AnyChat for iOS SDK

开发手册

（版本：V2.3）

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

2015 年 07 月

目 录

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、P2P 网络模块、音视频录制、文件传输、数据通信以及常见业务流程封装等模块，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，**支持 iOS 全系列架构（i386、x86_64、armv7、armv7s、arm64）**，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 NAPT <——> Cone NAPT

Cone NAPT <——> 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.）始创于 2005 年，是国内领先的网络语音视频技术研究与系统开发的高新技术企业。十年来我们一直专注于音视频核心技术、基础数据通信能力研究，为客户提供专业的音视频通信能力解决方案，打造了一款拥有独立知识产权、世界

一流的跨平台音视频解决方案产品“AnyChat”，该产品已成功应用于移动互联网、物联网、在线教育、远程医疗、视频客服以及智能家居等业务领域，某些应用领域占据整个行业 90%以上的市场份额。

成立十年以来，佰锐科技一直以“追求企业价值与客户价值共同成长”的经营理念，已为数千家客户提供了优质的服务，凭借资深的研发团队和良好的口碑，在激烈的竞争中脱颖而出，成为首屈一指的音视频能力解决方案提供商。

更多信息请参考佰锐科技官网：http://www.bairuitech.com

1.4 技术支持

在您使用 AnyChat 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、公司官网：http://www.bairuitech.com

3、AnyChat产品网站：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

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列表	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），则需要进行额外的处理，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 数据结构及常量定义

3.1 视频图像格式

```
enum BRAC_PixelFormat{
    BRAC_PIX_FMT_RGB24 = 0,    ///< Packed RGB 8:8:8, 24bpp, RGBRGB...（MEDIASUBTYPE_RGB24）
    BRAC_PIX_FMT_RGB32,        ///< 对应于：MEDIASUBTYPE_RGB32，Packed RGB 8:8:8, 32bpp, (msb)8A 8R 8G 8B(lsb), in cpu endianness
    BRAC_PIX_FMT_YV12,         ///< 对应于：MEDIASUBTYPE_YV12，Planar YUV 4:2:0, 12bpp, (1 Cr & Cb sample per 2x2 Y samples)
    BRAC_PIX_FMT_YUY2,         ///< 对应于：MEDIASUBTYPE_YUY2，Packed YUV 4:2:2, 16bpp, Y0 Cb Y1 Cr
    BRAC_PIX_FMT_YUV420P,      ///< Planar YUV 4:2:0, 12bpp, (1 Cr & Cb sample per 2x2 Y samples)
    BRAC_PIX_FMT_RGB565,       ///< 对应于：MEDIASUBTYPE_RGB565
    BRAC_PIX_FMT_RGB555,       ///< 对应于：MEDIASUBTYPE_RGB555
    BRAC_PIX_FMT_NV12,         ///< Planar YUV 4:2:0, 12bpp, Two arrays, one is all Y, the other is U and V
    BRAC_PIX_FMT_NV21,         ///< Planar YUV 4:2:0, 12bpp, Two arrays, one is all Y, the other is V and U
    BRAC_PIX_FMT_NV16,         ///< YUV422SP

    BRAC_PIX_FMT_MJPEG = 200,
    BRAC_PIX_FMT_H264,
};
```

该图像格式将会在请求视频数据回调函数中使用，表示上层需要的视频格式，如果从摄像头原始采集的格式与所请求的格式不同，则 SDK 将会自动进行转换，然后再回调。

3.2 功能模式定义

API： InitSDK 传入参数，用于确定 SDK 的一些功能是否生效。

```
#define BRAC_FUNC_VIDEO_CBDATA          0x00000001L    ///< 通过回调函数输出视频数据
#define BRAC_FUNC_VIDEO_AUTODISP        0x00000002L    ///< 由 SDK 包处理视频，将视频显示在指定的窗口上
#define BRAC_FUNC_AUDIO_CBDATA          0x00000004L    ///< 通过回调函数输出音频数据
#define BRAC_FUNC_AUDIO_AUTOPLAY        0x00000008L    ///< 由 SDK 包处理音频，直接播放
#define BRAC_FUNC_CONFIG_LOCALINI       0x00000010L    ///< 生成本地配置文件（AnyChatSDK.ini）
#define BRAC_FUNC_FIREWALL_OPEN         0x00000020L    ///< 允许 SDK 操作 Windows 防火墙，将当前应用程序加入防火墙访问列表（避免 Windows 提示用户是否阻止当前应用程序）
```

#define BRAC_FUNC_CHKDEPENDMODULE 0x00000040L ///< 自动检查 SDK 所依赖的组件，并自动注册
#define BRAC_FUNC_AUDIO_VOLUMECALC 0x00000080L ///< 由 SDK 自动计算语音的音量
#define BRAC_FUNC_AUDIO_AUTOVOLUME 0x00000100L ///< 允许 SDK 自动控制 Mic 录音音量
#define BRAC_FUNC_NET_SUPPORTUPNP 0x00000200L ///< 允许 SDK 打开用户网络中的 UPNP 设备，如果用户的路由器或是防火墙支持 UPNP 协议，则可提高 P2P 打洞的成功率
#define BRAC_FUNC_DISABLEDECODE 0x00000400L ///< 禁止对收到的数据进行解码和播放，为了提高代理客户端的数据转发性能，可设置该标志，否则不能设置该标志

3.3 SDK 内核参数定义

在使用 API“SetSDKOptionInt”、“GetSDKOptionInt”来设置、获取 SDK 内核心参数时，可根据不同的参数类型设置、获取不同的值，目前可用参数类型定义如下：

