目录

云屋视频SDK参考

> Active X插件方式，适合各种Windows客户端开发语言集成。

此文档适用v3.8系列版本的SDK

说明

> SDK的安装、开发配置和组件介绍。

SDK安装

SDK开发包内容说明：

```
|——readme.txt          sdk包说明文件
|——doc                 开发手册、版本历史文件目录
|——bin                 sdk及依赖的文件目录
|——setup               打包bin目录下的文件形成的安装包
|——examples            演示程序
```

运行setup/CloudroomSDK.exe，如果提示要求管理员权限，请选择[允许]。

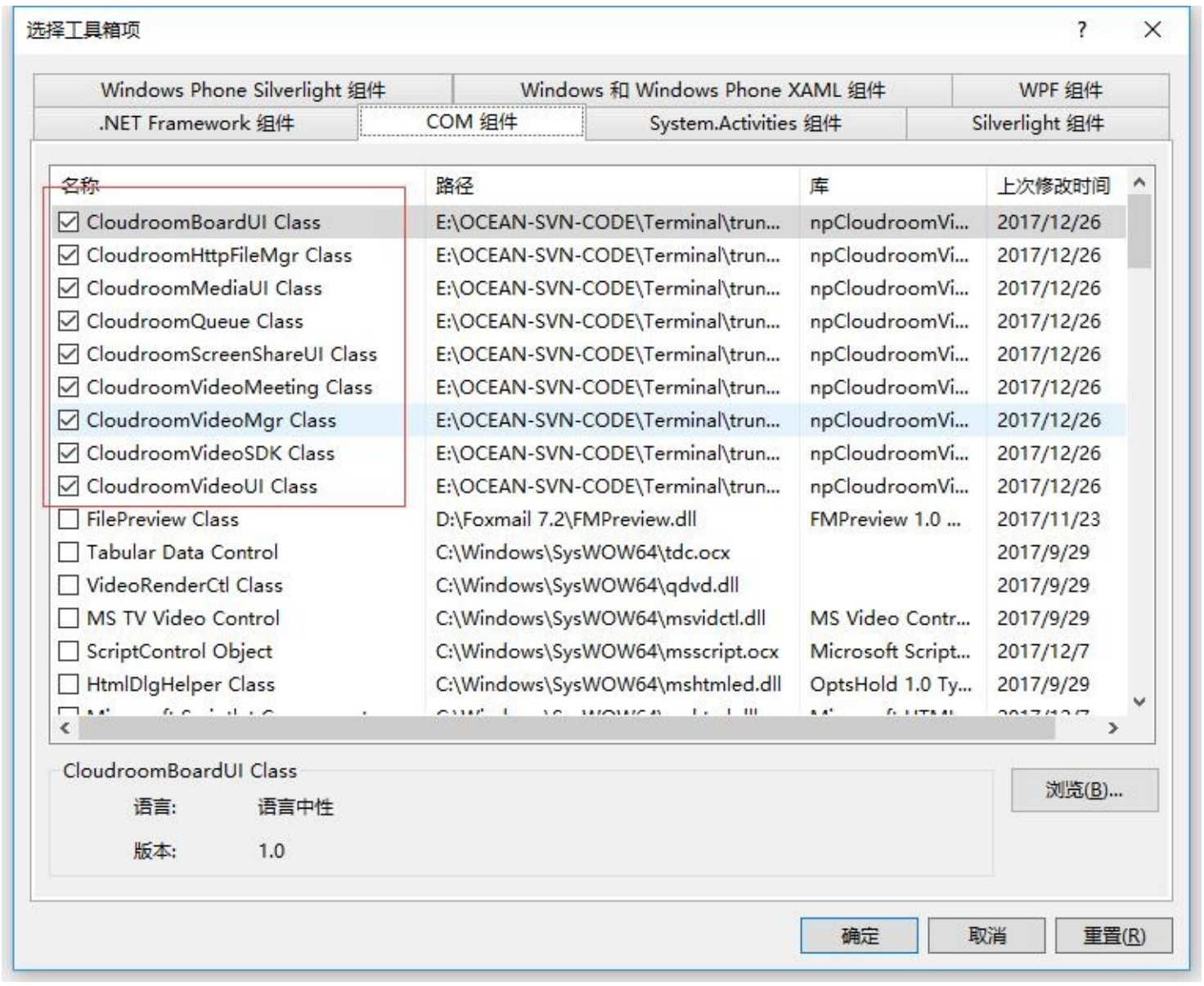
运行环境要求

> - Windows XP (SP2)及以后版本的系统

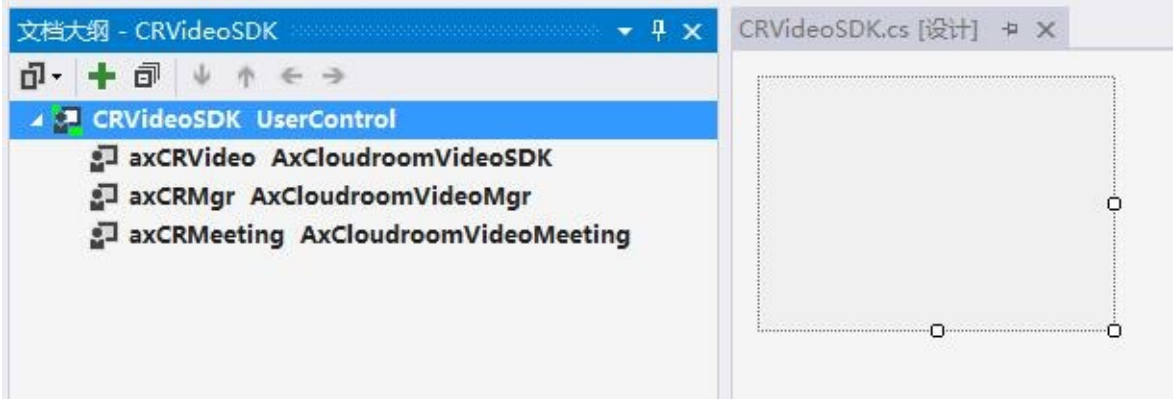
开发环境配置

适合各种windows桌面客户端开发平台, 如 VC, C#, VB, Qt, C++ Builder, Delphi……

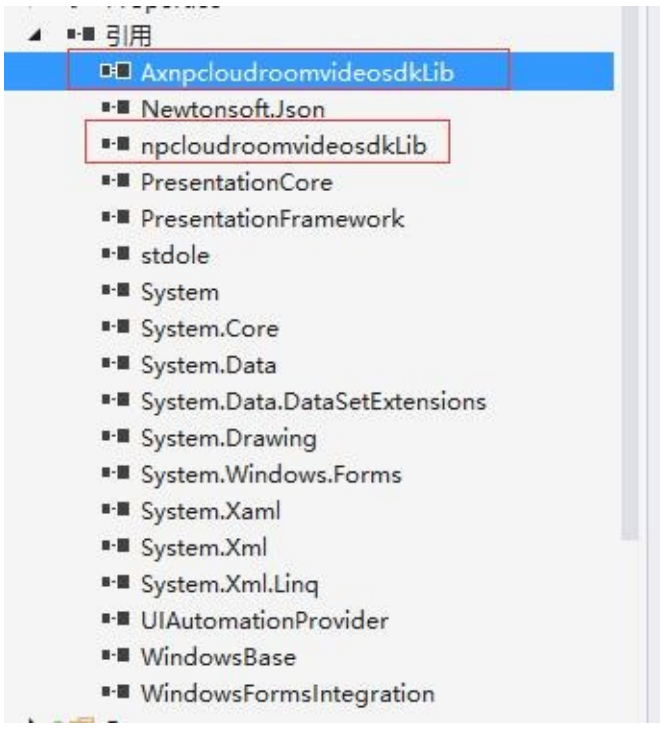
> 以Visual Studio 2008 C# 开发为例
>
> - 添加组件到Visual Studio工具箱



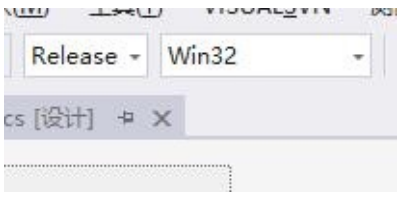
- 拖拽组件到自己项目内的窗体上



> 打开[引用]，检查组件是否引入成功（如下图）



- 选择解决方案配置[Release]、平台[Win32] 或 [x86]



- 命名引入工程内的SDK组件对象，并访问接口和回调函数

```
//crvideosdk.cs
using Axnpcloudroomvideosdk Lib;
namespace Meeting_WPF
{
    public partial class CRVideoSDK : UserControl
    {
        public AxCloudroomVideoSDK VideoSDK
        {
            get { return axCRVideo; }
        }
        public AxCloudroomVideoMeeting Meeting
        {
            get { return axCRMeeting; }
        }
        public AxCloudroomVideoMgr Mgr
        {
            get { return axCRMgr; }
        }
    }
}

//login.cs
using Axnpcloudroomvideosdk Lib;
namespace Meeting_WPF
{
    public partial class Login : Window
    {
        public CRVideoSDK CRVideo = new CRVideoSDK();
```

```
        private void initMeeting()
        {
            CRVideo.VideoSDK.init(Environment.CurrentDirectory);
        }
        private void login()
        {
          CRVideo.Mgr.login("demo@cloudroom.com", "e10adc3949ba59abbe56e057f20f883e",
"Tom", "user000001", "", "");
        }
        private void initDelegate()
        {
            CRVideo.Mgr.loginSuccess += new AxnpcloudroomvideosdkLib.ICloudroomVideoM
grEvents_loginSuccessEventHandler(loginSuccess);
            CRVideo.Mgr.loginFail += new AxnpcloudroomvideosdkLib.ICloudroomVideoMgrE
vents_loginFailEventHandler(loginFailed);
        }
    }
}
```

主要组件

> SDK是由众多的DLL组件组合而成的，对外提供以下几个组件：

- 基础组件 CloudroomVideoSDK
- 管理组件 CloudroomVideoMgr
- 队列组件 CloudroomQueue
- Http文件管理组件 CloudroomHttpFileMgr
- 视频会议组件 CloudroomVideoMeeting
    - 视频显示组件 CloudroomVideoUI
    - 屏幕共享画面显示组件 CloudroomScreenShareUI
    - 影音显示组件 CloudroomMediaUI
    - 白板显示组件 CloudroomBoardUI

基础组件CloudroomVideoSDK

```
CLSID: {07EFD662-A1BB-4d8d-9BEE-F7E43E5FEBF5}
ProgID: npCloudroomVideoSDK.CloudroomVideoSDK
MIME TYPE: application/x-cloudroom-videosdk
```

CloudroomVideoSDK是基础组件，是整个SDK使用的基础。

该组件一个进程内只能创建一个实例，直到应用退出时才反初始化并销毁。

组件使用过程主要包括：

```
1. 创建组件实例
2. 执行初始化
3. 程序退出时执行反初始化
```

管理组件CloudroomVideoMgr

```
CLSID: {120AD2B0-68F2-46c6-88D8-52173F501C0F}
ProgID: npCloudroomVideoSDK.CloudroomVideoMgr
MIME TYPE: application/x-cloudroom-videomgr
```

CloudroomVideoMgr是登录、呼叫、会议创建管理和透明传输类。

该组件一个进程内只能创建一个实例，实现了入会前的相关功能。

组件使用过程主要包括：

```
1. 创建组件实例
2. 登录
3. 创建会议
```

注意: 只有在CloudroomVideoSDK Init初始化成功后接口才可用。

队列组件CloudroomQueue

```
CLSID: {9AAD199D-A02F-4513-875D-AA81091E44B9}
ProgID: npCloudroomVideoSDK.CloudroomQueue
MIME TYPE: application/x-cloudroom-queue
```

CloudroomQueue是队列组件，它实现队列功能。

该组件一个进程内只能创建一个实例，是可选组件，用于用户分发，您可以使用它，也可以自已另外实现，这并不影响视频呼叫、音视频通话功能。

组件使用过程主要包括：

```
1. 创建组件实例，执行初始化
2. 队列获取，客户排队/座席服务
```

注意：只有在CloudroomVideoMgr登录成功后接口才可用。

Http文件管理组件CloudroomHttpFileMgr

```
CLSID: {7E44F8C9-7C8D-4004-8F45-D9819D78663C}
ProgID: npCloudroomVideoSDK.CloudroomHttpFileMgr
MIME: application/x-cloudroom-httpfilemgr
```

CloudroomHttpFileMgr是Http文件上传下载及文件管理类。

该组件一个进程内只能创建一个实例，主要应用于单方文件归档，单方文件下载，支持非云屋http服务器对接。

如果会议内临时文件共享，请使用CloudroomVideoMeeting中的会议网盘功能。

下载支持断点续传，上传暂不支持断点机制。

注意：只有在CloudroomVideoSDK初始化后接口才可用。

视频会议组件CloudroomVideoMeeting

```
CLSID: {9E9DD983-A9F8-4dff-B694-B1AE1C708B1E}
ProgID: npCloudroomVideoSDK.CloudroomVideoMeeting
MIME TYPE: application/x-cloudroom-videomeeting
```

CloudroomVideoMeeting是视频会议类。

该组件一个进程内只能创建一个实例，包含了视频会话相关的全部功能。

组件使用过程主要包括：

```
1. 创建组件实例
2. 进入会议
3. 会议内的各功能处理
4. 退出会议
```

注意：只有在CloudroomVideoSDK Init初始化成功后接口才可用。

进入视频会议可用以下组件：

视频显示组件**CloudroomVideoUI**

```
CLSID: {8A6BBBDC-C6BE-4a47-92F3-F9581C3FB95E}
ProgID: npCloudroomVideoSDK.CloudroomVideoUI
MIME: application/x-cloudroom-videoui
```

CloudroomVideoUI是视频显示组件，它显示设定的用户的视频。

该组件可以创建多个实例，然后分别配置大小、位置并设置要显示的用户ID和摄像头ID即可。

注意：只有在CloudroomVideoMeeting入会成功后才能正常工作。

屏幕共享画面显示组件**CloudroomScreenShareUI**

```
CLSID: {6FF142C5-8A36-49d7-B627-D60B803550FC}
ProgID: npCloudroomVideoSDK.CloudroomScreenShareUI
MIME: application/x-cloudroom-screenshareui
```

CloudroomScreenShareUI是屏幕共享显示组件，它用于显示会议内对方共享的屏幕图像。

该组件一个进程内只能创建一个实例，整个程序只能创建一个CloudroomScreenShareUI对象，用来接受显示他人开启共享后传过来的画面, 开启共享的接口是CloudroomVideoMeeting中的接口startScreenShare。

注意：只有在CloudroomVideoMeeting入会成功后才能正常工作。

影音显示组件**CloudroomMediaUI**

```
CLSID: {93A618D5-2535-42d0-B72B-95705263F398}
ProgID: npCloudroomVideoSDK.CloudroomMediaUI
MIME: application/x-cloudroom-mediaui
```

CloudroomMediaUI是影音显示组件，它用于显示自己或者对方会议内播放的影音图像和声音。

该组件一个进程内只能创建一个实例，整个程序只能创建一个CloudroomMediaUI对象，影音控制接口由CloudroomVideoMeeting统一提供。

注意：只有在CloudroomVideoMeeting入会成功后才能正常工作。

白板显示组件**CloudroomBoardUI**

```
CLSID: {D9ED4651-4461-458a-99F4-A455977582FF}
ProgID: npCloudroomVideoSDK.CloudroomBoardUI
MIME: application/x-cloudroom-boardui
```

CloudroomBoardUI是白板显示组件，用于显示会议内自己或他人创建的白板。

该组件一个进程内只能创建一个实例，整个程序只能创建一个CloudroomBoardUI对象，也可以不使用此组件，然后用CloudroomVideoMeeting内白板相关的接口实现个性化的白板功能。

