ERR_ROOM_FULLUSER	302	///< 房间已满员，不能进入
#define GV_ERR_ROOM_INVALID	303	///< 房间不存在
#define GV_ERR_ROOM_EXPIRE	304	///< 房间服务时间已到期
#define GV_ERR_ROOM_REJECT	305	///< 房主拒绝进入
#define GV_ERR_ROOM_OWNERBEOUT	306	///< 房主不在，不能进入房间
#define GV_ERR_ROOM_ENTERFAIL	307	///< 不能进入房间
#define GV_ERR_ROOM_ALREADYIN	308	///< 已经在房间里面了，本次进入房间请求忽略
//私聊		
#define GV_ERR_ROOM_PRINULL	401	///< 用户已经离开房间
#define GV_ERR_ROOM_REJECTPRI	402	///< 用户拒绝了私聊邀请
#define GV_ERR_ROOM_PRIDENY	403	///< 不允许与该用户私聊，或是用户禁止私聊
#define GV_ERR_ROOM_PRIREQIDERR	420	///< 私聊请求 ID 号错误，或请求不存在
#define GV_ERR_ROOM_PRIALRCHAT	421	///< 已经在私聊列表中

更多错误代码可参考：SDK\Client\AnyChatErrorCode.h 文件。