#define BRAC_SO_AUDIO_VADCTRL 1 ///< 音频静音检测控制（参数为：int 型：1 打开，0 关闭）
#define BRAC_SO_AUDIO_NSCTRL 2 ///< 音频噪音抑制控制（参数为：int 型：1 打开，0 关闭）
#define BRAC_SO_AUDIO_ECHOCTRL 3 ///< 音频回音消除控制（参数为：int 型：1 打开，0 关闭）
#define BRAC_SO_AUDIO_AGCCTRL 4 ///< 音频自动增益控制（参数为：int 型：1 打开，0 关闭）
#define BRAC_SO_AUDIO_CPATUREMODE 5 ///< 音频采集模式设置（参数为：int 型：0 发言模式，1 放歌模式，2 卡拉 OK 模式，3 线路输入模式）
#define BRAC_SO_AUDIO_MICBOOST 6 ///< 音频采集 Mic 增强控制（参数为：int 型：0 取消，1 选中，2 设备不存在[查询时返回值]）
#define BRAC_SO_AUDIO_AUTOPARAM 7 ///< 根据音频采集模式，自动选择合适的相关参数，包括编码器、采样参数、码率参数等（参数为 int 型：1 启用，0 关闭[默认]）
#define BRAC_SO_AUDIO_MONOBITRATE 8 ///< 设置单声道模式下音频编码目标码率（参数为：int 型，单位：bps）
#define BRAC_SO_AUDIO_STEREOBITRATE 9 ///< 设置双声道模式下音频编码目标码率（参数为：int 型，单位：bps）
#define BRAC_SO_AUDIO_PLAYDRVCTRL 70 ///< 音频播放驱动选择（参数为：int 型，0 默认驱动， 1 DSound 驱动， 2 WaveOut 驱动）

#define BRAC_SO_RECORD_VIDEOBR 10 ///< 录像视频码率设置（参数为：int 型，单位：bps）
#define BRAC_SO_RECORD_AUDIOBR 11 ///< 录像音频码率设置（参数为：int 型，单位：bps）
#define BRAC_SO_RECORD_TMPDIR 12 ///< 录像文件临时目录设置（参数为字符串 TCHAR 类型，必须是完整的绝对路径）
#define BRAC_SO_SNAPSHOT_TMPDIR 13 ///< 快照文件临时目录设置（参数为字符串 TCHAR 类型，必须是完整的绝对路径）
#define BRAC_SO_RECORD_FILETYPE 140 ///< 录制文件类型设置（参数为：int 型， 0 MP4[默认], 1 WMV, 2 FLV, 3 MP3）
#define BRAC_SO_RECORD_WIDTH 141 ///< 录制视频宽度设置（参数为：int 型，如：320）
#define BRAC_SO_RECORD_HEIGHT 142 ///< 录制文件高度设置（参数为：int 型，如：240）

#define BRAC_SO_RECORD_FILENAMERULE 143 ///< 录制文件名命名规则（参数为：int 型）
#define BRAC_SO_RECORD_CLIPMODE 144 ///< 录制视频裁剪模式（参数为：int 型）

#define BRAC_SO_CORESDK_TMPDIR 14 ///< 设置 AnyChat Core SDK 临时目录（参数为字符串 TCHAR 类型，必须是完整的绝对路径）
#define BRAC_SO_CORESDK_PATH 20 ///< 设置 AnyChat Core SDK 相关组件路径（参数为字符串 TCHAR 类型，必须是完整的绝对路径）
#define BRAC_SO_CORESDK_DUMPCOREINFO 21 ///< 输出内核信息到日志文件中，便于分析故障原因（参数为：int 型：1 输出）
#define BRAC_SO_CORESDK_MAINVERSION 22 ///< 查询 SDK 主版本号号（参数为 int 型）
#define BRAC_SO_CORESDK_SUBVERSION 23 ///< 查询 SDK 从版本号（参数为 int 型）
#define BRAC_SO_CORESDK_BUILDTIME 24 ///< 查询 SDK 编译时间(参数为字符串 TCHAR 类型）
#define BRAC_SO_CORESDK_ACTIVESTATE 25 ///< 应用程序活动状态控制（参数为 int 型， 1 应用程序处于活动状态， 0 应用程序处于非活动状态），适用于 iPhone 等设备程序可后台运行的场合
#define BRAC_SO_CORESDK_EXTVIDEOINPUT 26 ///< 外部扩展视频输入控制（参数为 int 型， 0 关闭外部视频输入[默认]， 1 启用外部视频输入）
#define BRAC_SO_CORESDK_EXTAUDIOINPUT 27 ///< 外部扩展音频输入控制（参数为 int 型， 0 关闭外部音频输入[默认]， 1 启用外部音频输入）

#define BRAC_SO_LOCALVIDEO_BITRATECTRL 30 ///< 本地视频编码码率设置（参数为 int 型，单位 bps，同服务器配置：VideoBitrate）
#define BRAC_SO_LOCALVIDEO_QUALITYCTRL 31 ///< 本地视频编码质量因子控制(参数为 int 型，同服务器配置：VideoQuality）
#define BRAC_SO_LOCALVIDEO_GOPCTRL 32 ///< 本地视频编码关键帧间隔控制（参数为 int 型，同服务器配置：VideoGOPSize）
#define BRAC_SO_LOCALVIDEO_FPSCTRL 33 ///< 本地视频编码帧率控制（参数为 int 型，同服务器配置：VideoFps）
#define BRAC_SO_LOCALVIDEO_PRESETCTRL 34 ///< 本地视频编码预设参数控制(参数为 int 型，1-5）
#define BRAC_SO_LOCALVIDEO_APPLYPARAM 35 ///< 应用本地视频编码参数，使得前述修改即时生效（参数为 int 型：1 使用新参数，0 使用默认参数）
#define BRAC_SO_LOCALVIDEO_VIDEOSIZEPOLITIC 36 ///< 本地视频采集分辩率控制策略（参数为 int 型，0 自动向下逐级匹配[默认]；1 使用采集设备默认分辩率），当配置的分辩率不被采集设备支持时有效
#define BRAC_SO_LOCALVIDEO_DEINTERLACE 37 ///< 本地视频反交织参数控制(参数为 int 型：0 不进行反交织处理[默认]；1 反交织处理），当输入视频源是隔行扫描源（如电视信号）时通过反交织处理可以提高画面质量
#define BRAC_SO_LOCALVIDEO_WIDTHCTRL 38 ///< 本地视频采集分辨率宽度控制（参数为 int 型，同服务器配置：VideoWidth）
#define BRAC_SO_LOCALVIDEO_HEIGHTCTRL 39 ///< 本地视频采集分辨率高度控制（参数为 int 型，同服务器配置：VideoHeight）
#define BRAC_SO_LOCALVIDEO_FOCUSCTRL 90 ///< 本地视频摄像头对焦控制(参数为 int 型，1 表示自动对焦， 0 表示手动对焦）