回到索引

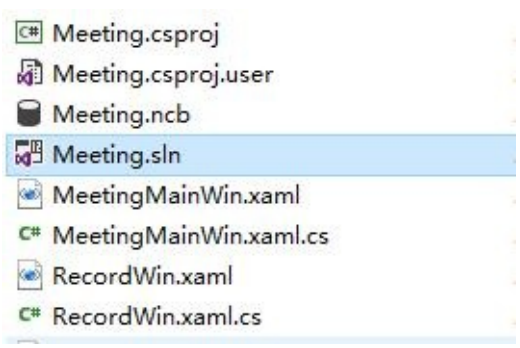
Demo程序介绍

> Examples 目录下Demo程序配置和功能简介。

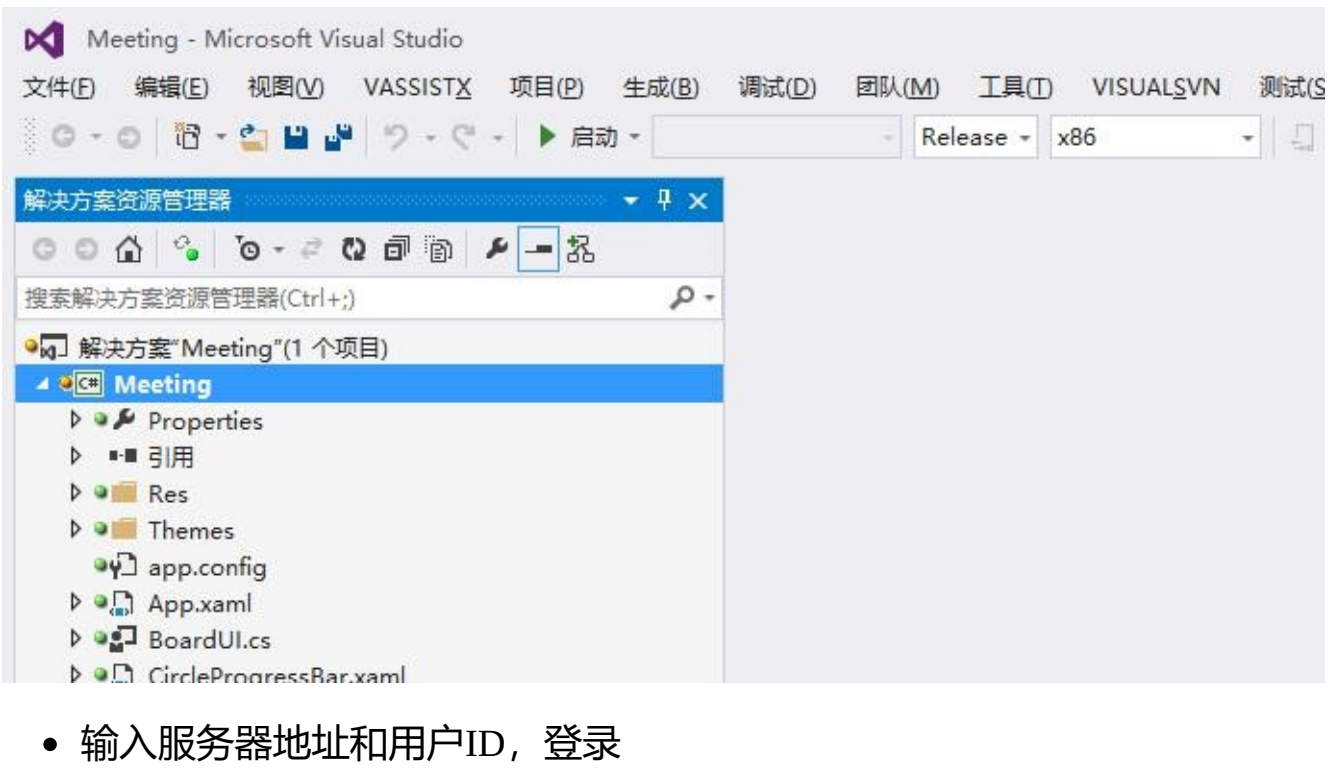
视频会议

> C# WPF编写，主要实现了透明传输，创建进入视频会议，用会议号进入视频会议，会议内视频墙、屏幕共享、电子白板、音视频设置、IM聊天、文件共享等功能。

- 安装SDK，然后用Visual Studio 2008(或更高版本)打开SDK开发包内项目文件



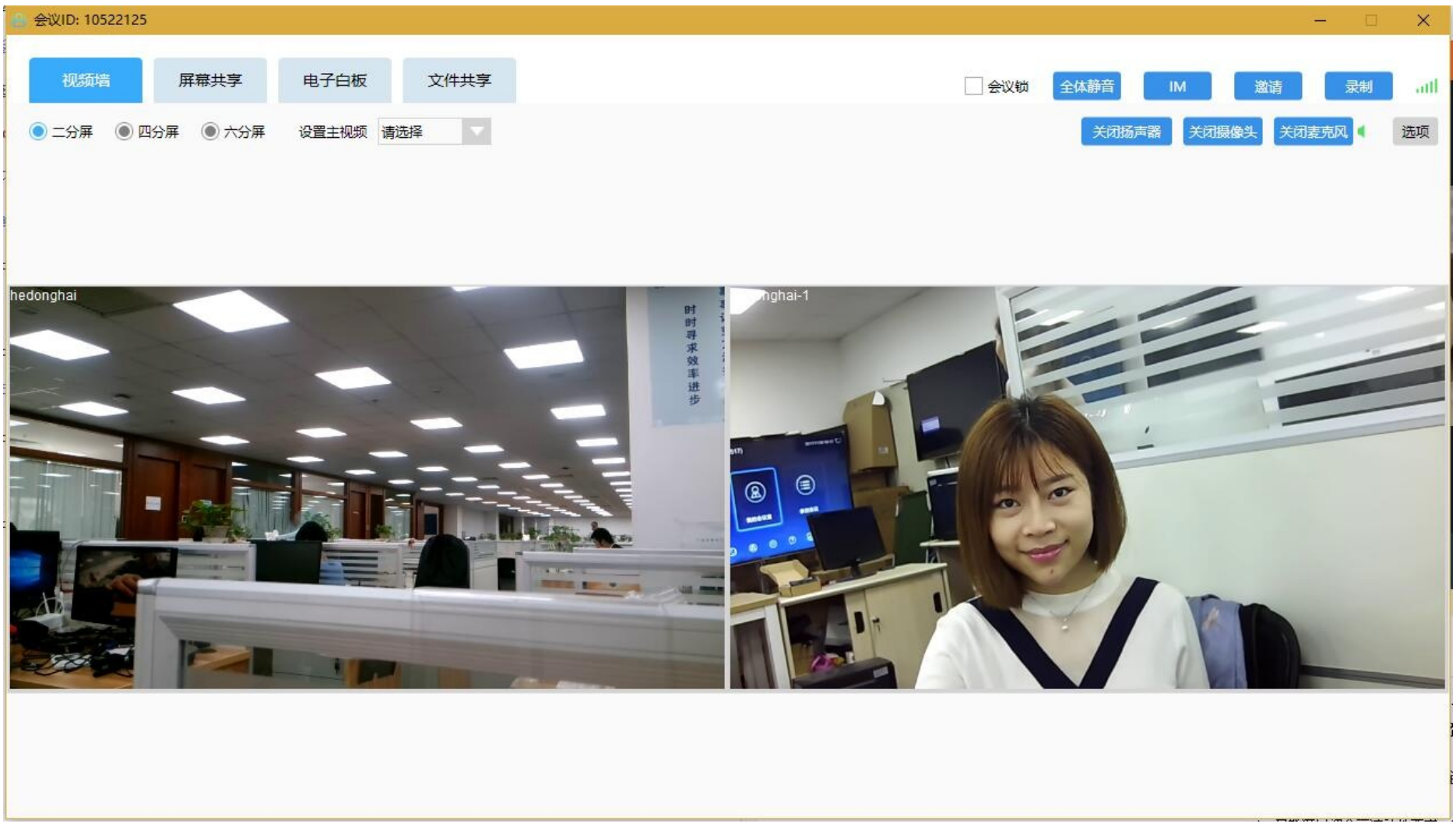
- 选择解决方案配置、平台



- 输入服务器地址和用户ID，登录

- 输入会议主题，创建并进入会议

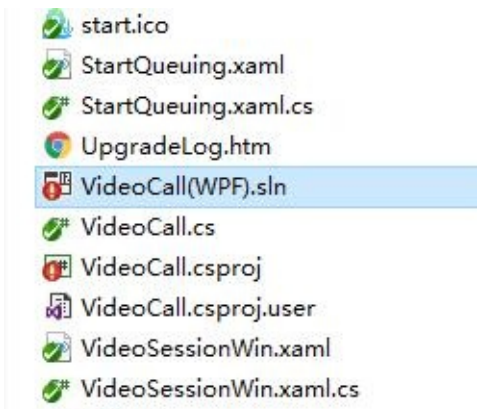
- 进入视频会议主界面



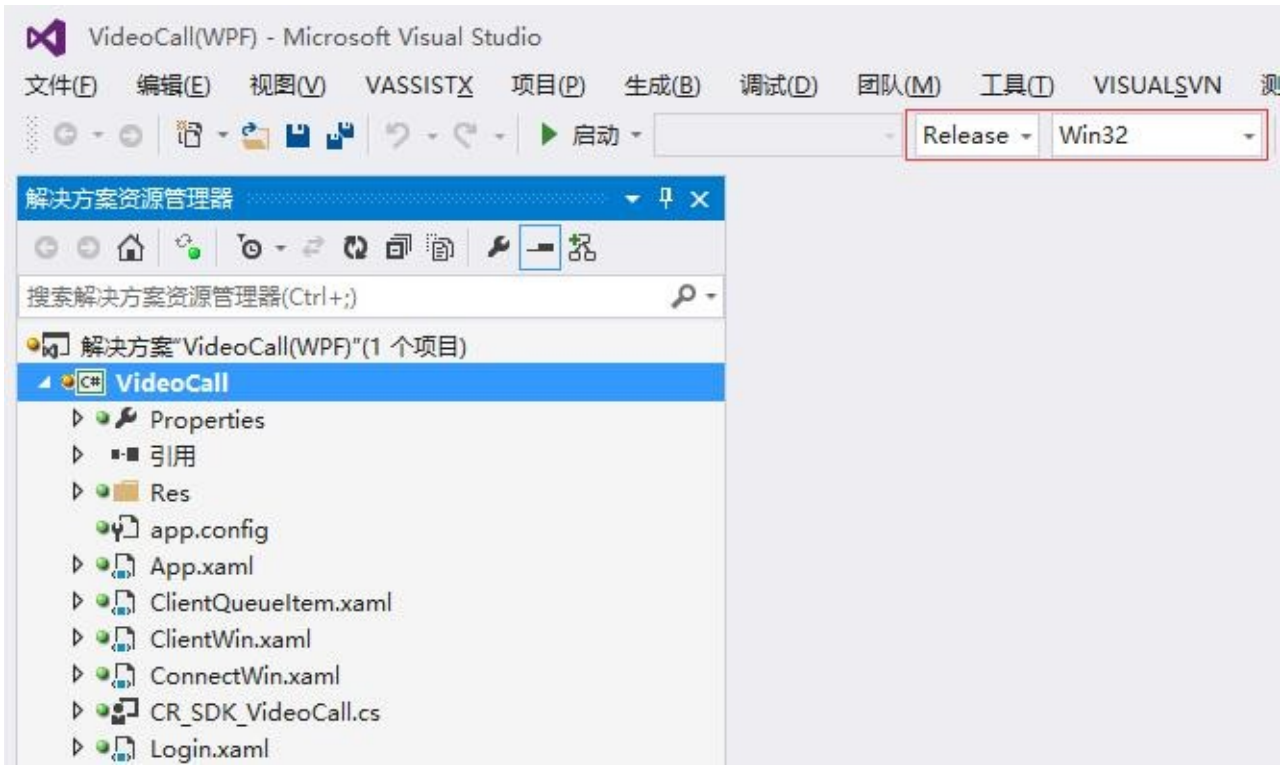
队列呼叫

> C# WPF编写，主要实现了利用队列系统进行用户分配，然后呼叫分到的用户，双方进入同一个视频会话中等功能。

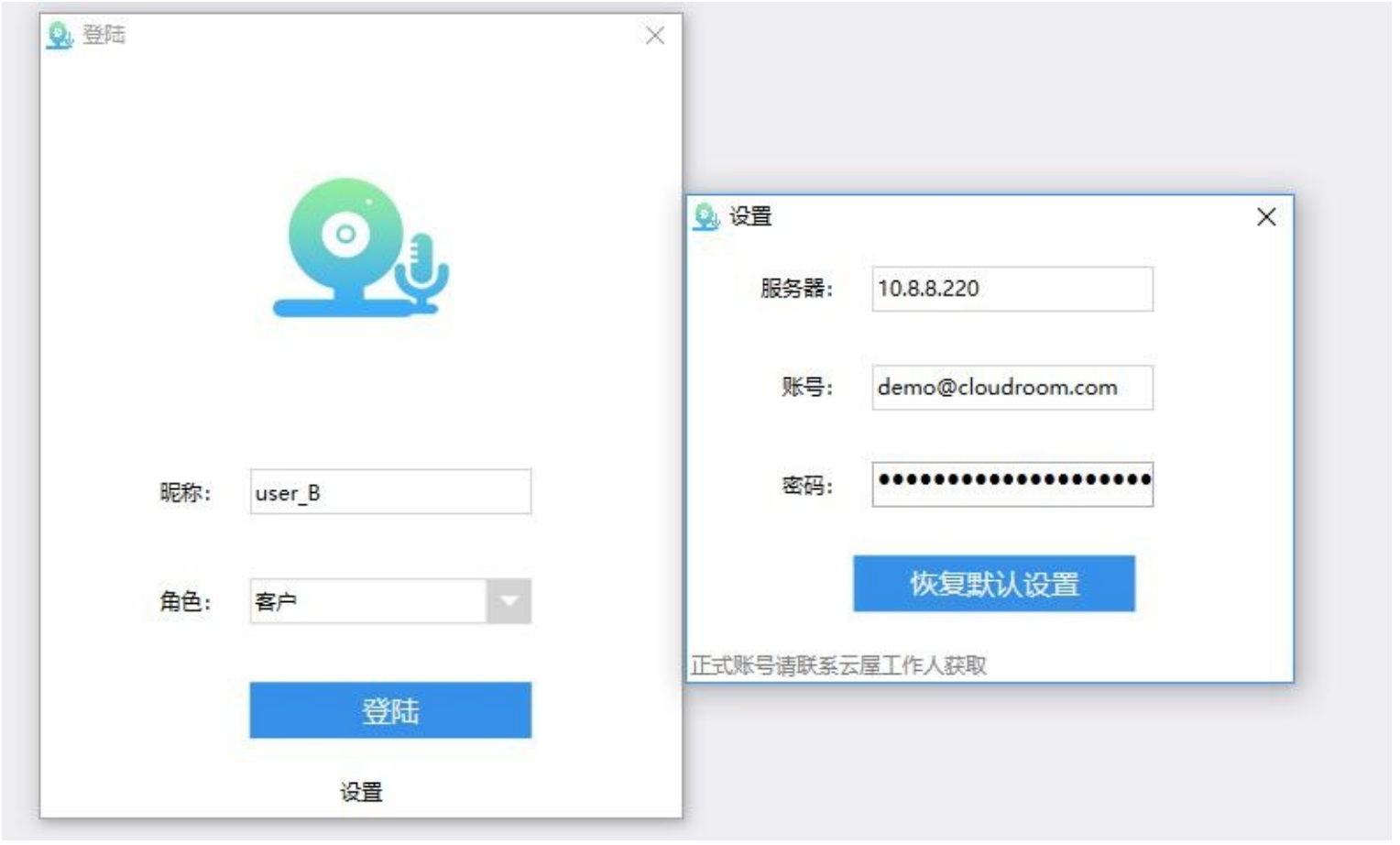
- 安装SDK，然后用Visual Studio 2008(或更高版本)打开SDK开发包内项目文件



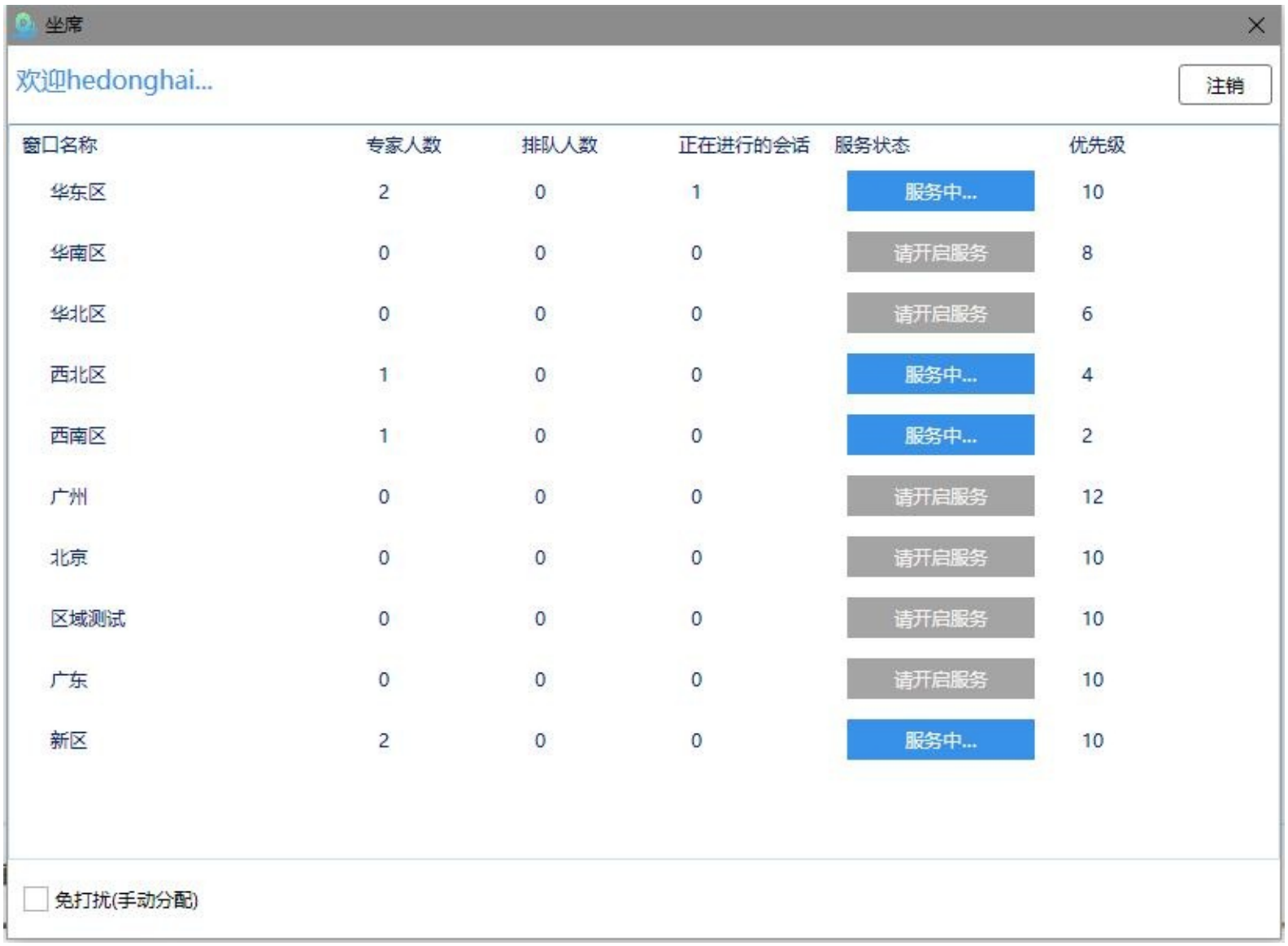
- 选择解决方案配置、平台

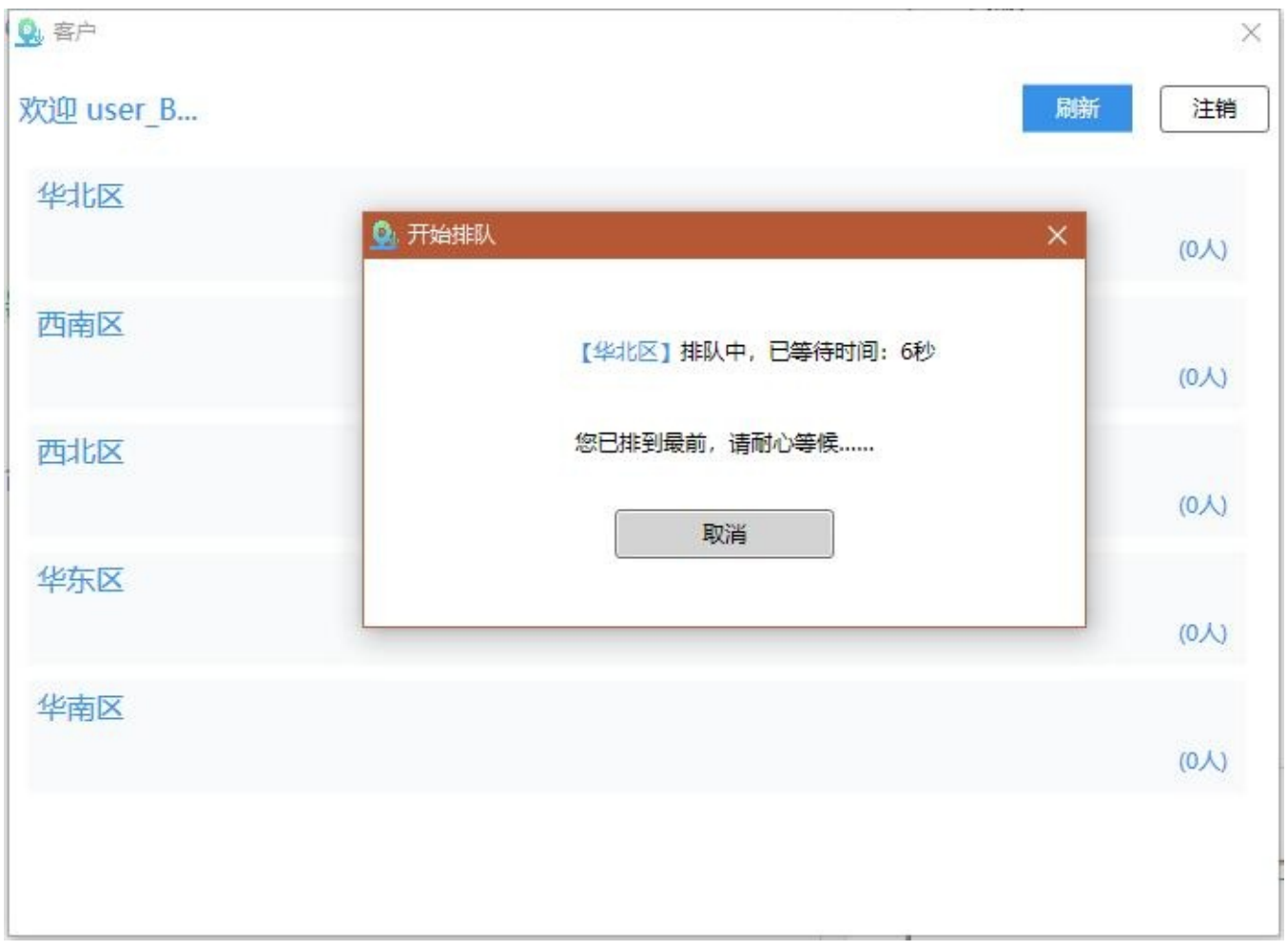


- 编译后在两台机器上分别运行，输入服务器地址（可以是自建服务器地址，也可以使用云屋公有地址），选择角色，登录

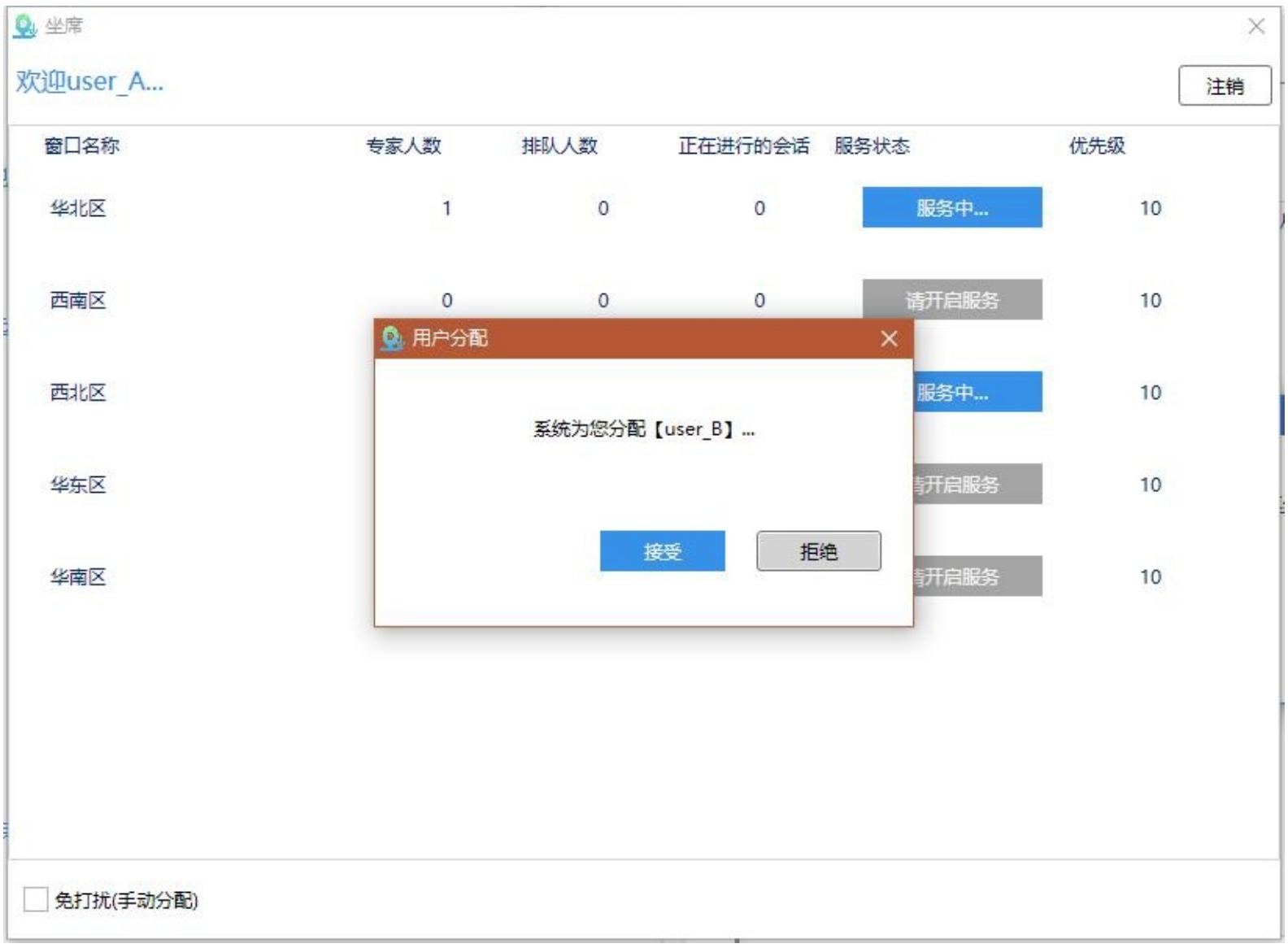


- 进入队列主界面，坐席可同时开始服务若干队列，客户每次则只能选择一个队列进行排队

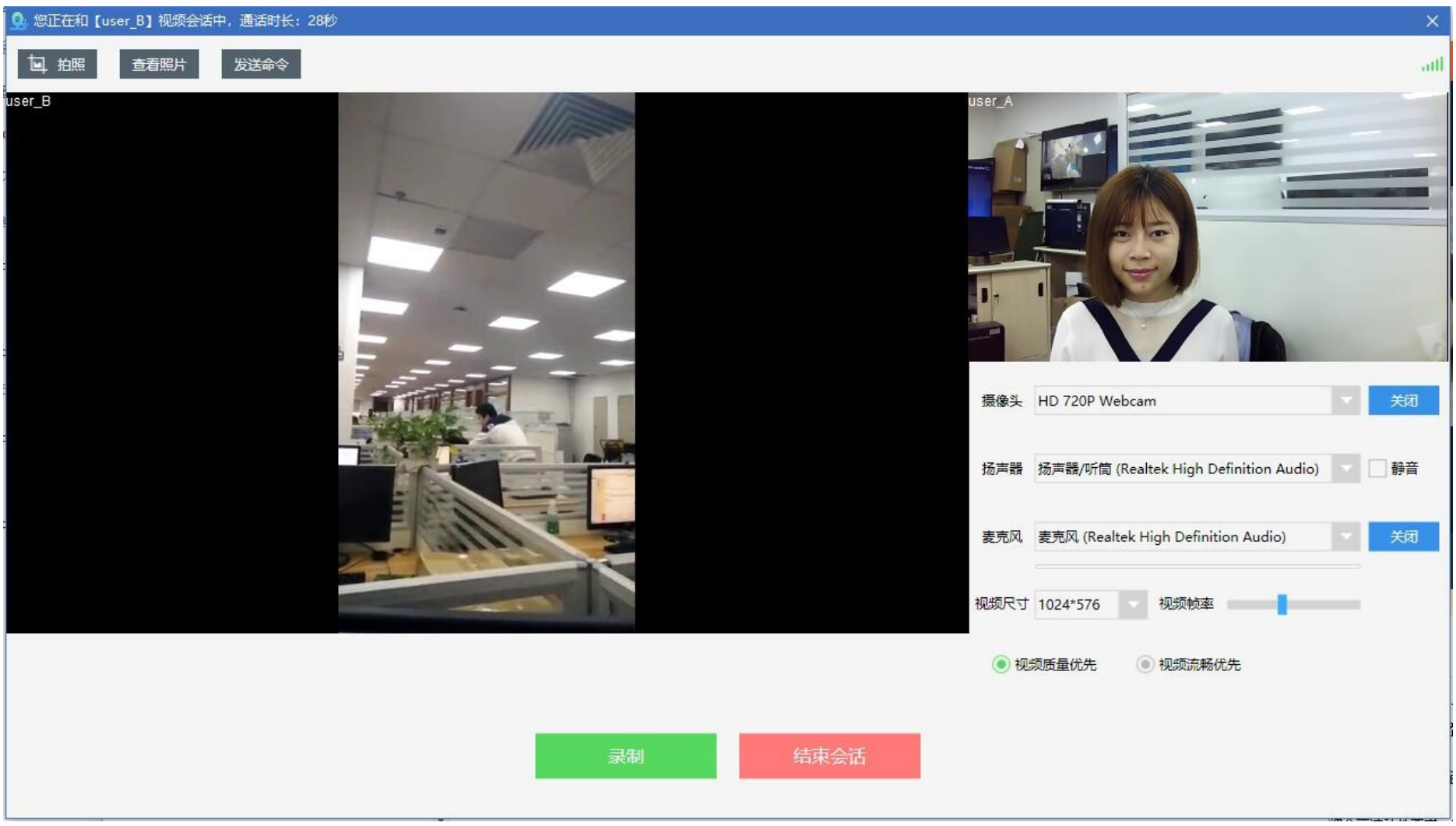




- 坐席收到队列系统分配的排队用户



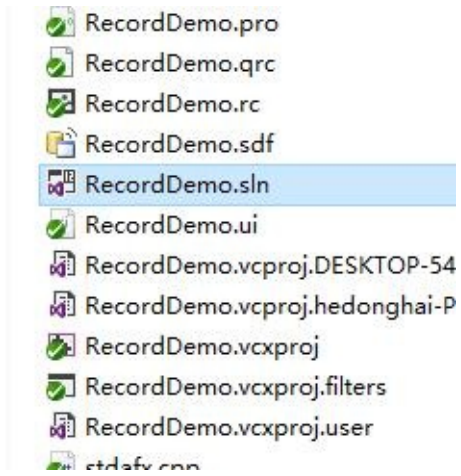
- 接受分配的用户即可进入视频会话中



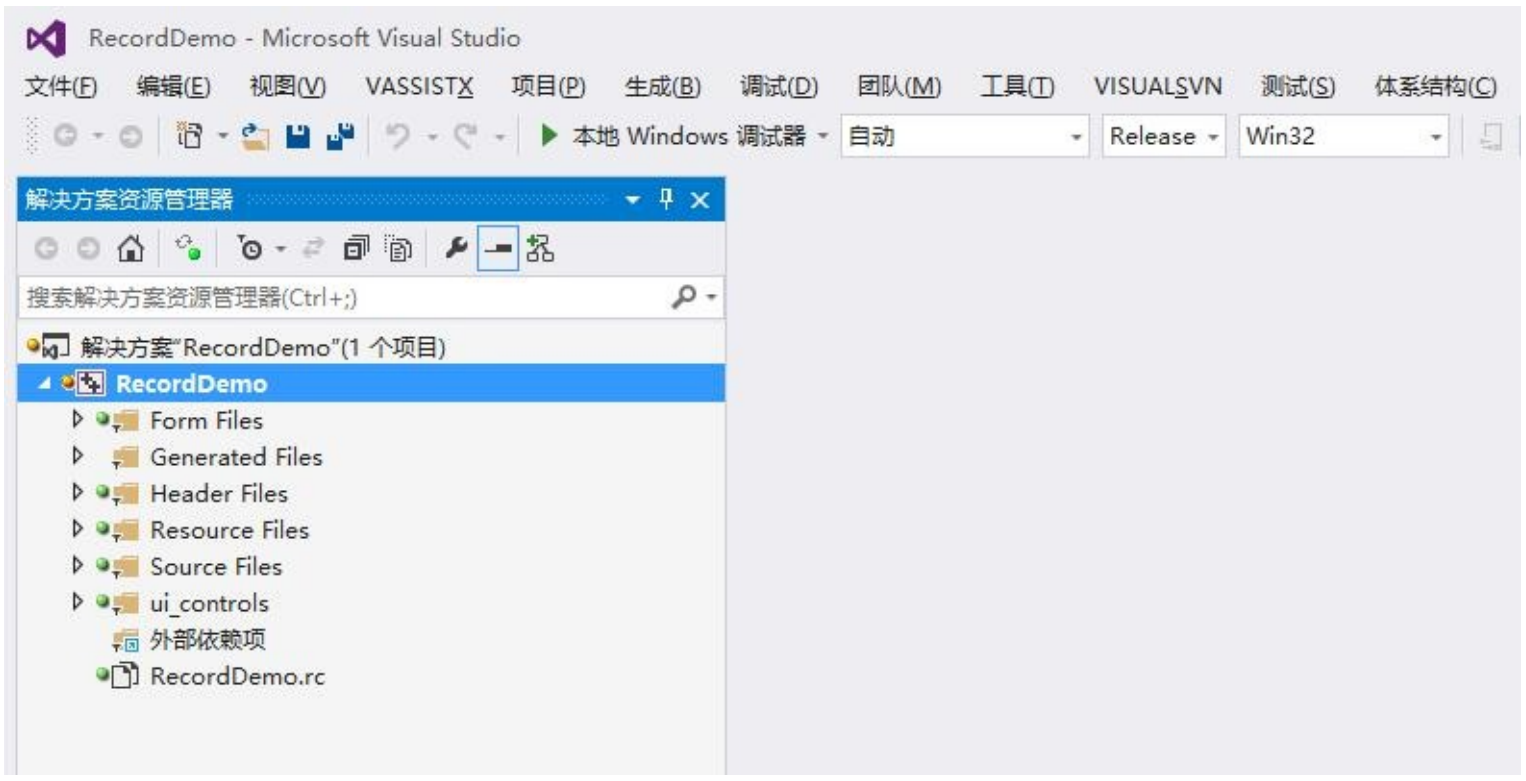
金融合规双录

> C++ Qt编写，主要实现了录制本地音视频并上传等功能。

- 安装SDK，然后用Visual Studio 2008(或更高版本)打开SDK开发包内项目文件

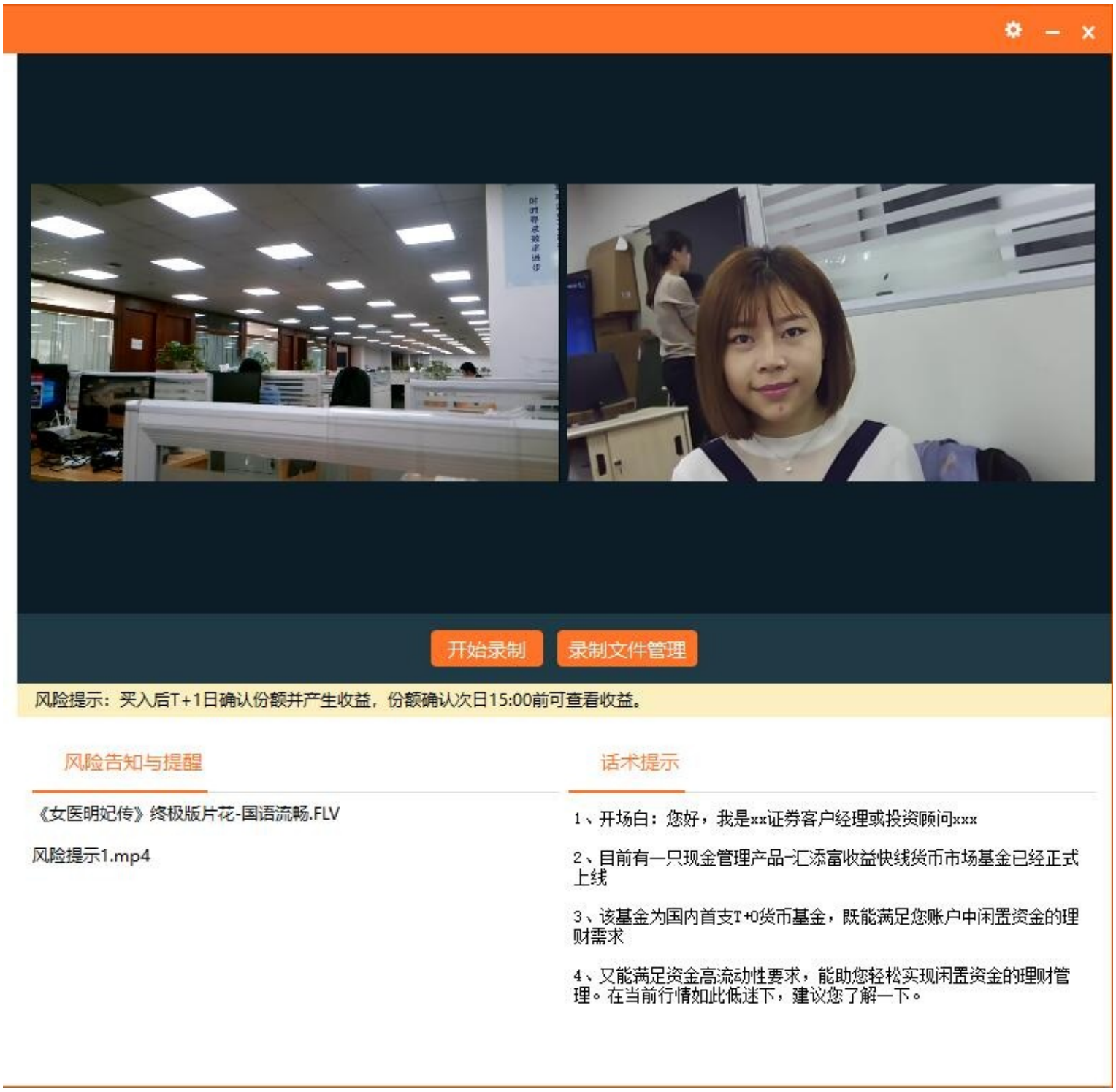


- 解决方案配置和平台



- 编译运行，输入服务器地址（可以是自建服务器地址，也可以使用云屋公有地址）

- 登录，进入程序主界面



SDK 集成指南

> 为开发者集成音视频会话及相关高级功能提供相关指导，以下均为C# (伪)代码，详细代码请参考Examples目录下Demo源代码。

文档中“会议”和“会话”含义等同。

开始音视频会话

> 快速创建并进入一个简单的音视频会话；
>
> 请先准备[说明]中的相关内容：1.安装视频SDK插件, 2.运行环境要求, 3.开发环境配置以及连接相关的麦克风摄像头并确认设备工作正常。