#define BRAC_SO_LOCALVIDEO_PIXFMTCTRL 91 ///< 本地视频采集优先格式控制(参数为 int 型，-1 表示智能匹配，否则优先采用指定格式，参考：BRAC_PixelFormat）
#define BRAC_SO_LOCALVIDEO_OVERLAY 92 ///< 本地视频采用 Overlay 模式（参数为 int 型，1 表示采用 Overlay 模式， 0 表示普通模式[默认]）
#define BRAC_SO_LOCALVIDEO_CODECID 93 ///< 本地视频编码器 ID 设置（参数为 int 型，-1 表示默认，如果设置的编码器 ID 不存在，则内核会采用默认的编码器）
#define BRAC_SO_LOCALVIDEO_ORIENTATION 97 ///< 本地视频设备方向（参数为：int 型，定义为常量：ANYCHAT_DEVICEORIENTATION_XXXX）
#define BRAC_SO_LOCALVIDEO_AUTOROTATION 98 ///< 本地视频自动旋转控制（参数为 int 型， 0 表示关闭， 1 开启[默认]，视频旋转时需要参考本地视频设备方向参数）

#define BRAC_SO_NETWORK_P2PPOLITIC 40 ///< 本地网络 P2P 策略控制（参数为：int 型：0 禁止本地 P2P，1 服务器控制 P2P[默认]，2 上层应用控制 P2P 连接，3 按需建立 P2P 连接）
#define BRAC_SO_NETWORK_P2PCONNECT 41 ///< 尝试与指定用户建立 P2P 连接（参数为 int 型，表示目标用户 ID)，连接建立成功后，会通过消息反馈给上层应用，P2P 控制策略=2 时有效
#define BRAC_SO_NETWORK_P2PBREAK 42 ///< 断开与指定用户的 P2P 连接(参数为 int 型，表示目标用户 ID）[暂不支持，保留]
#define BRAC_SO_NETWORK_TCPSERVICEPORT 43 ///< 设置本地 TCP 服务端口（参数为 int 型，连接服务器之前设置有效
#define BRAC_SO_NETWORK_UDPSERVICEPORT 44 ///< 设置本地 UDP 服务端口（参数为 int 型，连接服务器之前设置有效
#define BRAC_SO_NETWORK_MULTICASTPOLITIC 45 ///< 组播策略控制（参数为 int 型：0 执行服务器路由策略，禁止组播发送[默认]， 1 忽略服务器路由策略，只向组播组广播媒体流， 2 执行服务器路由策略，同时组播）
#define BRAC_SO_NETWORK_TRANSBUFMAXBITRATE 46 ///< 传输缓冲区、文件最大码率控制（参数为 int 型，0 不限制，以最快速率传输[默认]， 否则表示限制码率，单位为：bps）
#define BRAC_SO_NETWORK_AUTORECONNECT 47 ///< 网络掉线自动重连功能控制（参数为 int 型，0 关闭， 1 开启[默认]）

#define BRAC_SO_PROXY_FUNCTIONCTRL 50 ///< 本地用户代理功能控制,（参数为：int 型，1 启动代理，0 关闭代理[默认]）
#define BRAC_SO_PROXY_VIDEOCTRL 51 ///< 本地用户代理视频控制，将本地视频变为指定用户的视频对外发布（参数为 int 型，表示其它用户的 userid）
#define BRAC_SO_PROXY_AUDIOCTRL 52 ///< 本地用户代理音频控制，将本地音频变为指定用户的音频对外发布（参数同 BRAC_SO_PROXY_VIDEOCTRL）

#define BRAC_SO_STREAM_BUFFERTIME 60 ///< 流缓冲时间（参数为 int 型，单位：毫秒，取值范围：500～5000，默认：800）

#define BRAC_SO_VIDEOSHOW_MODECTRL 80 ///< 视频显示模式控制（参数为：int 型，0 单画面显示，1 视频迭加显示）
#define BRAC_SO_VIDEOSHOW_SETPRIMARYUSER 81 ///< 设置主显示用户编号(参数为：int 型，用户 ID 号）
#define BRAC_SO_VIDEOSHOW_SETOVERLAYUSER 82 ///< 设置迭加显示用户编号（参数为：int

型，用户 ID 号）
#define BRAC_SO_VIDEOSHOW_DRIVERCTRL 83 ///< 视频显示驱动控制（参数为：int 型，0 默认驱动， 1 Windows DirectShow，2 Windows GDI，3 SDL）
#define BRAC_SO_VIDEOSHOW_GPUDIRECTRENDER 84 ///< 视频数据经过 GPU 直接渲染，将解码后的视频数据直接传输到 GPU 的物理地址（参数为：int 型， 0 关闭[默认]， 1 打开），与硬件平台相关
#define BRAC_SO_VIDEOSHOW_AUTOROTATION 85 ///< 远程视频显示自动旋转控制（参数为 int 型， 0 表示关闭， 1 开启[默认]，视频旋转时需要参考本地视频设备方向参数）
#define BRAC_SO_VIDEOSHOW_CLIPMODE 86 ///< 远程视频显示旋转裁剪模式（参数为 int 型， 0 自动[默认]）
#define BRAC_SO_VIDEOSHOW_CBPIXFMT 87 ///< 视频数据回调格式（参数为 int 型）

#define BRAC_SO_CORESDK_TICKOUTUSER 110 ///< 从服务器上踢掉指定用户（参数为 int 型，表示目标用户 ID）
#define BRAC_SO_CORESDK_DEVICEMODE 130 ///< 设备模式控制（局域网设备之间可以互相通信，不依赖服务器；参数为 int 型，0 关闭[默认]，1 开启）
#define BRAC_SO_CORESDK_DATAENCRYPTION 132 ///< 数据加密控制（参数为：int 型， 0 关闭[默认]， 1 开启）
#define BRAC_SO_CORESDK_UPLOADLOGINFO 134 ///< 上传日志信息到客户端（参数为：int 型，0 关闭[默认]， 1 开启）
#define BRAC_SO_CORESDK_WRITELOG 135 ///< 写入调试信息到日志文件中

#define BRAC_SO_UDPTRACE_MODE 160 ///< UDP 数据包跟踪模式
#define BRAC_SO_UDPTRACE_PACKSIZE 161 ///< UDP 数据包跟踪的大小，单位：BYTE
#define BRAC_SO_UDPTRACE_BITRATE 162 ///< UDP 数据包跟踪的包速率，单位：bps
#define BRAC_SO_UDPTRACE_START 163 ///< UDP 数据包跟踪控制（参数为 int 型，1 启动， 0 停止）
#define BRAC_SO_UDPTRACE_LOCALRECVNUM 164 ///< UDP 数据包跟踪本地接收包数量
#define BRAC_SO_UDPTRACE_SERVERRECVNUM 165 ///< UDP 数据包跟踪服务器接收包数量
#define BRAC_SO_UDPTRACE_SOURCESENDNUM 166 ///< UDP 数据包跟踪源发包数量
#define BRAC_SO_UDPTRACE_SENDUSERID 167 ///< UDP 数据包跟踪源用户 ID

#define BRAC_SO_OBJECT_INITFLAGS 200 ///< 业务对象身份初始化

3.4 用户状态标志定义

在使用 API“QueryUserStateInt”查询用户的相关状态时，可根据不同的状态标志来返回不同的值，目前支持的用户状态类型定义如下：

#define BRAC_USERSTATE_CAMERA 1 ///< 用户摄像头状态(参数为 DWORD 型）
#define BRAC_USERSTATE_HOLDMIC 2 ///< 用户音频设备状态（参数为 DWORD 型，返回值：0 音频采集关闭， 1 音频采集开启）
#define BRAC_USERSTATE_SPEAKVOLUME 3 ///< 用户当前说话音量（参数为 DOUBLE 类型（0.0 ~ 100.0））

#define BRAC_USERSTATE_RECORDING 4 ///< 用户录像（音）状态（参数为 DWORD 型）
#define BRAC_USERSTATE_LEVEL 5 ///< 用户级别（参数为 DWORD 型）
#define BRAC_USERSTATE_NICKNAME 6 ///< 用户昵称（参数为字符串 TCHAR 类型）
#define BRAC_USERSTATE_LOCALIP 7 ///< 用户本地 IP 地址（内网，参数为字符串 TCHAR 类型）
#define BRAC_USERSTATE_INTERNETIP 8 ///< 用户互联网 IP 地址(参数为字符串 TCHAR 类型）
#define BRAC_USERSTATE_VIDEOBITRATE 9 ///< 用户当前的视频码率（参数为 DWORD 类型，Bps）
#define BRAC_USERSTATE_AUDIOBITRATE 10 ///< 用户当前的音频码率（参数为 DWORD 类型，Bps）
#define BRAC_USERSTATE_P2PCONNECT 11 ///< 查询本地用户与指定用户的当前 P2P 连接状态（参数为 DWORD 类型，返回值：0 P2P 不通， 1 P2P 连接成功[TCP]，2 P2P 连接成功[UDP]，3 P2P 连接成功[TCP、UDP]）
#define BRAC_USERSTATE_NETWORKSTATUS 12 ///< 查询指定用户的网络状态(参数为 DWORD 类型，返回值：0 优良，1 较好，2 一般，3 较差，4 非常差)，注：查询间隔需要>1s
#define BRAC_USERSTATE_VIDEOSIZE 13 ///< 查询指定用户的视频分辨率（参数为 DWORD 类型，返回值：低 16 位表示宽度，高 16 位表示高度）
#define BRAC_USERSTATE_PACKLOSSRATE 14 ///< 查询指定用户的网络流媒体数据丢包率(参数为 DWORD 类型，返回值：0 - 100，如：返回值为 5，表示丢包率为 5%）
#define BRAC_USERSTATE_DEVICETYPE 15 ///< 查询指定用户的终端类型(参数为 DWORD 类型，返回值：0 Unknow，1 Windows，2 Android，3 iOS，4 Web，5 Linux，6 Mac，7 Win Phone，8 WinCE）
#define BRAC_USERSTATE_SELFUSERSTATUS 16 ///< 查询本地用户的当前状态(参数为 DWORD 类型，返回值：0 Unknow，1 Connected，2 Logined，3 In Room，4 Logouted，5 Link Closed）
#define BRAC_USERSTATE_SELFUSERID 17 ///< 查询本地用户的 ID(参数为 DWORD 类型，若用户登录成功，返回用户实际的 userid，否则返回-1）

3.5 视频呼叫事件定义

API：VideoCallControl 传入参数、VideoCallEvent 回调参数：

#define BRAC_VIDEOCALL_EVENT_REQUEST 1 ///< 呼叫请求
#define BRAC_VIDEOCALL_EVENT_REPLY 2 ///< 呼叫请求回复 J
#define BRAC_VIDEOCALL_EVENT_START 3 ///< 视频呼叫会话开始事件
#define BRAC_VIDEOCALL_EVENT_FINISH 4 ///< 挂断（结束）呼叫会话

3.6 业务对象常量定义

AnyChat SDK 业务队列功能新定义了业务队列所涉及到的业务对象及业务对象相关属性及方法、事件常量

3.6.1 对象类型定义

3.6.1.1 对象类型定义

#define ANYCHAT_OBJECT_TYPE_AREA	4	///< 服务区域
#define ANYCHAT_OBJECT_TYPE_QUEUE	5	///< 队列对象
#define ANYCHAT_OBJECT_TYPE_AGENT	6	///< 客服对象
#define ANYCHAT_OBJECT_TYPE_CLIENTUSER	8	///< 客户端用户对象，用于与服务器交换数据

3.6.1.2 通用标识定义

#define ANYCHAT_OBJECT_FLAGS_CLIENT	0	///< 普通客户
#define ANYCHAT_OBJECT_FLAGS_AGENT	2	///< 坐席用户
#define ANYCHAT_OBJECT_FLAGS_MANANGER	4	///< 管理用户
#define ANYCHAT_INVALID_OBJECT_ID	-1	///< 无效的对象 ID

3.6.2 对象属性定义

3.6.2.1 对象公共属性定义

#define ANYCHAT_OBJECT_INFO_FLAGS	7	///< 对象属性标志
#define ANYCHAT_OBJECT_INFO_NAME	8	///< 对象名称
#define ANYCHAT_OBJECT_INFO_PRIORITY	9	///< 对象优先级
#define ANYCHAT_OBJECT_INFO_ATTRIBUTE	10	///< 对象业务属性
#define ANYCHAT_OBJECT_INFO_DESCRIPTION	11	///< 对象描述
#define ANYCHAT_OBJECT_INFO_INTTAG	12	///< 对象标签，整型，上层应用自定义
#define ANYCHAT_OBJECT_INFO_STRINGTAG	13	///< 对象标签，字符串，上层应用自定义