基本步骤如下：

1. 初始化SDK
2. 登录连接视频服务器
3. 创建视频会话
4. 进入会话
5. 打开麦克风/摄像头
6. 有其他人进入会话
7. 退出会话
8. 注销登陆

1.初始化SDK

> 初始化是整个SDK的使用基础，通常在程序启动的时候进行初始化(init)，退出的时候进行反初始化(uninit)，整个程序的生命周期中只进行一次初始化和反初始化。
>
> 相关API参考请见 初始化/反初始化

SDK内部的组件多为单例组件，整个程序中只能有一个实例，比如“基础组件”，“会议管理组件”，“视频会议组件”，具体请参见各个组件说明。

```
//引入并封装相关的SDK组件
//CR_SDK_VideoCall.cs
using Axnpcloudroomvideosdklib;
namespace VideoCall
{
    //本类封装的组件为单例组件，为整个程序提供SDK相关的操作
    public partial class CR_SDK_VideoCall : UserControl
    {
        public CR_SDK_VideoCall()
```

```
        {
            InitializeComponent();
        }
        //SDK基础组件
        public AxCloudroomVideoSDK Video
        {
            get { return axCRVideoSDK; }
        }
        //会话管理组件
        public AxCloudroomVideoMgr Mgr
        {
            get { return axCRMgr;  }
        }
        //视频会议组件
        public AxCloudroomVideoMeeting Meeting
        {
            get { return axCRMeeting; }
        }
    }
}
```

```
//登录类
// login.cs
using Axnpcloudroomvideosdklib;
namespace VideoCall
{
    public partial class Login : Window
    {
        //登录界面使用单例模式
        private static Login instance = null;
        //CRVideo内部封装了SDK相关的单例组件
        public CR_SDK_VideoCall CRVideo = new CR_SDK_VideoCall();
        public Login()
        {
            //初始化，传入sdk工作目录，注意目录的写权限问题
            CRVideo.Video.init(Environment.CurrentDirectory);
            //关联SDK的通知和回调函数
            initDelegate();
        }
        public static Login Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new Login();
                }
                return instance;
            }
        }
```

```
        private void initDelegate() //注意委托的连接和释放，重复连接委托会造成重复收到消息
        {
            //登录视频服务器结果
            CRVideo.Mgr.loginSuccess += new ICloudroomVideoMgrEvents_loginSuccessEven
tHandler(loginSuccess);
            CRVideo.Mgr.loginFail += new ICloudroomVideoMgrEvents_loginFailEventHandl
er(loginFailed);
            //收掉掉线通知后进行掉线处理
            CRVideo.Mgr.lineOff += new ICloudroomVideoMgrEvents_lineOffEventHandler(l
ineOff);
            //创建视频会议
            CRVideo.Mgr.createMeetingSuccess += new ICloudroomVideoMgrEvents_createMe
etingSuccessEventHandler(createMeetingSuccess);
            CRVideo.Mgr.createMeetingFail += new ICloudroomVideoMgrEvents_createMeeti
ngFailEventHandler(createMeetingFail);
            //自己进入视频会议
            CRVideo.Meeting.enterMeetingRslt += new ICloudroomVideoMeetingEvents_ente
rMeetingRsltEventHandler(enterMeetingRslt);
            //其他人进入会议通知
            CRVideo.Meeting.userEnterMeeting += new ICloudroomVideoMeetingEvents_user
EnterMeetingEventHandler(userEnterMeeting);
        }
    }
}
```

2.登录连接视频服务器

> 设置视频服务器地址，使用云屋授权账号和自定义用户编号登录
>
> 相关API请参考 服务器地址，登录/注销

```
// login.cs
using Axnpcloudroomvideosdklib;
namespace VideoCall
{
    public partial class Login : Window
    {
        public CR_SDK_VideoCall CRVideo = new CR_SDK_VideoCall();
        //....
        private void btnLogin_Click(object sender, RoutedEventArgs e)
        {
            //设置视频服务器地址
            CRVideo.Video.serverAddr = edtServer.Text.Trim();
            //此处的账户和密码是为公测账户和密码，正式使用请联系云屋工作人员获取私有账户和密码
            string account = "demo@cloudroom.com";
            string password = "123456";
            MyUserId = edtUserID.Text.Trim();
            nickName = edtNickname.Text.Trim();
            CRVideo.Mgr.login(account, pswMd5, nickName, MyUserId, "", "");
        }
```

```
        private void loginSuccess(object sender, ICloudroomVideoMgrEvents_loginSucces
sEvent e)
        {
            //登录成功，开始创建视频会话，见下一步
        }
        private void loginFailed(object sender, ICloudroomVideoMgrEvents_loginFailEve
nt e)
        {
            Console.WriteLine("login Failed:" + e.p_sdkErr);
            //登录出错，可以弹出错误提示，或调用登录接口再次重试登录
        }
    }
}
```

3.创建视频会话

> 输入会议标题，创建一个没有密码的视频会话
>
> 相关API请参考 创建/销毁视频会议

```
// login.cs
using Axnpcloudroomvideosdklib;
namespace VideoCall
{
    public partial class Login : Window
    {
        //保证SDK的相关组件在程序中只引入一次，不能重复引入
        public CR_SDK_VideoCall CRVideo = new CR_SDK_VideoCall();
        private void loginSuccess(object sender, ICloudroomVideoMgrEvents_loginSucces
sEvent e)
        {
            //0：不使用会议密码
            string subject = "测试视频会议";
            Login.Instance.CRVideo.Mgr.createMeeting(subject, 0, "");
        }
        public void createMeetingSuccess(object sender, ICloudroomVideoMgrEvents_crea
teMeetingSuccessEvent e)
        {
            //创建会议成功，可以开始进入会议了，见下一步
        }
        public void createMeetingFail(object sender, ICloudroomVideoMgrEvents_createM
eetingFailEvent e)
        {
            //创建会议失败……
        }
    }
}
```

4.进入会话

> 用创建成功的会话信息（会议ID和密码）进入会话，其他用户也是利用此会话信息进入该会话。
>
> 相关API请参考 进入/退出/结束会议

```
//login.cs
using Axnpcloudroomvideosdk Lib;
namespace VideoCall
{
    public partial class Login : Window
    {
        public CR_SDK_VideoCall CRVideo = new CR_SDK_VideoCall();
        //....
        public void createMeetingSuccess(object sender, ICloudroomVideoMgrEvents_crea
teMeetingSuccessEvent e)
        {
            Meet meet = JsonConvert.DeserializeObject<Meet>(e.p_meetObj);
            //用登录的用户名和密码进入
            CRVideo.Meeting.enterMeeting(meet.ID, meet.pswd, MyUserId, nickName, "");
        }
        private void enterMeetingRslt(object sender, ICloudroomVideoMeetingEvents_ent
erMeetingRsltEvent e)
        {
            if (e.p_sdkErr == 0)
            {
                //入会成功，进入视频会话的主界面，然后配置麦克风和摄像头
                VideoSessionWin videoWin = new VideoSessionWin();
                videoWin.show();    //此处应该用非模态对话框，否则会阻塞SDK内部的消息循环
            }
            else
            {
                //入会失败...
            }
        }
    }
}
```

5.打开麦克/摄像头

> 进入会话成功后，配置并打开自己的麦克风和摄像头
>
> 相关API请参考 麦克风/扬声器列表，麦克风设置，摄像头设备列表，视频设置，会议内可观看摄像头列表，开/关摄像头，视频状态，获取/设置默认视频，成员视频UI显示组件
> 相关结构定义请参考 音频配置，用户视频信息，用户视频信息列表

```
//会话窗口类
//VideoSessionWin.cs
using Axnpcloudroomvideosdk Lib;
namespace VideoCall
{
    public partial class VideoSessionWin : Window
```

```
    {
        public class audioCfg   //参见对象结构定义中的【音频配置AudioCfgObj】定义
        {
            public string micName;
            public string speakerName;
            public int privAgc;
            public int privEC;
        };
        public class VideoInfo  // 参见对象结构定义中的【用户视频信息VideoCfgObj】
        {
            public string userID;
            public int videoID;
            public string videoName;
        };
        public VideoSessionWin()
        {
            //打开相关设备
            initAVDevices();
            //入会成功以后可以给视频UI组件绑定视频源
            updateVideoUIs();
        }
        private void initAVDevices()
        {
            //获取自己机器上的麦克风和扬声器设备，结果字符串用换行符拆分
            string[] mics = Login.Instance.CRVideo.Meeting.getAudioMicNames().Split(n
ew Char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
            string[] speakers = Login.Instance.CRVideo.Meeting.getAudioSpkNames().Spl
it(new Char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
            //""为使用系统默认设备，也可以使用获取的机器上的某个设备
            audioCfg cfg = new audioCfg();
            cfg.micName = "";
            cfg.speakerName = "";
            Login.Instance.CRVideo.Meeting.setAudioCfg(JsonConvert.SerializeObject(cf
g));
            //获取自己机器上摄像头设备列表
            List<VideoInfo> devs = JsonConvert.DeserializeObject<List<VideoInfo>>(CRV
ideo.Meeting.getAllVideoInfo(Login.Instance.MyUserId));
            //可用下拉框展示上述获取到的音视频设备列表.....
            //选择我的第一个视频设备作为默认显示设备
            Login.Instance.CRVideo.Meeting.setDefaultVideo(MyUserId, devs[0].videoID)
;
            //打开麦克风和摄像头
            Login.Instance.CRVideo.Meeting.openMic(Login.Instance.MyUserId);
            Login.Instance.CRVideo.Meeting.openVideo(Login.Instance.MyUserId);
        }
    }
}
```

6.有其他人进入会话

> 其他人入会的步骤也是上述的[4、5]步，拿到会议信息后进入到他人创建的会议，此步骤的目的是为了实时关注比自己晚进来的人并刷新摄像头画面显示；如果想要获取之前进来的人，可以调用 getAllMembers 获取会议成员列表，也可以调用 getWatchableVideos 获取所有可以观看的摄像头列表进行加载。
>
> 用成员视频UI显示组件创建多个视频窗口，来显示进入会议内的成员。
>
> 相关API请参考 有人进入/离开会议通知，成员视频UI显示组件，会议内可观看摄像头列表
> 相关结构定义请参考 用户视频信息列表

```
//VideoSessionWin.cs
using Axnpcloudroomvideosdk Lib;
namespace VideoCall
{
    public partial class VideoSessionWin : Window
        // ......
        public class UserVideo
        {
            public string userID;
            public int videoID;
            public UserVideo(string uID, int vID)
            {
                userID = uID;
                videoID = vID;
            }
        };
        private void userEnterMeeting(object sender, ICloudroomVideoMeetingEvents_use
rEnterMeetingEvent e)
        {
            //有人进来了,把他的画面显示在程序的界面上
            updateVideoUIs();
        }
        //统一更新所有可以显示的设备的信息，【目前假设会话内只有三个人】
        //可以编写类似这样的函数，在任何需要更新视频画面的地方调用
        private void updateVideoUIs()
        {
            //清空之前残留的画面信息和视频源
            MyVideoUI.clear();
            myVideoUI.setVideo("", 0);
            VideoUI_1.clear();
            VideoUI_1.setVideo("", 0);
            VideoUI_2.clear();
            VideoUI_2.setVideo("", 0);
            //获取所有可观看的摄像头列表，【假设有三个】
            List<UserVideo> vidList = JsonConvert.DeserializeObject<List<UserVideo>>(
Login.Instance.CRVideo.Meeting.getWatchableVideos());
            myVideoUI.setVideo(vidList[0].userID, vidList[0].videoID);
            VideoUI_1.setVideo(vidList[1].userID, vidList[1].videoID);
            VideoUI_2.setVideo(vidList[2].userID, vidList[2].videoID);
        }
    }
}
```

> 至此，一个简单的音视频会话就建立起来了。

7.退出会话

> 在未注销的情况下可反复的进入退出同一个会议。
>
> 相关API请参考 进入/退出/结束会议

```
//离开会议
Login.Instance.CRVideo.Meeting.exitMeeting();
```

8.注销登陆

> 可重复的登录和注销。
>
> 相关API请参考 进入/退出/结束会议，登录/注销，初始化/反初始化

```
Login.Instance.Logout();
```

9.反初始化，退出SDK

> 执行反初始化后SDK功能不再可用。
>
> 相关API请参考 初始化/反初始化

```
CRVideo.Video.uninit();
```

添加音视频会话功能

> 添加会议内的高级功能

音视频控制

> 进入会话后实现设备的加载、选择、设置
>
> 1. 开关麦克风
> 2. 监控麦克风状态变化
> 3. 设置麦克风和扬声器音量
> 4. 监控麦克风声音大小变化

1.开关麦克风

通过传入参数来控制开关的对象，如果是本地机器上的麦克风，需要传入自己的ID，如果要远程开关他人麦克风，则传入对方的ID

- 相关API请参考 麦克风、扬声器设备的获取，开/关麦克风

```
//打开自己的麦克风
Login.Instance.CRVideo.Meeting.openMic(Login.Instance.MyUserId);
```

```
//关闭自己的麦克风
Login.Instance.CRVideo.Meeting.closeMic(Login.Instance.MyUserId);
```

2.监控麦克风状态变化

```
1. 开关自己或他人的的麦克风都会收到该回调函数
2. 自己的麦克风被他人开关也会收到该回调函数
3. 也可主动获取麦克风状态，一般用于各种判断
```

- 相关API请参考 麦克风状态变化
- 相关结构定义请参考 麦克风状态

```
//连接状态变化委托
Login.Instance.CRVideo.Meeting.audioStatusChanged += new ICloudroomVideoMeetingEvents
_audioStatusChangedEventHandler(audioStatusChanged);
```

```
//音频状态发生变化
public void audioStatusChanged(object sender, ICloudroomVideoMeetingEvents_audioStatu
sChangedEvent e)
{
    //任何参会人的麦克风状态变化了这里都会收到
    if (e.p_userID == Login.Instance.MyUserId) //自己的麦克风状态发生了变化
    {
        if (e.p_newStatus == 2)
        {
            btnMicOpr.Content = "打开麦克风";
        }
        else if(e.p_newStatus == 3)
        {
            btnMicOpr.Content = "关闭麦克风";
        }
        else if(e.p_newStatus == 4)
        {
            btnMicOpr.Content = "操作中……";
        }
    }
    else//处理其他的麦克风状态变化
    {
        //...
    }
}
```

3.设置麦克风和扬声器音量

- 相关API请参考 麦克风音量](meeting.md#micVolume)，扬声器音量

```
//用滑动条来调节麦克风和扬声器的声音
private void micBar_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double
> e)
{
    Login.Instance.CRVideo.Meeting.micVolume = (int)micBar.Value;
}
private void speakerBar_ValueChanged(object sender, RoutedPropertyChangedEventArgs<do
uble> e)
{
    Login.Instance.CRVideo.Meeting.speakerVolume = (int)speakerBar.Value;
}
```

4.监控麦克风声音变化

可用来实时展示当前麦克风采集到声音的大小

- 相关API请参考 麦克风声音变化

```
//连接声音变化委托
Login.Instance.CRVideo.Meeting.micEnergyUpdate += new ICloudroomVideoMeetingEvents_mi
cEnergyUpdateEventHandler(micEnergyUpdate);
```

```
//麦克风音量波动
public void micEnergyUpdate(object sender, ICloudroomVideoMeetingEvents_micEnergyUpda
teEvent e)
{
    if (e.p_userID == Login.Instance.MyUserId)
    {
        micEnergy.Value = e.p_newLevel;  //更新音量进度条
    }
    else  //
    {
        // ...
    }
}
```

录制

> 实现摄像头和屏幕录制，并可上传到服务器
>
> 1. 录制内容配置
> 2. 开始、停止录制
> 3. 录制文件的大小、时长、状态

4. 设置录制文件是否加密
5. 录制文件列表
6. 录制文件列表添加、删除文件
7. 上传、取消上传录制文件
8. 回放录制文件

1.录制内容配置

- 相关API请参考 录制内容配置
- 相关结构定义请参考 录制内容配置

```
//定义相关结构对象
public class RecordCfg  //录制文件定义
{
    public string filePathName;
    public int recordWidth;     //录制文件的宽高
    public int recordHeight;
    public int frameRate;
    public int bitRate;
    public int defaultQP;
    public int recDataType;
};
public class Record //录制内容定义
{
    public int left;    //录制有效视频内容的位置和宽高
    public int top;
    public int width;
    public int height;
    public int type; //录制类型，参看文档REC_VCONTENT_TYPE定义
    public int keepAspectRatio;
};
public class RecordCam : Record
{
    public class CamParams
    {
        public string camid;
    };
    public CamParams param = new CamParams();
};
public class RecordCamList  //录制摄像头
{
    public List<RecordCam> cams = new List<RecordCam>;
}
public class RecordScreen : Record //录制屏幕
{
    public class ScreenParams
    {};

    public ScreenParams param = new ScreenParams;
};
```

```
//配置录制内容
private void updateRecordContent()
{
    string recordContect = "";
    if (/*录摄像头*/) //画面布局为画中画模式，对方为大画面，我方为小画面，放在对方画面的右下角上
    {
        //对方画面参数
        RecordCam recordCamBig = new RecordCam();
        recordCamBig.left = 0;
        recordCamBig.top = 0;
        recordCamBig.width = 1280;
        recordCamBig.height = 720;
        recordCamBig.type = (int)REC_VCONTENT_TYPE.RECVTP_VIDEO;
        recordCamBig.keepAspectRatio = 1;
        recordCamBig.param.camid = String.Format("{0}.{1}", mPeerUserID, Login.Instan
ce.CRVideo.Meeting.getDefaultVideo(mPeerUserID)); //对方的userID和摄像头ID
        //我方画面参数
        RecordCam recordCamSmall = new RecordCam();
        recordCamSmall.left = 1280 - 336;
        recordCamSmall.top = 720 - 192; ;
        recordCamSmall.width = 336;
        recordCamSmall.height = 192;
        recordCamSmall.type = (int)REC_VCONTENT_TYPE.RECVTP_VIDEO;
        recordCamSmall.keepAspectRatio = 1;
        recordCamSmall.param.camid = String.Format("{0}.{1}", MyUserID, -1); //我的use
rID和默认摄像头
        //添加录制的摄像头
        List<RecordCam> camlist = new List<RecordCam>() { recordCamBig, recordCamSmal
l };
        recordContect = JsonConvert.SerializeObject(camlist);
    }
    else if(/*录屏幕区域*/)
    {
        RecordScreen recordScreen = new RecordScreen();
        recordScreen.left = 0;
        recordScreen.top = 0;
        recordScreen.width = 1920;
        recordScreen.height = 1080;
        recordScreen.type = (int)REC_VCONTENT_TYPE.RECVTP_SCREEN;
        //录制的屏幕区域，可全屏，可局部
        List<RecordScreen> screenList = new List<RecordScreen> { recordScreen };
        recordContect = JsonConvert.SerializeObject(screenList);
    }
    Login.Instance.CRVideo.Meeting.setRecordVideos(recordContect);
}
```

2.开始/停止录制

1. 先开始录制，配置好录制文件信息
2. 然后更新录制内容配置

3. 在录制过程中可以多次更新录制配置，从而变更录制内容

- 相关API请参考 开始/停止录制
- 相关结构定义请参考 录制内容类型，录制文件配置

```
//定义录制文件参数
RecordCfg cfg = new RecordCfg();
cfg.filePathName = "D:\\Record\\" + DateTime.Now.ToString("yyyyMMddHHmmss") + ".mp4";
cfg.recordWidth = 1280; //宽度
cfg.recordHeight = 720; //高度
cfg.frameRate = 15;     //帧率
cfg.bitRate = 1000000;  //码率
cfg.defaultQP = 24;     //清晰度
cfg.recDataType = (int)(REC_DATATYPE.REC_AUDIO_LOC | REC_DATATYPE.REC_AUDIO_OTHER | R
EC_DATATYPE.REC_VIDEO); //录制内容
//开始录制
Login.Instance.CRVideo.Meeting.startRecording(JsonConvert.SerializeObject(cfg));

//开始录制后，更新录制配置
updateRecordContent(); //上方自定义函数
```

```
//停止录制
Login.Instance.CRVideo.Meeting.stopRecording();
```

3.录制文件的大小、时长、状态

- 相关API请参考录制文件大小，时长，录制状态变化通知
- 相关结构定义请参考 错误码定义,录制状态，通知录制文件状态变化

```
//录制时长
int recordLen = Login.Instance.CRVideo.Meeting.getRecDuration();
//录制大小（MB）
double fileSzie = (double)Login.Instance.CRVideo.Meeting.getRecFileSize() / (1024 * 1
024);
```

```
//关联录制相关委托
Login.Instance.CRVideo.Meeting.recordErr += new ICloudroomVideoMeetingEvents_recordEr
rEventHandler(recordErr);
Login.Instance.CRVideo.Meeting.notifyRecordFileStateChanged += new ICloudroomVideoMee
tingEvents_notifyRecordFileStateChangedEventHandler(Meeting_notifyRecordFileStateChan
ged);
```

```
//处理录制错误状态通知
private void recordErr(object sender, ICloudroomVideoMeetingEvents_recordErrEvent e)
{
    //录制发生错误，代码：  e.p_sdkErr,见[错误码定义]
```

```
}
```

```
//录制文件状态变化通知
private void Meeting_notifyRecordFileStateChanged(object sender, ICloudroomVideoMeeti
ngEvents_notifyRecordFileStateChangedEvent e)
{
    //p_fileName, 文件名
    //p_state, 状态
}
```

4.设置录制文件是否加密

启动录制时调用，则本次录制是否加密本地录制文件，也可全局配置，这样每一次录制都是加密的。

- 相关API请参考 设置录制文件是否加密

```
Login.Instance.CRVideo.Meeting.setRecordFileEncrypt(1);
```

5.录制文件列表

- 相关API请参考 录制文件列表
- 相关结构定义请参考 录制文件列表

```
public class RecordFileList
{
    public class RecordFile
    {
        public string fileName = "";
        public ulong size = 0;
        public int state = 0;
        public int uploadPercent=0;
    }
    public List<RecordFile> fileList = new List<RecordFile>();
}
//获取文件列表
RecordFileList recordFileList = JsonConvert.DeserializeObject<RecordFileList>(Login.I
nstance.CRVideo.Meeting.getAllRecordFiles());
```

6.录制文件列表添加、删除文件

1. 添加录制文件到录制文件列表，这样此文件便可上传和回放
2. 移除文件时本地文件会被删掉，正在上传的文件会被取消上传，已经上传的文件不受影响

- 相关API请参考 录制列表添加/删除文件

```
//添加文件到录制文件列表中
Login.Instance.CRVideo.Meeting.addFileToRecordMgr("test.mp4", "d:\downloads\videos\")
;
```

```
//从录制文件列表中移除文件，并删除本地文件
Login.Instance.CRVideo.Meeting.removeFromFileMgr("d:\downloads\videos\test.mp4");
```

7.上传、取消上传录制文件

参数是绝对路径文件名

- 相关API请参考 上传/取消上传录制文件，通知录制文件上传进度，通知录制文件状态变化
- 相关结构定义请参考 录制文件列表，录制文件上传状态

```
//开始上传
Login.Instance.CRVideo.Meeting.uploadRecordFile("d:\downloads\videos\test.mp4");
```

```
//关联上传进度通知委托
Login.Instance.CRVideo.Meeting.notifyRecordFileUploadProgress+=new ICloudroomVideoMee
tingEvents_notifyRecordFileUploadProgressEventHandler(Meeting_notifyRecordFileUploadP
rogress);

Login.Instance.CRVideo.Meeting.notifyRecordFileStateChanged += new ICloudroomVideoMee
tingEvents_notifyRecordFileStateChangedEventHandler(Meeting_notifyRecordFileStateChan
ged);
```

```
//处理上传进度通知
private void Meeting_notifyRecordFileUploadProgress(object sender, ICloudroomVideoMee
tingEvents_notifyRecordFileUploadProgressEvent e)
{
    // 上传文件:  p_fileName
    // 上传进度:  p_percent
}
```

```
//录制文件上传状态通知
private void Meeting_notifyRecordFileStateChanged(object sender, ICloudroomVideoMeeti
ngEvents_notifyRecordFileStateChangedEvent e)
{
    //p_fileName, 文件名
    //p_state,  状态
}
```

```
//取消上传
```

```
Login.Instance.CRVideo.Meeting.cancelUploadRecordFile("d:\downloads\videos\test.mp4")
;
```

8.回放录制文件

此接口需要配合影音共享UI显示组件或者影音播放相关的接口进行画面和声音展示

> 相关API请参考 影音共享UI显示组件，回放录制文件

```
//开始回放
Login.Instance.CRVideo.Meeting.playbackRecordFile("d:\downloads\videos\test.mp4");
```

文件上传下载

1. 网盘容量、文件列表
2. 网盘文件操作（上传、下载、暂停、删除）

1.网盘容量、已上传文件列表

获取网盘的使用情况和已经上传到服务器的文件列表

- 相关API请参考 会议网盘容量，获取网盘文件列表
- 相关结构定义请参考 网盘文件，网盘文件列表

```
//关联相关委托
Login.Instance.CRVideo.Meeting.getNetDiskSummaryRslt += new ICloudroomVideoMeetingEve
nts_getNetDiskSummaryRsltEventHandler(Meeting_getNetDiskSummaryRslt);
Login.Instance.CRVideo.Meeting.getNetDiskFileListRslt += new ICloudroomVideoMeetingEv
ents_getNetDiskFileListRsltEventHandler(Meeting_getNetDiskFileListRslt);
```

```
//查询容量和文件列表
Login.Instance.CRVideo.Meeting.getNetDiskSummary();
Login.Instance.CRVideo.Meeting.getNetDiskFileList();
```

```
//查询到容量
private void Meeting_getNetDiskSummaryRslt(object sender, ICloudroomVideoMeetingEvent
s_getNetDiskSummaryRsltEvent e)
{
    string desc = String.Format("总容量{0}MB, 已使用{1}MB", e.p_diskLimit/1024, e.p_dis
kUsed/1024);
}
```

```
//已经上传的文件列表
```

```
private void Meeting_getNetDiskFileListRslt(object sender, ICloudroomVideoMeetingEven
ts_getNetDiskFileListRsltEvent e)
{
    List<NetDiskObj> netDiskFiles = JsonConvert.DeserializeObject<List<NetDiskObj>>(e
.p_fileList);
    // ...
}
```

2.网盘文件操作

```
1.上传、下载、删除、暂停（上传下载）、取消操作
2.上传时需要调用SDK接口生成网盘文件ID
```

- 相关API请参考 生成网盘文件ID，上传/下载/删除网盘文件，取消网盘文件操作，暂停/继续网盘文件传输，获取网盘容量信息结果，获取网盘文件列表结果，删除网盘文件结果，网盘容量不足通知，通知网盘文件传输进度

```
string localFile = fileDialog.FileName;
FileInfo file = new FileInfo(localFile);
//调用SDK接口生成上传的文件ID
string fileID = Login.Instance.CRVideo.Meeting.makeNetDiskFileID(0, Guid.NewGuid().To
String() + file.Extension);
//开始上传
Login.Instance.CRVideo.Meeting.uploadNetDiskFile(fileID, localFile);
```

```
//下载
string fileName = ""; //从已上传的文件列表中取得文件的原始名称，有也可以自定义新名称
Login.Instance.CRVideo.Meeting.downloadNetDiskFile(fileID, "D:\\Downloads\" + fileNam
e);
```

```
//删除
Login.Instance.CRVideo.Meeting.deleteNetDiskFile(fileID);
```

```
//暂停
Login.Instance.CRVideo.Meeting.setNetDiskTransportPause(fileID, 1);
//继续
Login.Instance.CRVideo.Meeting.setNetDiskTransportPause(fileID, 0);
```

```
//取消网盘文件操作（上传/下载）
Login.Instance.CRVideo.Meeting.cancleNetDiskFile(fileID);
```

```
//网盘文件删除结果通知
private void Meeting_notifyNetDiskFileDeleteRslt(object sender, ICloudroomVideoMeetin
gEvents_notifyNetDiskFileDeleteRsltEvent e)
```

```
{
    if(e.p_isSucceed != 1)
    {
        MessageBox.Show("删除网盘文件失败: " + obj.orgFileName);
    }
}
```

```
//网盘文件操作进度展示
private void Meeting_notifyNetDiskTransforProgress(object sender, ICloudroomVideoMeet
ingEvents_notifyNetDiskTransforProgressEvent e)
{
    // p_fileID, 上传文件标识
    // p_isUpload, 上传还是下载
    // p_percent, 进度百分比
}
```

电子白板

> 实现文件内容实时同步共享

引入白板UI显示组件

使用可视化UI组件创建并使用白板

- 相关API请参考 白板显示UI显示组件

如下图:

影音播放共享

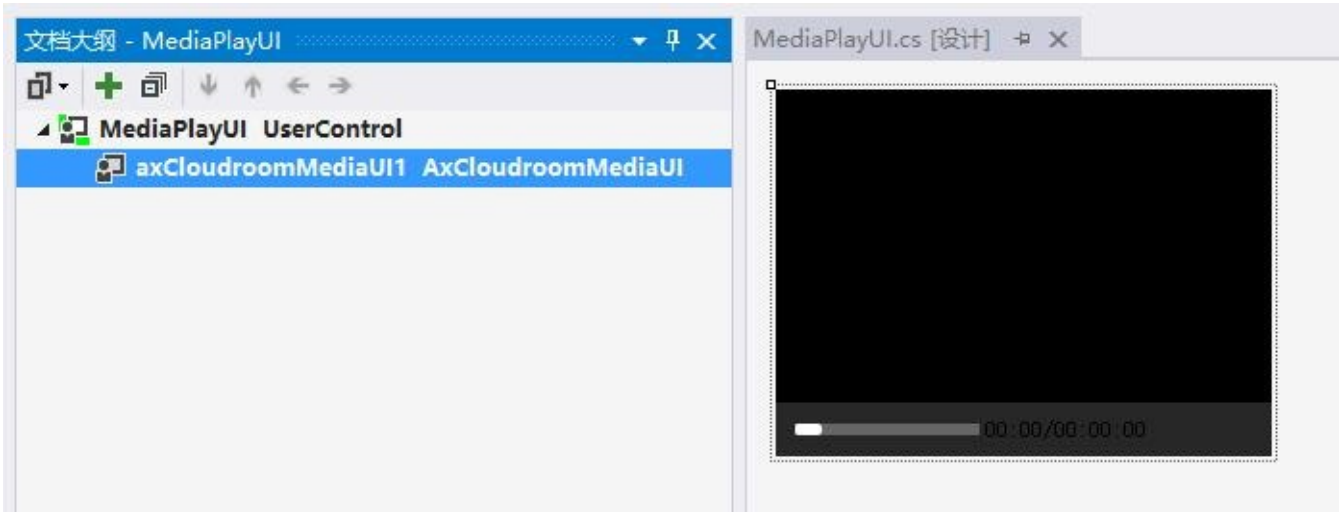
1. 引入影音播放UI显示组件
2. 播放配置
3. 播放、暂停、停止
4. 设置播放进度
5. 文件列表、播放信息、播放音量

1.引入影音播放UI显示组件

使用可视化UI组件进行影音播放

- 相关API请参考 影音共享UI显示组件

如下图



2.播放配置

此配置主要是为了定义共享播放时会议内其他人看到的效果

- 相关API请参考 影音播放配置
- 相关结构定义请参考 视频尺寸定义

```
VideoCfg cfg = new VideoCfg();
cfg.sizeType = 13;      //VSIZE_SZ_1080 13 1920*1080, 最大码率: 2mbps
cfg.fps = 24;           //帧率
cfg.qp_min = 22;        //
cfg.qp_max = 36;        //流畅优先
cfg.wh_rate = 0;        //宽高比, 0:16/9, 1:4/3, 2:1/1
Login.Instance.CRVideo.Meeting.setMediaCfg(JsonConvert.SerializeObject(cfg));
```

3.播放、暂停、停止

每次只能播放一个视频，当前播放需要先停止然后才能进行下一个视频播放；
开始播放参数可控制此播放是只有自己可见还是会议内所有人可见。

- 相关API请参考 开始/暂停/停止影音播放，通知影音打开/播放/暂停/停止，通知更新影音播放进度

```
//开始播放
Login.Instance.CRVideo.Meeting.startPlayMedia("D:\Videos\video.mp4", 0, 0);
```

```
//暂停播放
Login.Instance.CRVideo.Meeting.pausePlayMedia(0);
//继续播放
Login.Instance.CRVideo.Meeting.pausePlayMedia(1);
```

```
//停止当前播放
Login.Instance.CRVideo.Meeting.stopPlayMedia();
```

4.设置播放进度

可以通过播放组件上的工具条拖动来调整播放进度，也可以用代码来设置播放的进度

- 相关API请参考 设置播放进度

```
int pos = 123456;
Login.Instance.CRVideo.Meeting.setMediaPlayPos(pos);
```

5.文件列表、播放信息、播放音量

- 相关API请参考 影音文件列表，影音播放信息，影音播放音量
- 相关结构定义请参考 影音文件

```
//获取media目录下的所有影音文件
string[] files = Login.Instance.CRVideo.Meeting.getAllFilesInMediaPath().Split(new Ch
ar[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
```

```
//获取当前播放的影音文件信息
public class MediaInfo
{
    public string userID;
    public int state;
    public string mediaName;
}
MediaInfo mediaInfo = JsonConvert.DeserializeObject<MediaInfo>(Login.Instance.CRVideo
.Meeting.getMediaInfo());
```

```
//设置当前播放的影音声音
int vol = 24;
Login.Instance.CRVideo.Meeting.mediaVolume = vol;
```

屏幕共享

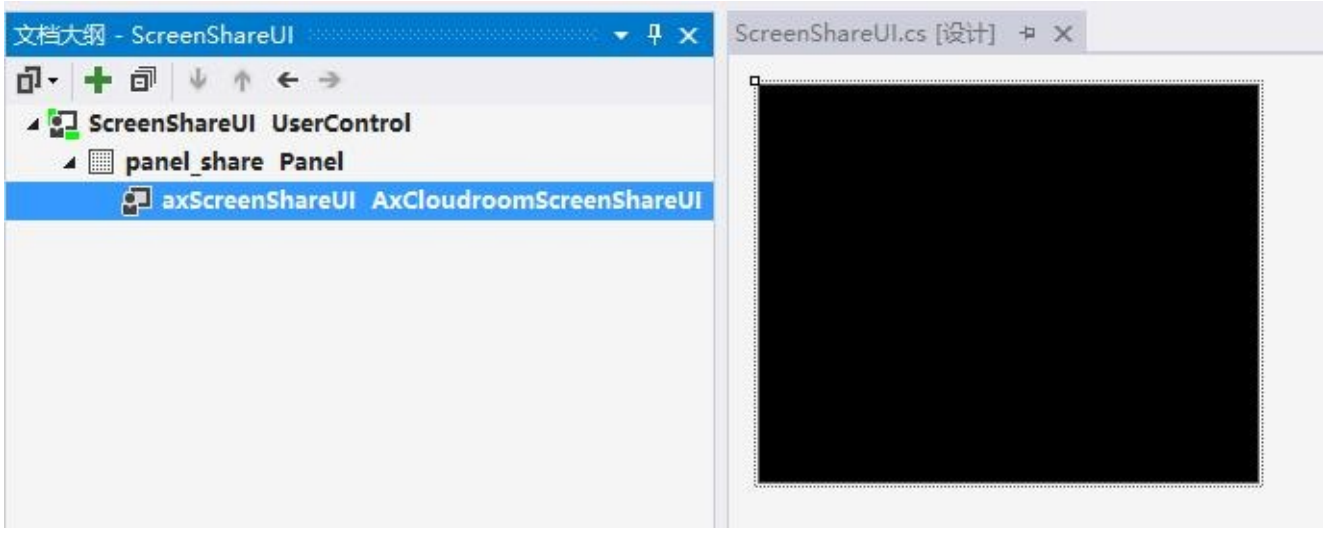
1. 引入共享UI显示组件
2. 开始、停止共享
3. 远程操作权限

1.引入共享UI显示组件

使用可视化UI组件进行屏幕共享显示和操作

- 相关API请参考 屏幕共享UI显示组件

如下图



2.开始、停止共享

使用接口启动共享，出现共享内容显示组件后，用组件上的功能开始标注和结束共享。

- 相关API请参考 屏幕共享配置，开始/停止屏幕共享，屏幕共享状态，开始/停止屏幕共享操作结果，开始/停止屏幕共享通知
- 相关结构定义请参考 屏幕共享配置

```
//关联相关委托
//开始、停止共享操作响应
Login.Instance.CRVideo.Meeting.startScreenShareRslt += new ICloudroomVideoMeetingEvents_startScreenShareRsltEventHandler(startScreenShareRslt);
Login.Instance.CRVideo.Meeting.stopScreenShareRslt += new ICloudroomVideoMeetingEvents_stopScreenShareRsltEventHandler(stopScreenShareRslt);
//会议内其他人开始、停止共享后通知
Login.Instance.CRVideo.Meeting.notifyScreenShareStarted += new EventHandler(notifyScreenShareStarted);
Login.Instance.CRVideo.Meeting.notifyScreenShareStopped += new EventHandler(notifyScreenShareStopped);
```

```
//定义共享配置
ScreenShareCfg cfg = new ScreenShareCfg();
ScreenShareCfg.CatchRect rect = new ScreenShareCfg.CatchRect();
rect.left = 0;
rect.top = 0;
rect.width = Screen.PrimaryScreen.Bounds.Width;
rect.height = Screen.PrimaryScreen.Bounds.Height;
cfg.catchRect = rect;     //区域共享，本例中共享整个桌面区域
Login.Instance.CRVideo.Meeting.setScreenShareCfg(JsonConvert.SerializeObject(cfg));
//开始共享
Login.Instance.CRVideo.Meeting.startScreenShare();
```

```
//停止共享
Login.Instance.CRVideo.Meeting.stopScreenShare();
```

```
//开始共享操作响应
public void startScreenShareRslt(object sender, ICloudroomVideoMeetingEvents_startScr
eenShareRsltEvent e)
{
    if (e.p_sdkErr != 0)
    {
        //开启屏幕共享失败: e.p_sdkErr
    }
}
```

```
//停止共享操作响应
public void stopScreenShareRslt(object sender, ICloudroomVideoMeetingEvents_stopScree
nShareRsltEvent e)
{
    if (e.p_sdkErr != 0
    {
        //停止屏幕共享失败:" e.p_sdkErr
    }
}
```

```
//对方启动屏幕共享
private void notifyScreenShareStarted(object sender, EventArgs e)
{
    // ...
}
```

```
//对方关闭屏幕共享
private void notifyScreenShareStopped(object sender, EventArgs e)
{
    // ...
}
```