3.6.2.2 服务区域属性定义

#define ANYCHAT_AREA_INFO_AGENTCOUNT	401	///< 服务区域客服用户数
#define ANYCHAT_AREA_INFO_GUESTCOUNT	402	///< 服务区域内访客的用户数（没有排入队列的用户）
#define ANYCHAT_AREA_INFO_QUEUEUSERCOUNT	403	///< 服务区域内排队的用户数
#define ANYCHAT_AREA_INFO_QUEUECOUNT	404	///< 服务区域内队列的数量

3.6.2.3 队列属性定义

#define ANYCHAT_QUEUE_INFO_MYSEQUENCENO 501 ///< 自己在该队列中的序号
#define ANYCHAT_QUEUE_INFO_BEFOREUSERNUM 502 ///< 排在自己前面的用户数
#define ANYCHAT_QUEUE_INFO_MYENTERQUEUETIME 503 ///< 进入队列的时间
#define ANYCHAT_QUEUE_INFO_LENGTH 504 ///< 队列长度（有多少人在排队），整型
#define ANYCHAT_QUEUE_INFO_WAITTIMESECOND 508 ///< 自己在队列中的等待时间（排队时长），单位：秒

3.6.2.4 客服属性定义

#define ANYCHAT_AGENT_INFO_SERVICESTATUS 601 ///< 服务状态，整型
#define ANYCHAT_AGENT_INFO_SERVICEUSERID 602 ///< 当前服务的用户 ID，整型
#define ANYCHAT_AGENT_INFO_SERVICEBEGINTIME 603 ///< 当前服务的开始时间，整型
#define ANYCHAT_AGENT_INFO_SERVICETOTALTIME 604 ///< 累计服务时间，整型，单位：秒
#define ANYCHAT_AGENT_INFO_SERVICETOTALNUM 605 ///< 累计服务的用户数，整型

3.6.2.5 客服服务状态定义

#define ANYCHAT_AGENT_STATUS_CLOSEED 0 ///< 关闭，不对外提供服务
#define ANYCHAT_AGENT_STATUS_WAITTING 1 ///< 等待中，可随时接受用户服务
#define ANYCHAT_AGENT_STATUS_WORKING 2 ///< 工作中，正在为用户服务
#define ANYCHAT_AGENT_STATUS_PAUSED 3 ///< 暂停服务

3.6.3 对象方法定义

3.6.3.1 对象公共参数控制方法

#define ANYCHAT_OBJECT_CTRL_CREATE 2 ///< 创建一个对象
#define ANYCHAT_OBJECT_CTRL_SYNCDATA 3 ///< 同步对象数据给指定用户，dwObjectId=-1，表示同步该类型的所有对象
#define ANYCHAT_OBJECT_CTRL_DEBUGOUTPUT 4 ///< 对象调试信息输出

3.6.3.2 服务区域控制方法

#define ANYCHAT_AREA_CTRL_USERENTER 401 ///< 进入服务区域
#define ANYCHAT_AREA_CTRL_USERLEAVE 402 ///< 离开服务区域

3.6.3.3 队列控制方法

```
#define ANYCHAT_QUEUE_CTRL_USERENTER    501  ///< 进入队列
#define ANYCHAT_QUEUE_CTRL_USERLEAVE    502  ///< 离开队列
```

3.6.3.4 客服控制方法

```
#define ANYCHAT_AGENT_CTRL_SERVICESTATUS    601  ///< 坐席服务状态控制（暂停服务、工作中、关闭）
#define ANYCHAT_AGENT_CTRL_SERVICEREQUEST   602  ///< 服务请求
#define ANYCHAT_AGENT_CTRL_FINISHSERVICE    604  ///< 结束服务
#define ANYCHAT_AGENT_CTRL_EVALUATION       605  ///< 服务评价，wParam 为客服 userid，lParam 为评分，lpStrValue 为留言
```

3.6.4 对象异步事件定义

3.6.4.1 对象公共事件定义

```
#define ANYCHAT_OBJECT_EVENT_UPDATE          1    ///< 对象数据更新
#define ANYCHAT_OBJECT_EVENT_SYNCDATAFINISH  2    ///< 对象数据同步结束
```

3.6.4.2 服务区域事件

```
#define ANYCHAT_AREA_EVENT_STATUSCHANGE     401  ///< 服务区域状态变化
#define ANYCHAT_AREA_EVENT_ENTERRESULT      402  ///< 进入服务区域结果
#define ANYCHAT_AREA_EVENT_USERENTER        403  ///< 用户进入服务区域
#define ANYCHAT_AREA_EVENT_USERLEAVE        404  ///< 用户离开服务区域
#define ANYCHAT_AREA_EVENT_LEAVERESULT      405  ///< 离开服务区域结果
```

3.6.4.3 队列事件

```
#define ANYCHAT_QUEUE_EVENT_STATUSCHANGE    501  ///< 队列状态变化
#define ANYCHAT_QUEUE_EVENT_ENTERRESULT     502  ///< 进入队列结果
#define ANYCHAT_QUEUE_EVENT_USERENTER       503  ///< 用户进入队列
#define ANYCHAT_QUEUE_EVENT_USERLEAVE       504  ///< 用户离开队列
#define ANYCHAT_QUEUE_EVENT_LEAVERESULT     505  ///< 离开队列结果
```

3.6.4.4 客服事件

```
#define ANYCHAT_AGENT_EVENT_STATUSCHANGE    601  ///< 坐席状态变化
#define ANYCHAT_AGENT_EVENT_SERVICENOTIFY   602  ///< 坐席服务通知（哪个用户到哪个客服办理业务）
#define ANYCHAT_AGENT_EVENT_WAITINGUSER     603  ///< 暂时没有客户，请等待
```

4 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。

4.1 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。

4.1.1 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
```

4.1.2 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。

4.1.3 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。

4.1.4 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。

4.1.5 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。

4.1.6 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的 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。

4.1.7 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的 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。

4.1.8 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。

4.2 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。

4.2.1 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
```