3.远程操作权限

把共享区域的操作控制权限赋予某人，自己也可以获取他人赋予的操作权限

- 相关API请参考 赋予/收回远程屏幕控制权限，通知赋予/收回屏幕共享操作权限

```
//关联相关委托
Login.Instance.CRVideo.Meeting.notifyGiveCtrlRight += new ICloudroomVideoMeetingEvent
s_notifyGiveCtrlRightEventHandler(notifyGiveCtrlRight);
Login.Instance.CRVideo.Meeting.notifyReleaseCtrlRight += new ICloudroomVideoMeetingEv
ents_notifyReleaseCtrlRightEventHandler(notifyReleaseCtrlRight);
```

```
//给某人共享操作权限
Login.Instance.CRVideo.Meeting.giveCtrlRight("user_0005")
```

```
//收回操作权限
Login.Instance.CRVideo.Meeting.releaseCtrlRight("user_0005");
```

```
//我被给了操作权限
private void notifyGiveCtrlRight(object sender, ICloudroomVideoMeetingEvents_notifyGiveCtrlRightEvent e)
{
    mShareUI.ShareUI.ctrlOpen = 1;
    //您已经获得了远程屏幕控制权限
}
```

```
//我的操作权限被收回了
private void notifyReleaseCtrlRight(object sender, ICloudroomVideoMeetingEvents_notifyReleaseCtrlRightEvent e)
{
    mShareUI.ShareUI.ctrlOpen = 0;
    //您的远程屏幕控制权限已经被收回
}
```

聊天

> 实现会话内文本聊天，如果需要更加丰富的聊天内容，可自己传输文本格式，并进行相关解析

- 相关API请参考 发送IM文本消息，通知收到IM消息

```
//关联相关委托
Login.Instance.CRVideo.Meeting.sendIMmsgRlst += new ICloudroomVideoMeetingEvents_sendIMmsgRlstEventHandler(sendIMmsgRlst);
Login.Instance.CRVideo.Meeting.notifyIMmsg += new ICloudroomVideoMeetingEvents_notifyIMmsgEventHandler(notifyIMmsg);
```

```
//发送IM信息
string userID = "user_00005"; //为空时表示广播，发给所有人
Login.Instance.CRVideo.Meeting.sendIMmsg(txtMsg.Text, userID, "");
```

```
//发送结果
private void sendIMmsgRlst(object sender, ICloudroomVideoMeetingEvents_sendIMmsgRlstEvent e)
{
    if (e.p_sdkErr != 0)
    {
```

```
            //消息发送失败 e.p_sdkErr
        }
    }
```

```
//收到IM消息
private void notifyIMmsg(object sender, ICloudroomVideoMeetingEvents_notifyIMmsgEvent
 e)
{
    //int to time
    DateTime dt = new DateTime(1970, 1, 1, 0, 0, 0, 0);
    DateTime newDateTime = dt.AddSeconds(e.p_sendTime);
    string sendTime = newDateTime.ToLocalTime().ToString();
    if (e.p_fromUserID != Login.Instance.MyUserID) //收到别人的信息
    {
        string fromUsrName = Login.Instance.CRVideo.Meeting.getMemberNickName(e.p_fro
mUserID);
        string msg = "[" + fromUsrName + "]  "+ sendTime + "\n" + e.p_text + "\n";
    }
    else //自己的信息
    {
        string msg = String.Format("[我]  {0}\n ", sendTime) + e.p_text + "\n";
    }
}
```

参会成员管理

> 获取会话内人员及相关信息，得到成员的userID后可以对其进行相关的远程音视频操作

- 相关API请参考 会议成员列表，会议成员信息
- 相关结构定义请参考 会议成员

```
//获取所有参会人
List<MemberInfo> allMembers = JsonConvert.DeserializeObject<List<MemberInfo>>(Login.I
nstance.CRVideo.Meeting.getAllMembers());
```

```
//获取某个参会人的信息
MemberInfo memInfo = JsonConvert.DeserializeObject<MemberInfo>(Login.Instance.CRVideo
.Meeting.getMemberInfo("user_00005"));
```

功能页同步

> 用户会话内同步所有人的功能，包括视频墙、影音共享、屏幕共享、电子白板

- 相关API请参考 设置/获取会话内主功能页，会话内主功能页切换通知
- 相关结构定义请参考 主功能类型

```
//视频墙、共享、白板、影音共享
```

```
enum MAIN_PAGE_TYPE { MAINPAGE_VIDEOWALL, MAINPAGE_SHARE, MAINPAGE_WHITEBOARD, MAINPA
GE_MEDIASHARE};
```

```
//切换功能页
MAIN_PAGE_TYPE mainPage = MAINPAGE_VIDEOWALL;
Login.Instance.CRVideo.Meeting.switchToPage((int)mainPage, "");
```

```
//关联相关委托
Login.Instance.CRVideo.Meeting.notifySwitchToPage += new ICloudroomVideoMeetingEvents
_notifySwitchToPageEventHandler(notifySwitchToPage);
```

```
//当他人切换功能页时，收到通知消息
private void notifySwitchToPage(object sender, ICloudroomVideoMeetingEvents_notifySwi
tchToPageEvent e)
{
    switch (e.p_mainPage)
    {
        case (int)MAIN_PAGE_TYPE.MAINPAGE_VIDEOWALL:
            //...
            break;
        case (int)MAIN_PAGE_TYPE.MAINPAGE_SHARE:
            //...
            break;
        case (int)MAIN_PAGE_TYPE.MAINPAGE_WHITEBOARD:
           //...
           //p_subPageID 大功能内部的小功能页
            break;
        case (int)MAIN_PAGE_TYPE.MAINPAGE_MEDIASHARE:
           //...
            break;
        default:
            // 错误的功能页
            break;
    }
}
```

队列管理

> 利用队列功能，实现用户分配。使用队列时通常会有两种角色，坐席和客户，队列模块把排队的客户分配给某个服务队列的坐席。
>
> 组件介绍请参考 队列管理组件
>
> 1. 初始化队列，获取队列数据
> 2. 坐席队列操作
> 3. 坐席请求用户
> 4. 系统自动给坐席分配用户

> 5. 客户排队操作

1.初始化队列，获取队列数据

在登录成功后初始化队列数据

- 相关API请参考 初始化队列，初始化队列结果，查询所有队列
- 相关结构定义请参考 会话信息，队列信息

```
//关联队列初始化结果委托
CRVideo.Queue.initQueueDatRslt += new ICloudroomQueueEvents_initQueueDatRsltEventHand
ler(initQueueDatRslt);
```

```
//可在登录成功后初始化队列数据
private void loginSuccess(object sender, ICloudroomVideoMgrEvents_loginSuccessEvent e
)
{
    CRVideo.Queue.initQueueDat("");  //初始化专家坐席用户队列
}
```

```
//队列初始化响应
private void initQueueDatRslt(object sender, ICloudroomQueueEvents_initQueueDatRsltEv
ent e)
{
    if (e.p_sdkErr != 0)
    {
        // "队列初始化出错，请重新登录"
        return;
    }
    //队列初始化成功后获取上一次意外结束的视频会话信息，如果存在，则可以选择恢复会话
    VideoSessionInfo sessionInfo = JsonConvert.DeserializeObject<VideoSessionInfo>(CR
Video.Queue.getSessionInfo());
    if (sessionInfo.callID != "" && sessionInfo.duration > 0)
    {
        if (MessageBox.Show("是否恢复意外关闭的视频会话？", "提示", MessageBoxButton.YesNo)
 == MessageBoxResult.Yes)
        {
            //进入之前的会话
        }
        else
        {
            //结束上次的会话，准备新的会话
        }
    }
    //取到所有的队列信息
    List<QueueInfo> queues = JsonConvert.DeserializeObject<List<QueueInfo>>(CRVideo.Q
ueue.getAllQueueInfo());
    //选择坐席加载队列信息还是客户加载信息
```

```
    //ServicesShow(queues);
    // or
    //ClientsShow(queues);
}
```

2.坐席队列操作

坐席角色开始和停止服务队列，以及操作后队列状态的变化

- 相关API请参考 开始/停止服务队列，开始/停止队列服务结果
- 相关结构定义请参考 队列状态

```
//开始服务队列
Login.Instance.CRVideo.Queue.startService(queID, "");
```

```
//停止服务队列
Login.Instance.CRVideo.Queue.stopService(queID, "");
```

```
//获取服务的所有队列
string[] queIDs = Login.Instance.CRVideo.Queue.getServingQueues().Split('\n');
```

```
//关联相关委托
Login.Instance.CRVideo.Queue.startServiceRslt += new ICloudroomQueueEvents_startServi
ceRsltEventHandler(startServiceRslt);
Login.Instance.CRVideo.Queue.stopServiceRslt += new ICloudroomQueueEvents_stopService
RsltEventHandler(stopServiceRslt);

Login.Instance.CRVideo.Queue.queueStatusChanged += new ICloudroomQueueEvents_queueSta
tusChangedEventHandler(queueStatusChanged);
```

```
//开始队列服务结果
public void startServiceRslt(object sender, ICloudroomQueueEvents_startServiceRsltEve
nt e)
{
    //
}
```

```
//通知队列服务结果
public void stopServiceRslt(object sender, ICloudroomQueueEvents_stopServiceRsltEvent
 e)
{
    //
}
```

```
//队列状态变化通知
public void queueStatusChanged(object sender, ICloudroomQueueEvents_queueStatusChange
dEvent e)
{
    QueueStatus state = JsonConvert.DeserializeObject<QueueStatus>(e.p_jsonQueStatus)
;
}
```

3.坐席请求用户

在设置DND免打扰状下态，系统不再自动分配，需要手动申请用户

- 相关API请参考 免打扰，设置免打扰结果，请求分配用户，请求分配用户结果

```
Login.Instance.CRVideo.Queue.reqAssignUserRslt += new ICloudroomQueueEvents_reqAssign
UserRsltEventHandler(reqAssignUserRslt);
```

```
//设置免打扰状态，关掉系统自动推送
Login.Instance.CRVideo.Mgr.setDNDStatus(1, "");
```

```
//手动请求一个用户
Login.Instance.CRVideo.Queue.reqAssignUser("");
```

```
//请求用户的结果
public void reqAssignUserRslt(object sender, ICloudroomQueueEvents_reqAssignUserRsltE
vent e)
{
    if (e.p_sdkErr == (int)VCALLSDK_ERR_DEF.VCALLSDK_NOERR)
    {
        UserInfo user = JsonConvert.DeserializeObject<UserInfo>(e.p_jsonUsr);
        // 请求用户成功
    }
    else if(e.p_sdkErr == (int)VCALLSDK_ERR_DEF.VCALLSDK_QUE_NOUSER)
    {
        // 队列中没有排队人员
    }
    else
    {
       // 手动请求用户失败，代码：e.p_sdkErr
    }
}
```

```
//取消免打扰，开启系统自动推送
Login.Instance.CRVideo.Mgr.setDNDStatus(0, "");
```

4.系统自动给坐席分配用户

系统自动分配的用户在坐席还未选择接受或拒绝时，系统可以撤回分配

- 相关API请参考 自动分配用户通知，接受/拒绝分配的用户，自动分配用户被取消

```
Login.Instance.CRVideo.Queue.autoAssignUser += new ICloudroomQueueEvents_autoAssignUs
erEventHandler(autoAssignUser);
Login.Instance.CRVideo.Queue.cancelAssignUser += new ICloudroomQueueEvents_cancelAssi
gnUserEventHandler(cancelAssignUser);
Login.Instance.CRVideo.Queue.responseAssignUserRslt += new ICloudroomQueueEvents_resp
onseAssignUserRsltEventHandler(responseAssignUserRslt);
```

```
//系统自动分配用户通知
public void autoAssignUser(object sender, ICloudroomQueueEvents_autoAssignUserEvent e
)
{
    UserInfo user = JsonConvert.DeserializeObject<UserInfo>(e.p_jsonUsr);
    if(MessageBox.Show("是否接受系统分配的用户:" + user.name +" ?", "提示", MessageBoxButt
on.YesNo) == MessageBoxResult.Yes)
    {
        Login.Instance.CRVideo.Queue.acceptAssignUser(user.queID, user.userID, "");
        // do something with this user...
    }
    else
    {
        Login.Instance.CRVideo.Queue.rejectAssignUser(user.queID, user.usrID, "");
    }
}
```

```
//接受或拒绝分配的用户操作响应
public void responseAssignUserRslt(object sender, ICloudroomQueueEvents_responseAssig
nUserRsltEvent e)
{
    //
}
```

```
//系统撤回分配此用户通知
public void cancelAssignUser(object sender, ICloudroomQueueEvents_cancelAssignUserEve
nt e)
{
    //取消分配用户的弹框
}
```

5.客户排队操作

客户选择一个队列进行排队，每次只能排一个队列

- 相关API请参考 开始/停止排队，开始/停止排队操作结果，队列状态变化，排队信息变化
- 相关结构定义请参考 队列状态，排队信息

```
//关联相关委托
Login.Instance.CRVideo.Queue.startQueuingRslt += new ICloudroomQueueEvents_startQueuingRsltEventHandler(startQueuingRslt);
Login.Instance.CRVideo.Queue.stopQueuingRslt += new ICloudroomQueueEvents_stopQueuingRsltEventHandler(stopQueuingRslt);

Login.Instance.CRVideo.Queue.queueStatusChanged += new ICloudroomQueueEvents_queueStatusChangedEventHandler(queueStatusChanged);
Login.Instance.CRVideo.Queue.queuingInfoChanged += new ICloudroomQueueEvents_queuingInfoChangedEventHandler(queuingInfoChanged);
```

```
//开始排队
int queID = 1;
Login.Instance.CRVideo.Queue.startQueuing(queID, "");
```

```
//停止排队
Login.Instance.CRVideo.Queue.stopQueuing("");
```

```
//开始排队结果
public void startQueuingRslt(object sender, ICloudroomQueueEvents_startQueuingRsltEvent e)
{
    if (e.p_sdkErr != 0) //开始排队失败
    {
        // ....
    }
}
```

```
//停止排队结果
public void stopQueuingRslt(object sender, ICloudroomQueueEvents_stopQueuingRsltEvent e)
{
    //……
}
```

```
//队列状态变化通知
public void queueStatusChanged(object sender, ICloudroomQueueEvents_queueStatusChangedEvent e)
{
```

```
    QueueStatus state = JsonConvert.DeserializeObject<QueueStatus>(e.p_jsonQueStatus)
;
}
```

```
//排队信息更新
public void queuingInfoChanged(object sender, ICloudroomQueueEvents_queuingInfoChange
dEvent e)
{
    QueuingInfo queuingInfo = JsonConvert.DeserializeObject<QueuingInfo>(e.p_queuingI
nfo);
}
```

呼叫他人

> 实现用户到用户的呼叫，以此来实现会话信息的分发以及相关信息的传递
>
> 1. 主叫
> 2. 被叫

1.主叫

呼叫发起方

- 相关API请参考 开始呼叫，挂断呼叫，开始呼叫结果，挂断呼叫结果，通知呼叫被对方接受/拒绝

```
//关联相关委托
Login.Instance.CRVideo.Mgr.callSuccess += new ICloudroomVideoMgrEvents_callSuccessEve
ntHandler(callSuccess);
Login.Instance.CRVideo.Mgr.callFail += new ICloudroomVideoMgrEvents_callFailEventHand
ler(callFailed);

Login.Instance.CRVideo.Mgr.notifyCallAccepted += new ICloudroomVideoMgrEvents_notifyC
allAcceptedEventHandler(notifyCallAccepted);
Login.Instance.CRVideo.Mgr.notifyCallRejected += new ICloudroomVideoMgrEvents_notifyC
allRejectedEventHandler(notifyCallRejected);
```

```
//开始呼叫，meetObj为之前创建的会议对象字符串
string userID = "user_000007";
string callID = Login.Instance.CRVideo.Mgr.call(userID, meetObj, "", "");
```

```
//挂断呼叫
Login.Instance.CRVideo.Mgr.hungupCall(callID, "", "");
```

```
//呼叫成功发出，等待对方响应
public void callSuccess(object sender, ICloudroomVideoMgrEvents_callSuccessEvent e)
```

```
{
    //
}
```

```
//呼叫发出失败
public void callFailed(object sender, ICloudroomVideoMgrEvents_callFailEvent e)
{
    // 呼叫失败，代码：  e.p_sdkErr
}
```

```
//我的呼叫被对方接受，得到会议对象，可以进入会议
public void notifyCallAccepted(object sender, ICloudroomVideoMgrEvents_notifyCallAcce
ptedEvent e)
{
    MeetObj meet = JsonConvert.DeserializeObject<MeetObj>(e.p_meetObj);
    Login.Instance.CRVideo.Meeting.enterMeeting(meet.ID, meet.pswd, MyUserId, MyNickn
ame, "");
    //打开会话界面……
}
//我的呼叫被对方拒绝
public void notifyCallRejected(object sender, ICloudroomVideoMgrEvents_notifyCallReje
ctedEvent e)
{
    //被拒绝了 o.o ……
}
```

2.被叫

被呼叫方

- 相关API请参考 通知有人呼入，接受/拒绝他人的呼叫，接受/拒绝他人呼叫结果

```
//关联委托
Login.Instance.CRVideo.Mgr.notifyCallIn += new ICloudroomVideoMgrEvents_notifyCallInE
ventHandler(notifyCallIn);
```

```
//有呼叫进入
public void notifyCallIn(object sender, ICloudroomVideoMgrEvents_notifyCallInEvent e)
{
    if(/*接受呼叫，  进入会议*/)
    {
        Login.Instance.CRVideo.Mgr.acceptCall(e.p_callID, e.p_meetObj, "", "");
        MeetObj meet = JsonConvert.DeserializeObject<MeetObj>(e.p_meetObj);
        Login.Instance.CRVideo.Meeting.enterMeeting(meet.ID, meet.pswd, MyUserId, MyN
ickname, "");
        //打开会话界面……
```

```
    }
    else //拒绝对方的呼叫
    {
        Login.Instance.CRVideo.Mgr.rejectCall(e.p_callID, "", "", "");
    }
}
```

透明传输

> 独立于会话的传输功能，对SDK透明，发送对象必须要先成功登录
>
> 1. 发送命令、文本、文件
> 2. 收到命令、数据、文件

1.发送命令、文本、文件

小数据走命令接口，大数据走文本接口，命令的发送不可以被取消，也没有进度通知

- 相关API请参考 发送命令/数据/文件，取消发送，发送命令/数据/文件结果，发送进度，取消发送结果

```
//发送命令和数据
private void btnSendCmd_Click(object sender, RoutedEventArgs e)
{
    string userID = "user_000008";
    byte[] cmdBytes = Encoding.Default.GetBytes(txtCmdData.Text.Trim());
    //小数据走sendcmd接口，大数据走sendbuffer接口
    if (0 < cmdBytes.Length < 50000)
    {
        Login.Instance.CRVideo.Mgr.sendCmd(userID, Convert.ToBase64String(cmdBytes));
    }
    else
    {
        Login.Instance.CRVideo.Mgr.sendBuffer(userID, Convert.ToBase64String(cmdBytes
));
    }
}
```

```
//发送文件
private void sendFile()
{
    using (FileStream stream = new FileInfo(mSelectedFile).OpenRead())
    {
        label_sendBuffer_desc.Text = "文件大小: " + stream.Length / 1024 + "KB";
    }
    string userID = "user_000022";
    mFileTaskID = (string)Login.Instance.CRVideo.Mgr.sendFile(userID, mSelectedFile);
}
```

```
//取消发送
Login.Instance.CRVideo.Mgr.cancelSend(mFileTaskID);
```

```
//关联相关事件委托
Login.Instance.CRVideo.Mgr.sendCmdRlst += new ICloudroomVideoMgrEvents_sendCmdRlstEve
ntHandler(sendCmdRlst);
Login.Instance.CRVideo.Mgr.sendBufferRlst += new ICloudroomVideoMgrEvents_sendBufferR
lstEventHandler(sendBufferRlst);
Login.Instance.CRVideo.Mgr.sendFileRlst += new ICloudroomVideoMgrEvents_sendFileRlstE
ventHandler(sendFileRlst);

Login.Instance.CRVideo.Mgr.cancelSendRlst += new ICloudroomVideoMgrEvents_cancelSendR
lstEventHandler(cancelSendRlst);
Login.Instance.CRVideo.Mgr.sendProgress += new ICloudroomVideoMgrEvents_sendProgressE
ventHandler(sendProgress);
```

```
//发送大数据和文件的进入通知
private void sendProgress(object sender, ICloudroomVideoMgrEvents_sendProgressEvent e
)
{
    string text = "总大小: " + e.p_totalLen + ", 已发送" + e.p_sendedLen;
    //发完了
    if (e.p_sendedLen == e.p_totalLen)
    {
        //发送成功
    }
}
```

```
//发送命令结果
private void sendCmdRlst(object sender, ICloudroomVideoMgrEvents_sendCmdRlstEvent e)
{
    if (e.p_sdkErr != 0)
    {
        //发送命令数据失败: e.p_sdkErr
    }
}
```

```
//发送数据结果
private void sendBufferRlst(object sender, ICloudroomVideoMgrEvents_sendBufferRlstEve
nt e)
{
    if (e.p_sdkErr != 0)
    {
        //发送失败: e.p_sdkErr
    }
}
```

```
//发送文件结果
private void sendFileRlst(object sender, ICloudroomVideoMgrEvents_sendFileRlstEvent e
)
{
    if (e.p_sdkErr != 0)
    {
        mBufferTaskID = "";
        //发送失败: e.p_sdkErr
    }
}
```

```
//取消发送的结果
private void cancelSendRlst(object sender, ICloudroomVideoMgrEvents_cancelSendRlstEve
nt e)
{
    if (e.p_sdkErr != 0)
    {
       //取消发送失败"
    }
}
```

2.收到命令、数据、文件

收到别人发送数据的通知

- 相关API请参考 通知有命令/数据/文件发来

```
Login.Instance.CRVideo.Mgr.notifyCmdData += new ICloudroomVideoMgrEvents_notifyCmdDat
aEventHandler(notifyCmdData);
Login.Instance.CRVideo.Mgr.notifyBufferData += new ICloudroomVideoMgrEvents_notifyBuf
ferDataEventHandler(notifyBufferData);
Login.Instance.CRVideo.Mgr.notifyFileData += new ICloudroomVideoMgrEvents_notifyFileD
ataEventHandler(notifyFileData);
```

```
//收到命令
private void notifyCmdData(object sender, ICloudroomVideoMgrEvents_notifyCmdDataEvent
 e)
{
    byte[] cmdBytes = Convert.FromBase64String(e.p_data);
    string text = "来自[" + e.p_sourceUserId + "]的命令:" + Encoding.Default.GetString(
cmdBytes);
}
```

```
//收到大数据
private void notifyBufferData(object sender, ICloudroomVideoMgrEvents_notifyBufferDat
aEvent e)
```

```
{
    byte[] byteArray = Convert.FromBase64String(e.p_data);
    string text = "来自[" + e.p_sourceUserId + "]的数据" + Encoding.Default.GetString(b
yteArray);
}
```

```
//收到文件
private void notifyFileData(object sender, ICloudroomVideoMgrEvents_notifyFileDataEve
nt e)
{
    string text = "收到" + e.p_sourceUserId + "发来的文件: " + e.p_orgFileName + ", 临时
存放位置: " + e.p_tmpFile;
}
```

基础组件

> CloudroomVideoSDK 是基础控件，负责整个SDK的初始化和反初始化操作。 单例组件，整个程序的生命过程中只能有有一个实例。

接口函数

版本

```
string Version
```

- 功能 获取sdk的版本信息

> 只读属性，在init之前可用

SDK安装目录

```
string sdkPath
```

- 功能 获取sdk所在的目录

> 只读属性， 在init之前可用

设置SDK参数

```
void setSDKParams(string jsonStr);
```

- 功能 设置SDK参数
- 返回值 无
- 参数
  - jsonStr json格式字符串, 如：{ "NoCall" : 1 }，可用参数见下方表格

> 这个接口在init之前调用有效，具体可用参数如下：

参数名	取值	功能
NoCall	0/1 (缺省:0)	设为1可加快登录速度（适合于无呼叫需求的业务，如：临柜双录业务、链接入会业务）
NoQueue	0/1 (缺省:0)	设为1可加快登录速度（适合于不使用sdk的排队功能的业务，如：临柜双录业务、链接入会业务）
NoMediaDatToSvr	0/1 (缺省:0)	设为1可加快登录速度（适合于没有实时音视频到服务器的业务，如：临柜双录业务）
Timeout	（缺	网络通信超时时间，单位是毫秒

Timeout	省:90000)	网络通信超时时间，单位是毫秒

日志

```
void enableLog2File(int bEnable)
```

- 功能 设置是否开启日志
- 返回值 无
- 参数
  - bEnable 1表示开启日志，0表示不开启日志

> SDK中日志功能默认是打开的,这个接口在init之前或者之后均可调用

初始化 / 反初始化

```
CRVIDEOSDK_ERR_DEF init_2(string sdkFilePath)
```

- 功能 SDK初始化
- 返回值 CRVIDEOSDK_NOERR为成功，否则是失败代码
- 参数
  - sdkFilePath SDK配置、临时文件存放位置，可为空

> 程序开始时init，程序结束时uninit，整个程序的生命周期中只进行一次初始化和反初始化

```
void uninit()
```

- 功能 SDK反初始化
- 返回值 无
- 参数 无

服务器地址

```
string serverAddr
```

- 功能 设置和获取服务器地址

> 读写属性，支持单个服务器地址（如：www.cloudroom.com）或多个服务器地址串（如：www.cloudroom.com:8080;183.60.47.52:8080;）

文件上传 / 下载速度控制

> SDK默认不开启流控。目前对文件上传控制的功能有：CloudroomVideoMeeting录制文件上传、CloudroomVideoMeeting网盘文件、CloudroomHttpFileMgr文件传输。

```
void setFileUploadRate(int maxBytePerSec)
```

- 功能 文件上传的流量控制
- 返回值 无
- 参数
  - maxBytePerSec 每秒上传的最大字节数。-1代表不进行流控

```
void setFileDownloadRate(int maxBytePerSec)
```

- 功能 文件下载的流量控制
- 返回值 无
- 参数
  - maxBytePerSec 每秒上传的最大字节数。-1代表不进行流控

会议管理组件

> CloudroomVideoMgr 提供登录、呼叫、会议创建销毁、透明传输等功能 单例组件，整个程序的生命过程中只能有有一个实例。

接口函数

登陆/注销

```
void login(string authAcnt, string authPswd, string nickName, string privAcnt, string
 privAuthCode, string cookie)
```

- 功能 登陆服务器
- 返回值 无
- 参数
  - authAcnt 云屋鉴权帐号
  - authPswd 云屋鉴权MD5密码
  - nickName 昵称
  - privAcnt 自定义帐号，不需要时传空字符串
  - privAuthCode 自定义用户帐号验证信息，缺省传空字符串（不验证），如需验证则需要SDK服务端连接客户服务器进行验证。
  - cookie 自定义数据 (在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 loginSuccess, loginFail

```
void logout()
```

- 功能 注销本次登陆
- 返回值 无
- 参数 无

> 注销后如果没有执行反初始化可再次登陆

设置免打扰

```
void setDNDStatus(int DNDStatus, string cookie)
```

- 功能 设置免打扰状态
- 返回值 无
- 参数
  - DNDStatus 0代表关闭免打扰， 其它值代表开启免打扰，含义自由定义
  - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串

- 回调函数 setDNDStatusSuccess, setDNDStatusFail

> 如果是使用到了SDK的队列功能，则入会后需调用此接口，设置自己的为免打扰状态，防止系统再次推送自己

获取用户状态信息列表

```
void getUserStatus(string cookie)
```

- 功能 将获取企业下所有用户在线状态（包括呼叫会议状态、免打扰状态）
- 返回值 无
- 参数
  - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 getUserStatusSuccess, getUserStatusFail

开启/关闭用户状态推送

```
void startUserStatusNotify(string cookie)
```

- 功能 开启用户的状态推送
- 返回值 无
- 参数
  - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 startUserStatusNotifyRslt

> 说明：开启后，企业下所有用户状态有变化时(包括呼叫会议状态、免打扰状态)，都会收到通知。开启后，用户量越大消息量越大，所以请按需开启； 在startUserStatusNotify前，应该先通过getUserStatus获取所有用户状态。

```
void stopUserStatusNotify (string cookie)
```

- 功能 关闭用户的状态推送
- 返回值 无
- 参数
  - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 stopUserStatusNotifyRslt

创建/销毁视频会议

```
void createMeeting(string meetSubject, int createPswd, string cookie)
```

- 功能 创建视频会议
- 返回值 无
- 参数

    - meetSubject 会议主题（字符长度最大值50）
    - createPswd 是否创建会议密码（0：会议无密码，1：密码由系统自动生成）
    - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 createMeetingSuccess, createMeetingFail

```
void destroyMeeting(int meetID, string cookie)
```

- 功能 销毁视频会议
- 返回值 无
- 参数
    - meetID 会议号
    - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 destroyMeetingRslt

获取视频会议列表

```
void getMeetings(string cookie)
```

- 功能 获取会议列表
- 返回值 无
- 参数
    - cookie 自定义数据(在响应消息中回传给调用者)，不需要时传空字符串
- 回调函数 getMeetingsSuccess, getMeetingsFail

开始呼叫

```
string call(string calledUserID, string meetObj, string usrExtDat, string cookie)
```

- 功能 发起呼叫，邀请用户参加视频会话
- 返回值 本次呼叫标识码（呼叫ID）
- 参数
    - calledUserID 被叫用户的账户ID
    - meetObj 会议信息，json结构体定义请参见MeetInfoObj，空时代表无会议信息，应由被叫创建或提供会议信息
    - usrExtDat 自定义扩展参数
    - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 callSuccess, callFail

> 呼叫时，对方迟迟不响应，30秒后系统自动结束呼叫

接受/拒绝他人的呼叫

```
void acceptCall(string callID, string meetObj, string usrExtDat, string cookie)
```

- 功能 接受对方发起的视频请求，开始进入视频会话
- 返回值 无
- 参数
  - callID 呼叫标识码
  - meetObj 会议信息，json结构体请参见MeetInfoObj
  - usrExtDat 自定义扩展参数
  - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 acceptCallSuccess, acceptCallFail

```
void rejectCall(string callID, string usrExtDat, string cookie)
```

- 功能 拒绝对方的视频请求
- 返回值 无
- 参数
  - callID 呼叫标识码
  - usrExtDat 自定义扩展参数
  - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 rejectCallSuccess, rejectCallFail

挂断呼叫

```
void hungupCall(string callID, string usrExtDat, string cookie)
```

- 功能 挂断正在进行的视频呼叫或视频通话
- 返回值 无
- 参数
  - callID 呼叫标识码
  - usrExtDat 自定义扩展参数
  - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 hungupCallSuccess, hungupCallFail

邀请/取消邀请第三方入会

```
string callMoreParty(string called, string meetObj, string usrExtDat, string cookie)
```

- 功能 2方通话中再呼叫第3方
- 返回值 本次邀请标识码（邀请ID）
- 参数
  - called 被叫用户的账户ID
  - meetObj 当前会议信息，json结构体请参见MeetInfoObj
  - usrExtDat 自定义扩展参数
  - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 callMorePartyRslt

```
void  cancelCallMoreParty(string inviteID, string usrExtDat, string cookie)
```

- 功能 取消第3方呼叫
- 返回值 无
- 参数
  - inviteID 邀请标识码，邀请ID
  - usrExtDat 自定义扩展参数
  - cookie 自定义用户数据
- 回调函数 cancelCallMorePartyRslt

发送命令/数据/文件

```
string sendCmd(string targetUserId, string data)
```

- 功能 发送小块数据（50K以内）
- 返回值 分配的任务ID
- 参数
  - targetUserId 目标用户ID
  - data 发送的数据，base64编码
- 回调函数 sendCmdRlst

> 此接口不能被cancelSend, 一次性发送，不会有进度通知

```
string sendBuffer(string targetUserId, string data)
```

- 功能 发送大块数据
- 返回值 分配的任务ID
- 参数
  - targetUserId 目标用户ID
  - data 发送的数据，base64编码
- 回调函数 sendBufferRlst

> 分块发送，进度通知事件sendProgress, 调用cancelSend 取消发送

```
string sendFile(string targetUserId, string fileName)
```

- 功能 发送文件
- 返回值 分配的任务ID
- 参数
  - targetUserId 目标用户ID
  - fileName 需要发送的文件名
- 回调函数 sendFileRlst

> 分块发送，进度通知事件sendProgress，调用cancelSend取消发送

取消发送

```
void cancelSend(string taskID)
```

- 功能 取消数据发送
- 返回值 无
- 参数
  - taskID 任务ID
- 回调函数 cancelSendRlst

通知回调函数

登陆结果

```
void loginSuccess(string usrID,  string cookie)
```

- 功能 登录成功响应
- 参数
  - usrID 用户账户
  - cookie 自定义用户数据

```
void loginFail(int sdkErr,  string cookie)
```

- 功能 登录失败响应
- 参数
  - sdkErr 操作失败代码，定义见 CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

设置免打扰结果

```
void setDNDStatusSuccess(string cookie)
```

- 功能 客户端设置免打扰状态操作成功响应
- 参数
  - cookie 自定义用户数据

```
void setDNDStatusFail(int sdkErr, string cookie)
```

- 功能 客户端设置免打扰状态操作失败响应
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

获取所有用户在线状态结果

```
void getUserStatusSuccess (string usersStatus, string cookie)
```

- 功能 获取企业内所有用户在线状态成功响应
- 参数
  - usersStatus 用户在线状态信息列表， 详见json结构之UserStatusObjs
  - cookie 自定义用户数据

> 不在列表中的用户，代表离线状态

```
void getUserStatusFail(int sdkErr, string cookie)
```

- 功能 获取所有用户在线状态失败响应
- 参数
  - sdkErr 操作失败代码，数值参考 CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

开启/关闭用户状态推送结果

```
void startUserStatusNotifyRslt(int sdkErr, string cookie)
```

- 功能 启动用户状态推送响应
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

```
void stopUserStatusNotifyRslt(int sdkErr, string cookie)
```

- 功能 结束用户状态推送响应
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

用户状态变化通知

```
void ntifyUserStatus(string userStatus, string cookie)
```

- 功能 某个用户状态变化通知
- 参数
  - userStatus 单个用户在线状态信息， 详见json结构之UserStatusObj
  - cookie 自定义用户数据

通知登陆掉线

```
void lineOff(int sdkErr)
```

- 功能 SDK通知自己掉线
- 参数
  - sdkErr 掉线的错误代码，数值参考CRVIDEOSDK_ERR_DEF

创建/销毁会议结果

```
void createMeetingSuccess(string meetObj, string cookie)
```

- 功能 创建会议成功响应
- 参数
  - meetObj 会议信息, json结构体，请参见MeetInfoObj
  - cookie 自定义用户数据

```
void createMeetingFail(int sdkErr, string cookie)
```

- 功能 创建会议失败响应
- 参数
  - sdkErr 操作失败代码，数值参考 CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

```
void destroyMeetingRslt(int sdkErr, string cookie)
```

- 功能 结束会议响应
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

获取会议列表结果

```
void getMeetingsSuccess(string jsonMeetings, string cookie);
```

- 功能 获取当前会议列表响应
- 参数
  - jsonMeetings 会议列表json字符串，结构参考MeetInfoObjs
  - cookie 自定义用户数据

```
void getMeetingsFail(int sdkErr, string cookie);
```