4.2.2 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。

4.2.3 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。

4.3 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。

4.3.1 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
```

4.3.2 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。

4.3.3 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。

4.3.4 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。

4.3.5 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。

4.3.6 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。

4.4 数据传输事件接口

4.4.1 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
```

4.4.2 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。

4.4.3 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。

4.4.4 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。

4.4.5 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。

4.5 文字消息事件接口

4.5.1 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
```

4.5.2 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。

4.6 视频呼叫事件接口

4.6.1 接口定义

```
@protocol AnyChatVideoCallDelegate <NSObject>
// 视频呼叫事件
- (void) OnAnyChatVideoCallEventCallBack:(int) dwEventType : (int) dwUserId : (int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr;
@end
```

4.6.2 视频呼叫事件

- (void) OnAnyChatVideoCallEventCallBack:(int) dwEventType : (int) dwUserId : (int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr

参数：

dwEventType	呼叫事件类型，详见函数 BRAC_VideoCallControl 中的定义
dwUserId：	视频呼叫事件发起方用户 ID
dwErrorCode：	错误代码，当事件类型为“Reply”和“Finish”时有效
dwFlags：	视频呼叫标志
dwParam：	事件附带参数（整型）
lpUserStr：	事件附带参数（字符串）
lpUserValue：	用户自定义参数，在设置回调函数时传入

说明：

当其它用户通过 API：BRAC_VideoCallControl 发起视频呼叫时，将触发该回调函数。

用户 A 向用户 B 发送（Request）请求，用户 B 回复（Reply）同意通话之后，服务器会自动向 A、B 同时发送（Start）指令，表示会话开始，当客户端在回调函数中收到 dwEventType= BRAC_VIDEOCALL_EVENT_START 事件时，dwParam 表示 RoomId，由服务器自动分配，这时用户 A、B 均需要主动进入分配的房间，打开本地音频、视频，同时请求对方的音频、视频才能完成整个视频呼叫过程。

更多关于视频呼叫事件的信息可参考技术论坛相关内容：http://bbs.anychat.cn/forum.php?mod=viewthread&tid=150&extra=page%3D1

4.7 录像、拍照事件接口

4.7.1 接口定义

```
@protocol AnyChatRecordSnapShotDelegate <NSObject>
// 录像完成事件
- (void) OnAnyChatRecordCallBack:(int) dwUserid : (NSString*) lpFileName : (int) dwElapse : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr;
// 拍照完成事件
- (void) OnAnyChatSnapShotCallBack:(int) dwUserid : (NSString*) lpFileName : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr;
@end
```

4.7.2 视频录制完成事件

- (void) OnAnyChatRecordCallBack:(int) dwUserid : (NSString*) lpFileName : (int) dwElapse : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr;

参数：

dwUserId： 被录制用户 ID

lpFileName： 文件保存路径

dwElapse： 录像时长，单位：秒

dwFlags： 录像标志

dwParam： 用户自定义参数，整型

lpUserStr： 用户自定义参数，字符串类型

备注：

客户端调用 API：StreamRecordCtrlEx 录像完成之后，将会触发该回调事件，其中 lpFileName 为本地文件路径；

用户自定义参数包括整型（dwParam）、字符串类型（lpUserStr）与 API：StreamRecordCtrlEx 的传入参数对应；

进行中心服务器录像时，也可以触发客户端的本地回调函数，更多信息可参考：中心服务器录像支持触发客户端回调事件、AnyChat音视频录制整体解决方案

4.7.3 拍照完成事件

- (void) OnAnyChatSnapShotCallBack:(int) dwUserId : (NSString*) lpFileName : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr;

参数：

dwUserId： 被拍照用户 ID

lpFileName： 文件保存路径

dwFlags： 拍照标志

dwParam： 用户自定义参数，整型

lpUserStr： 用户自定义参数，字符串类型

备注：

客户端调用 API：SnapShot 完成图像抓拍之后，将会触发该回调事件，其中 lpFileName 为本地文件路径。

4.8 业务对象事件接口

4.8.1 接口定义

```
@protocol AnyChatObjectEventDelegate <NSObject>
//业务对象完成事件
- (void) OnAnyChatObjectEventCallBack: (int) dwObjectType : (int) dwObjectId : (int) dwEventType : (int) dwParam1 : (int) dwParam2 : (int) dwParam3 : (int) dwParam4 : (NSString*) lpStrParam;
@end
```

4.8.2 业务对象完成事件

- (void) OnAnyChatObjectEventCallBack: (int) dwObjectType : (int) dwObjectId : (int) dwEventType : (int) dwParam1 : (int) dwParam2 : (int) dwParam3 : (int) dwParam4 : (NSString*) lpStrParam;

参数：

dwObjectType：	业务对象类型，见 3.6.1.1 章节描述内容；
dwObjectId：	业务对象 ID
dwEventType：	业务对象回调事件类型，见 3.6.4 章节描述内容；
dwParam1：	整型参数值一（需要和业务事件类型参数值匹配）
dwParam2：	整型参数值二（需要和业务事件类型参数值匹配）
dwParam3：	整型参数值三（需要和业务事件类型参数值匹配）
dwParam4：	整型参数值四（需要和业务事件类型参数值匹配）
lpStrParam：	字符串参数值（需要和业务事件类型参数值匹配）

备注：

客户端调用 API：调用 AnyChatPlatform 的 ObjectControl 方法，将会触发一次或多次该回调事件。

5 函数说明

5.1 初始化与资源释放

5.1.1 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。

5.1.2 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。

5.2 登录流程

5.2.1 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。

5.2.2 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。

5.2.3 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》。

5.2.4 视频呼叫控制

+ (int) VideoCallControl: (int) dwEventType : (int) dwUserId : (int) dwErrorCode : (int) dwFlags : (int) dwParam : (NSString*) lpUserStr

功能：对视频呼叫业务流程进行控制，发起视频呼叫，或是结束视频通话等。

返回值：0 表示成功，否则为出错代码

参数：

dwEventType 事件类型，定义为：

```
#define BRAC_VIDEOCALL_EVENT_REQUEST 1    ///< 呼叫请求
#define BRAC_VIDEOCALL_EVENT_REPLY   2    ///< 呼叫请求回复
#define BRAC_VIDEOCALL_EVENT_START   3    ///< 视频呼叫会话开始事件
#define BRAC_VIDEOCALL_EVENT_FINISH  4    ///< 挂断（结束）呼叫会话
```

dwUserId 目标用户 ID

dwErrorCode 出错代码，与事件类型相关，默认为 0

dwFlags 视频呼叫标志，默认为 0

dwParam 用户自定义参数（整型），默认为 0

lpUserStr 用户自定义参数（字符串），默认为空

备注：

视频呼叫业务逻辑主要实现两个终端（PC、手机、Pad 等）之间的通话请求流程控制，包括请求（Request）、回复（Reply）、开始（Start）以及结束（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。为 V4.9 版本新增接口。

该 API 接口和回调函数（BRAC_VideoCallEvent_CallBack）中的 dwUserId 均为对方（被呼叫方）的用户 ID；

被呼叫方拒绝通话时，可通过发送发送回复（Reply）指令，其中 dwErrorCode=100104 来拒绝；

被呼叫方同意通话时，发送回复（Reply）指令，其中 dwErrorCode=0，然后服务器会向双方发送通话开始（Start）指令；

用户 A 向用户 B 发送（Request）请求，用户 B 回复（Reply）同意通话之后，服务器会自动向 A、B 同时发送（Start）指令，表示会话开始，当客户端在回调函数中收到 dwEventType= BRAC_VIDEOCALL_EVENT_START 事件时，dwParam 表示 RoomId，由服务器自动分配，这时用户 A、B 均需要主动进入分配的房间，打开本地音频、视频，同时请求对方的音频、视频才能完成整个视频呼叫过程。

结束通话时，任何一方（包括业务服务器）均可以发送结束（Finish）指令，

然后服务器会向双方发送通话结束（Finish）指令；

业务服务器可干预呼叫流程：在业务服务器端回调函数（BRAS_OnVideoCallEvent_CallBack）收到呼叫请求指令后，返回 0 表示允许呼叫，否则为出错代码，不允许呼叫；在会话过程中业务服务器可以发送结束（Finish）指令，强制挂断指定用户的通话；

API 接口中的 dwParam（整型）、lpUserStr（字符串）均为用户自定义用途；

一个用户同时只能发起一路呼叫请求，也同时只能被一个用户呼叫；

更多关于视频呼叫事件的信息可参考技术论坛相关内容：http://bbs.anychat.cn/forum.php?mod=viewthread&tid=150&extra=page%3D1

5.2.5 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_PrepareEnterRoom，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_PrepareEnterRoom_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》。

5.2.6 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。

5.2.7 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。

5.2.8 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。

5.3 音视频操作

5.3.1 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。

5.3.2 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。

5.3.3 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。

5.4 查询状态

5.4.1 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。

5.4.2 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。

5.4.3 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。

5.4.4 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。

5.4.5 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。

5.4.6 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。

5.4.7 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。

5.4.8 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。

5.4.9 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。

5.5 普通功能

5.5.1 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

5.5.2 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。

5.5.3 传送文本消息

+ (int) SendTextMessage: (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。

5.5.4 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，表示向该房间的其它用户广播数据（注：自己不能发送给自己）。

5.5.5 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；

5.5.6 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。

5.5.7 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
infolen	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

5.5.8 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。

5.5.9 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。

5.5.10 音视频录制

+(int) StreamRecordCtrlEx:(int)dwUserId : (BOOL)bStartRecord : (int)dwFlags : (int) dwParam : (NSString*) szUserStr;

功能：对指定用户的音视频流进行录制。

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。

szUserStr： 录制指令附带参数（字符串类型），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。

视频录制可以在本地进行，也可以在服务器端进行，通过 dwFlags 标志进行控制，视频录制完成之后， 将触发回调事件：OnAnyChatRecordCallBack。

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：

```
ANYCHAT_RECORD_FLAGS_VIDEO      ///< 录制视频
ANYCHAT_RECORD_FLAGS_AUDIO      ///< 录制音频
ANYCHAT_RECORD_FLAGS_SERVER  ///< 服务器端录制
ANYCHAT_RECORD_FLAGS_MIXAUDIO   ///< 录制音频时，将其它人的声音混音后
```

录制

ANYCHAT_RECORD_FLAGS_MIXVIDEO ///< 录制视频时，将其它人的视频迭加后录制

ANYCHAT_RECORD_FLAGS_ABREAST ///< 录制视频时，将其它人的视频并列录制

ANYCHAT_RECORD_FLAGS_STEREO ///< 录制音频时，将其它人的声音混合为立体声后录制

ANYCHAT_RECORD_FLAGS_SNAPSHOT ///< 拍照

ANYCHAT_RECORD_FLAGS_LOCALCB ///< 触发本地回调

ANYCHAT_RECORD_FLAGS_STREAM ///< 视频流录制

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： Windows平台中心录像服务器部署、Linux平台中心录像服务器部署

更多信息可参考：

AnyChat音视频录制整体解决方案

AnyChat支持录像文件格式设置（MP4、WMV、FLV、MP3）

中心录像服务器返回录像文件路径可配置

中心服务器录像支持触发客户端回调事件

5.5.11 图像抓拍

+(int) SnapShot:(int)dwUserId : (int)dwFlags : (int) dwParam;

功能：对指定用户的视频进行抓拍。

返回值：0 表示抓拍指令被 SDK 成功接收，否则为出错代码

参数：

dwUserId： 需要抓拍视频的用户编号，可用-1 表示本地用户（自己）；

dwFlags： 功能标志；

dwParam： 抓拍指令附带参数（整形），抓拍图像成功之后，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 下达图像抓拍任务，视频抓拍完成之后， 将触发回调事件：OnAnyChatSnapShotCallBack。

5.6 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。

5.6.1 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。

5.6.2 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。

5.6.3 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。

5.6.4 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。

5.6.5 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。

5.6.6 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 dInfoId 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。

5.6.7 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。

5.7 系统设置

5.7.1 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

5.7.2 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。

5.7.3 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。

5.7.4 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

5.7.5 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。

5.7.6 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。

5.7.7 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。

5.7.8 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。

5.7.9 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。

5.7.10 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。

5.7.11 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。

5.7.12 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。

5.7.13 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。

5.8 业务排队

自 AnyChat for iOS SDK V2.3 版本开始，AnyChat 提供了业务排队功能，抽象出了业务排队应用场景中需要的营业厅、队列、坐席、客户等业务对象，通过调用提供的客户端 API 来操作这些对象的属性、方法及事件，如：进出营业厅、进出队列方法；获取排队人数、在队列中所排位置属性；进出队列、坐席服务响应事件等。通过响应不同的业务对象事件来实现排队业务逻辑功能，开发人员只需关注业务逻辑的实现，AnyChat 会自动的维护业务对象的数据变化。

具体可以参考：AnyChat提供业务排队整体解决方案。

5.8.1 获取对象 ID 列表

+ (NSMutableArray*) ObjectGetIdList: (int) dwObjectType;

功能：获取业务对象的 Id 数组

返回值：id 数组

参数：

dwObjectType 业务对象类型，见 3.6.1.1 章节描述内容；

备注

可以通过该方法查询业务对象的整形 id 数组，如服务区域的业务队列的 id 数组。

5.8.2 获取对象属性值

+ (int) ObjectGetIntValue: (int) dwObjectType : (int) dwObjectId : (int) dwInfoName;

功能：获取对象属性值（整型）

返回值：业务对象属性值

参数：

dwObjectType 业务对象类型，见 3.6.1.1 章节描述内容；

dwObjectId 业务对象的 Id

dwInfoName 要查询的业务对象的属性名，各类业务对象的属性定义见 3.6.2 章节描述内容；

备注

可以通过该方法查询业务对象的整形属性值，如队列人数，排队时间等。

+ (NSString*) ObjectGetStringValue: (int) dwObjectType : (int) dwObjectId : (int) dwInfoName;

功能：获取对象属性值（字符串）

返回值：业务对象属性值

参数：

dwObjectType 业务对象类型，见 3.6.1.1 章节描述内容；

dwObjectId 业务对象的 Id

dwInfoName 要查询的业务对象的属性名，各类业务对象的属性定义见 3.6.2 章节描述内容；

备注

可以通过该方法查询业务对象的字符串属性，如服务区域名称，队列名称等。

5.8.3 设置对象属性值

+ (int) ObjectSetIntValue: (int) dwObjectType : (int) dwObjectId : (int) dwInfoName : (int) dwValue;

功能：设置对象属性值（整形）

返回值：0 表示成功，否则为错误代码

参数：

dwObjectType 业务对象类型，见 3.6.1.1 章节描述内容；

dwObjectId 业务对象的 Id

dwInfoName 要设置的业务对象的属性名，各类业务对象的属性定义

见 3.6.2 章节描述内容；

dwValue 要设置的业务对象的属性值

备注

可以通过该方法来设置业务对象的属性值，如设置用户对象的优先级。

+ (int) ObjectSetStringValue: (int) dwObjectType : (int) dwObjectId : (int) dwInfoName : (NSString*) lpStrValue;

功能：设置对象属性值（字符串）

返回值：0 表示成功，否则为错误代码

参数：

dwObjectType 业务对象类型，见 3.6.1.1 章节描述内容；

dwObjectId 业务对象的 Id

dwInfoName 要设置的业务对象的属性名，各类业务对象的属性定义见 3.6.2 章节描述内容；

lpStrValue 要设置的业务对象的属性值

备注

可以通过该方法来设置业务对象的属性值，如设置服务区域的对象名称。

5.8.4 业务对象参数控制

+ (int) ObjectControl: (int) dwObjectType : (int) dwObjectId : (int) dwCtrlCode : (int) dwParam1 : (int) dwParam2 : (int) dwParam3 : (int) dwParam4 : (NSString*) lpStrValue;

功能：对业务对象进行参数控制

返回值：0 表示成功，否则表示错误代码

参数：

dwObjectType 业务对象类型，见 3.6.1.1 章节描述内容；

dwObjectId 业务对象的 Id

dwCtrlCode	对象公共参数定义，见 3.6.3 章节描述内容；
dwParam1	默认为 0
dwParam2	默认为 0
dwParam3	默认为 0
dwParam4	默认为 0
lpStrValue	默认为空

备注

可以通过该方法对业务对象进行参数控制，如进/出服务区域，进/出队列，开始/结束服务等。

6 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；

2014-06-01 V2.0

基于 AnyChat Platform Core SDK V5.0 版本编译；
增加数据加密、解密 API 接口，可实现上层应用对语音、视频等数据的加解密；
AnyChat Server SDK 支持 64bit Java 环境

增加视频方向手工修正 API 接口，包括本地视频采集方向和远程视频显示方向
修正文件传输时在某些网络环境下文件内容接收不完整的问题；
修正 iOS 设备在某些分辨率下视频采集显示花屏的问题
修正 iOS 平台部分设备前置摄像头方向不准确的问题；
修正 iOS 音频在关闭后，需要退出房间才能重新打开的问题；
iOS 平台支持显示远程视频时视频方向不随设备的旋转而改变；

2014-09-01 V2.1

优化并更新内核 Codec 库，整体性能有较大幅度提升
针对网络状况较差时网络连接的稳定性进行优化
AnyChat 支持录像文件格式设置（MP4、WMV、FLV、MP3）
实现服务器集中收集客户端日志信息功能
修正 iOS 设备平放（FACEUP）时视频采集方向不准确的问题
修正 Android 平台特定场景下会导致应用闪退的问题
修正低性能设备上视频通话时延迟累计增大的问题

2015-01-14 V2.2

优化音视频内核，编解码效率更高；
增加服务器合成录制、合成流录制功能；
视频录制过程中支持动态改变视频分辨率；
优化网络抖动，减少网络连接断开，提高连接速度；
支持通过 API 接口开启 AnyChat 内核调试模式；
支持 iOS 全系列架构（i386、x86_64、armv7、armv7s、arm64），兼容 iOS 8；
修正 iOS 设备外接耳机之后声音继续从喇叭播放的问题；
修正 iOS 写入中文文件名为乱码的问题；
iOS 平台增加本地视频录制，拍照功能；

2015-07-15 V2.3

优化移动设备上服务器合成流录制，功能更稳定；
针对 iOS 8 进行适配，兼容 iPhone 6、iPhone 6 Plus；
增加业务对象 API，支持智能排队等新功能；
支持视频数据回调，可用于活体检测、人脸识别等业务场景；
优化网络数据传输，提高移动网络抖动时的音视频质量；

7 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_ALREADIN        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。