- 功能 获取当前会议列表失败
- 参数
  - sdkErr 操作失败代码，数值参考定义CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

开始呼叫结果

```
void callSuccess(string callID, string cookie)
```

- 功能 呼叫他人操作成功响应
- 参数
  - callID 呼叫全局标识
  - cookie 自定义用户数据

```
void callFail( string callID, int sdkErr, string cookie)
```

- 功能 呼叫他人操作失败响应
- 参数
  - callID 呼叫全局标识
  - sdkErr 操作失败代码，数值参考定义 CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

接受/拒绝他人呼叫结果

```
void acceptCallSuccess(string callID, string cookie)
```

- 功能 接受他人呼叫操作成功响应
- 参数
  - callID 呼叫全局标识
  - cookie 自定义用户数据

```
void acceptCallFail( string callID, int sdkErr, string cookie)
```

- 功能 接受他人呼叫操作失败响应
- 参数
  - callID 呼叫全局标识
  - sdkErr 操作失败代码，数值参考定义CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

```
void rejectCallSuccess(string callID, string cookie)
```

- 功能 拒绝他人的呼叫成功响应
- 参数

    - callID 呼叫全局标识
    - cookie 自定义用户数据

```
void rejectCallFail(string callID, int sdkErr, string cookie)
```

- 功能 拒绝他人的呼叫失败响应
- 参数
    - callID 呼叫全局标识
    - sdkErr 操作失败代码，数值参考定义CRVIDEOSDK_ERR_DEF
    - cookie 自定义用户数据

挂断呼叫结果

```
void hungupCallSuccess(string callID, string cookie)
```

- 功能 挂断呼叫操作成功响应
- 参数
    - callID 呼叫全局标识
    - cookie 自定义用户数据

```
void hungupCallFail(string callID, int sdkErr, string cookie)
```

- 功能 拒绝他人呼叫操作失败响应
- 参数
    - callID 呼叫全局标识
    - sdkErr 操作失败代码，数值参考定义 CRVIDEOSDK_ERR_DEF
    - cookie 自定义用户数据

通知有人呼入

```
void notifyCallIn(string callID, string meetObj, string callerID, string usrExtDat)
```

- 功能 SDK通知自己被呼叫
- 参数
    - callID 呼叫全局标识
    - meetObj 会议信息（json结构体请参见MeetInfoObj，空时代表无会议信息，应由被叫创建或提供会议信息）
    - callerID 呼叫人员的标识ID
    - usrExtDat 自定义扩展参数

通知呼叫被对方接受/拒绝

```
void notifyCallAccepted(string callID, string usrExtDat, string meetObj)
```

- 功能 SDK通知自己视频呼叫被对方接受
- 参数
  - callID 呼叫全局标识
  - meetObj 会议信息(json结构体请参见MeetInfoObj，空时代表无会议信息)
  - usrExtDat 自定义扩展参数

```
void notifyCallRejected( string callID, int reason, string usrExtDat)
```

- 功能 SDK通知自己呼叫被对方拒绝
- 参数
  - callID 呼叫全局标识
  - sdkErr 呼叫被对方拒绝的原因代码，数值参考定义CRVIDEOSDK_ERR_DEF
  - usrExtDat 自定义扩展参数

通知呼叫被对方挂断

```
void notifyCallHungup(string callID, string usrExtDat)
```

- 功能 SDK通知呼叫被挂断
- 参数
  - callID 呼叫全局标识
  - usrExtDat 自定义扩展参数

邀请/取消邀请第三方结果

```
void callMorePartyRslt(string callID, int sdkErr, string cookie)
```

- 功能 第三方呼叫操作结果
- 参数
  - callID 邀请标识码，邀请ID
  - sdkErr 操作结果代码，数值参考CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

```
void cancelCallMorePartyRslt(string callID, int sdkErr, string cookie)
```

- 功能 取消第3方呼叫操作结果
- 参数
  - callID 邀请标识码，邀请ID
  - sdkErr 操作结果代码，数值参考CRVIDEOSDK_ERR_DEF
  - cookie 自定义用户数据

通知邀请状态

```
void notifyCallMorePartyStatus(string callID, int status)
```

- 功能 通知第3方呼叫状态改变
- 参数
  - callID 邀请标识码，邀请ID
  - status 状态，0振铃，1接通，2拒绝，3未应答，4挂断

发送命令/数据/文件结果

```
void sendCmdRlst(string taskID, int sdkErr, string cookie)
```

- 功能 发送数据时，SDK通知发送结果
- 参数
  - taskID 发送任务id
  - sdkErr 数值参考CRVIDEOSDK_ERR_DEF
  - cookie 用户自定义数据

```
void sendBufferRlst(string taskID, int sdkErr, string cookie)
```

- 功能 发送数据时，SDK通知发送结果
- 参数
  - taskID 发送任务id
  - sdkErr 数值参考CRVIDEOSDK_ERR_DEF
  - cookie 用户自定义数据

```
void sendFileRlst(string taskID, string fileName, int sdkErr, string cookie)
```

- 功能 发送文件时，SDK通知发送结果
- 参数
  - taskID 发送任务id
  - fileName 文件名
  - sdkErr 数值参考CRVIDEOSDK_ERR_DEF
  - cookie 用户自定义数据

发送进度

```
void sendProgress(string taskID, int sendedLen, int totalLen, string cookie)
```

- 功能 发送数据时，SDK通知发送进度
- 参数
  - taskID 发送任务id
  - sendedLen 已发送的数据长度

    - totalLen 需要发送的总长度
    - cookie 用户自定义数据

取消发送结果

```
void cancelSendRlst(string taskID, int sdkErr, string cookie)
```

- 功能 取消发送
- 参数
    - taskID 发送的任务ID
    - sdkErr 数值参考CRVIDEOSDK_ERR_DEF
    - cookie 用户自定义数据

通知有命令/数据/文件发来

```
void notifyCmdData(string sourceUserId, string data)
```

- 功能 SDK通知收到小块数据
- 参数
    - sourceUserId 数据来源
    - data 数组数据,base64编码

```
void notifyBufferData(string sourceUserId, string data)
```

- 功能 SDK通知收到大块数据
- 参数
    - sourceUserId 数据来源
    - data 数组数据,base64编码

```
void notifyFileData(string sourceUserId, string tmpFile, string orgFileName)
```

- 功能 SDK通知收到文件数据
- 参数
    - sourceUserId 数据来源
    - tmpFile 临时文件，不需要时，请移除或删除对应文件
    - orgFileName 源始文件名

> 收到的文件生成在系统临时目录下，请尽快移走对应文件

通知发来的数据/文件被取消

```
void notifyCancelSend(string sendId)
```

- 功能 SDK通知取消发送文件数据
- 参数
  - sendId 任务id

视频会议组件

> CloudroomVideoMeeting 是会议核心控件，实现通话建立、音频采集播入、视频采集编解码、屏幕共享、录制、影音播放等。 单例组件，整个程序的生命过程中只能有有一个实例。

接口函数

进入/退出/结束会议

```
void enterMeeting(int meetID, string pswd , string userID, string nickName, string cookie)
```

- 功能 使用会议ID和密码（可为空）进入指定的视频会议
- 返回值 无
- 参数
  - meetID 视频会议ID
  - pswd 本次会议中的密码（在创建会议时指定参数）
  - userID 用户账号
  - nickName 用户昵称
  - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 enterMeetingRslt

```
void exitMeeting()
```

- 功能 离开会议
- 返回值 无
- 参数 无

> 调用此接口离开会话时，其他会话用户会收到userLeftMeeting的消息通知

```
void stopMeeting(int meetID, string cookie)
```

- 功能 结束会议
- 返回值 无
- 参数
  - meetID 会议id
  - cookie 自定义数据(在响应消息中回传给调用者)
- 回调函数 stopMeetingRslt

> 调用此接口结束会议时，其他会话用户会收到 meetingStopped

会议成员列表

```
string getAllMembers()
```

- 功能 获取所有用户的信息
- 返回值 json格式的字符串,详见MemberObjs说明
- 参数 无

会议成员信息

```
string getMemberInfo(string userID)
```

- 功能 获取某个用户的信息
- 返回值 json格式的字符串,详见MemberObj说明
- 参数
  - userID 用户ID

会议成员昵称

```
string getMemberNickName(string userID)
```

- 功能 获取某个用户的昵称
- 返回值 用户的昵称
- 参数
  - userID 用户ID

用户是否在会议中

```
int isUserInMeeting(string userID)
```

- 功能 判断某个用户是否在会议中
- 返回值 1：在，0：不在
- 参数
  - userID 用户的id

麦克风/扬声器列表

```
string getAudioMicNames()
```

- 功能 获取系统上的麦克风列表
- 返回值 麦克风设备列表，以’\n’分割;
- 参数 无

```
string getAudioSpkNames ()
```

- 功能 获取系统上的扬声器列表
- 返回值 扬声器列表，以’\n’分割;
- 参数 无

麦克风设置

```
void setAudioCfg(string jsonCfg);
```

- 功能 设置麦克风音频参数
- 返回值 无
- 参数
  - jsonCfg 音频参数json字符串，详见AudioCfgObj说明

获取麦克风设置

```
string getAudioCfg()
```

- 功能 获取音频参数
- 返回值 json格式的字符串，详见AudioCfgObj说明
- 参数 无

麦克风声音大小

```
int getMicEnergy(string userID)
```

- 功能 获取用户说话声音大小
- 返回值 音量大小（0~10）
- 参数
  - userID 用户的ID

开/关麦克风

```
void openMic(string userID)
```

- 功能 打开用户的麦克风
- 返回值 无
- 参数
  - userID 用户的ID

> 打开麦克风会触发音频状态变化的回调函数audioStatusChanged 新状态参数先会进入到AOPENING状态，等服务器处理后才会进入AOPEN状态，此时说话才能被采集到；

```
void closeMic(string userID)
```

- 功能 关闭用户的麦克风
- 返回值 无
- 参数
  - userID 用户的ID

> 关麦克风会触发音频状态变化的回调函数audioStatusChanged 新状态参数会变为ACLOSE 关麦操作是立即生效的，会立即停止采集；

麦克风状态

```
int getAudioStatus(string userID)
```

- 功能 获取用户的麦状态
- 返回值 麦克风设备状态，请参见ASTATUS定义
- 参数
  - userID 登录成功后分配的ID

麦克风音量

```
int micVolume
```

- 功能 设置、获取麦克风音量大小

> 读写属性，音量范围取值 0-255

扬声器音量

```
int speakerVolume
```

- 功能 设置、获取本地扬声器音量

> 读写属性，音量等级取值 0-255

扬声器静音

```
int speakerMute
```

- 功能 设置、获取播放是否静音

- 说明 对会议内所有声音生效

> 读写属性，取值 0:不静音, 1:静音

关闭所有人麦克风

```
void setAllAudioClose()
```

- 功能 关闭所有用户的麦克风
- 返回值 无
- 参数 无

> 调用此接口后会话内其他人的麦克风会关闭，同时收到消息audioStatusChanged

摄像头设备列表

```
string getAllVideoInfo(string userID)
```

- 功能 获取用户所有的摄像头信息
- 返回值 json格式的字符串,详见VideoInfoObjs说明
- 参数
  - userID 用户ID

视频设置

```
void setVideoCfg(string jsonCfg)
```

- 功能 设置摄像头参数
- 返回值 无
- 参数
  - jsonCfg json格式的字符串，详见VideoCfgOjb说明

会议内可观看摄像头列表

```
string getWatchableVideos()
```

- 功能 获取会议内所有可观看的摄像头
- 返回值 json格式的字符串,详见VideoIDObjs说明
- 参数 无

开/关摄像头

```
void openVideo(string userID)
```

```
```

- 功能 打开用户的摄像头，以便本地、远端显示视频图像
- 返回值 无
- 参数
  - userID 用户的ID
- 回调函数 openVideoDevRslt

```
void closeVideo(string userID)
```

- 功能 关闭用户的摄像头
- 返回值 无
- 参数
  - userID 用户的ID

> 调用打开和关闭麦克风接口都会触发对应用户的videoStatusChanged

视频状态

```
VSTATUS getVideoStatus(string userID)
```

- 功能 获取用户的摄像头状态
- 返回值 麦克风摄像头状态，请参见VSTATUS定义
- 参数
  - userID 用户的ID

获取/设置默认视频

```
int getDefaultVideo(string userID)
```

- 功能 获取指定用户的默认摄像头
- 返回值 摄像头ID
- 参数
  - userID 用户ID

> 如果用户没有摄像头，返回0；

```
void setDefaultVideo(string userID, long videoID)
```

- 功能 设置默认的摄像头
- 返回值 无
- 参数
  - userID 用户ID
  - videoID 摄像头ID

> videoID 应该从getAllVideoInfo返回值中获取。

获取/开关本地多摄像头

```
int getEnableMutiVideo(string userID)
```

- 功能 查询用户是否启用多摄像头
- 返回值 1 表示开启了，0 表示未开启
- 参数
  - userID 用户ID

```
void setEnableMutiVideo(string userID, int bEnable)
```

- 功能 设置用户是否启用多摄像头
- 返回值 无
- 参数
  - userID 用户ID
  - bEnable 1表示开启，0表示关闭

> 用于实现一个用户同时打开多个视频设备

摄像头图像数据

```
string getVideoImg(string userID, int videoID)
```

- 功能 获取指定用户的最新图像
- 返回值 json格式的字符串,详见VideoImgObj说明
- 参数
  - userID 用户ID
  - videoID 摄像头ID

屏幕共享配置

```
string getScreenShareCfg()
```

- 功能 获取当前屏幕共享配置
- 返回值 json格式的字符串,详见ScreenShareCfgObj说明
- 参数 无

```
void setScreenShareCfg(string jsonCfg)
```

- 功能 设置当前屏幕共享配置
- 返回值 无

- 参数
  - jsonCfg json格式的字符串,详见ScreenShareCfgObj说明

开始/停止屏幕共享

```
void startScreenShare()
```

- 功能 开启屏幕共享
- 返回值 无
- 参数 无
- 回调函数 startScreenShareRslt

```
void stopScreenShare ()
```

- 功能 停止屏幕共享
- 返回值 无
- 参数 无
- 回调函数 stopScreenShareRslt

屏幕共享图像数据

```
string getShareScreenDecodeImg()
```

- 功能 获取屏幕共享解码图像
- 返回值 json格式的字符串,详见VideoImgObj说明
- 参数 无

自定义的抓屏图像数据

```
void setCustomizeScreenImg(int format, int width, int height, string dat)
```

- 功能 设置自定义的抓屏图像数据
- 返回值 无
- 参数
  - format 参见VIDEO_FORMAT；
  - width 图像的宽度；
  - height 图像的高度；
  - dat 承载argb数据, base64编码；

> 说明：当前只支持VFMT_ARGB32格式；如果在收到notiyCatchScreen事件时，如当前无图像可送，可送空数据进去(width=0, height=0, dat为空)

屏幕共享状态

```
int isScreenShareStarted
```

- 功能 检查屏幕共享是否已开启

  > 只读属性，只读，0：没有开启，1：已经开启

屏幕可共享尺寸

```
int supportMaxScreenWidth
```

- 功能 支持的最大屏幕宽度

> 只读属性

```
int supportMaxScreenHeight
```

- 功能 支持的最大屏幕高度

> 只读属性

自定义屏幕共享抓取

```
int customizeCatchScreen
```

- 功能 设置、获取自定义抓屏功能

> 读写属性, 0：默认抓屏处理；1：自定议抓屏；设置自定义抓屏后，sdk需要图像时将产生notiyCatchScreen事件，使用者再通过setCustomizeScreenImg接口送入图像

赋予/收回远程屏幕控制权限

```
void giveCtrlRight(string userID)
```

- 功能 赋予控制权限
- 返回值 无
- 参数
  - userID 目标用户

> 调用此接口后，对方收到notifyGiveCtrlRight

```
void releaseCtrlRight(string userID)
```

- 功能 收回控制权限
- 返回值 无
- 参数

- userID 目标用户

> 调用此接口后，对方收到notifyReleaseCtrlRight

发送鼠标/键盘控制消息

```
void sendMouseCtrlMsg(int msgType, int key, int ptX, int ptY)
```

- 功能 发送鼠标控制消息
- 返回值 无
- 参数
  - msgType 鼠标事件类型
  - key 鼠标键类型
  - ptX 鼠标在屏幕中的横坐标
  - ptY 鼠标在屏幕中的纵坐标

> msgType 数值详见MOUSE_MSG_TYPE类型, key 数值详见MOUSE_KEY_TYPE类型

```
void sendKeyCtrlMsg(int keyMsgType, int vk, int bExtendedKey)
```

- 功能 发送键盘控制消息
- 返回值 无
- 参数
  - keyMsgType 键盘事件类型
  - vk 键盘虚拟键值
  - bExtendedKey 是否是扩展键值

> keyMsgType数值详见KEY_MSG_TYPE类型

录制内容配置

```
void setRecordVideos(string json)
```

- 功能 设置要录制的视频
- 返回值 无
- 参数
  - json格式的字符串,详见RecordContentObj说明

> 调用startRecording后，录制过程中配置有效

开始/停止录制

```
int startRecording(string json)
```

- 功能 开启录制

- 返回值 0 开启失败，1 开启成功
- 参数
  - json 录制参数, json格式的字符串,详见RecordCfgObj说明

```
void stopRecording()
```

- 功能 停止录制
- 返回值 无
- 参数 无

录制文件大小

```
int getRecFileSize()
```

- 功能 得到录制结果文件大小
- 返回值 文件大小
- 参数 无

录制时长

```
int getRecDuration()
```

- 功能 得到录制的时长
- 返回值 无
- 参数 无

是否加密录制文件

```
void setRecordFileEncrypt(int encrypt)
```

- 功能 设置本地生成的录制文件是否加密
- 返回值 无
- 参数
  - encrypt 1表示加密，0表示不加密

> 未加密的录制文件用其他播放器也可播放，加密后只能利用SDK回放功能才可播放。回放接口为playbackRecordFile

录制文件列表

```
string getAllRecordFiles()
```

- 功能 取得所有录制文件信息
- 返回值 json格式的字符串,详见RecordFileObjs
- 参数 无

录制列表添加/删除文件

```
int addFileToRecordMgr (string fileName, string filePath)
```

- 功能 添加本地文件到录制文件管理中
- 返回值 -1：本地文件不存在，0：成功，1：文件已经被添加过
- 参数
  - filename 文件名，不含路径
  - filePath 文件路径，不含文件名

> 第三方录制文件调用此接口后可进行本地回放和上传到视频服务器上，和自己录制的文件一样可以正常在线播放和下载

```
void removeFromFileMgr(string fileName)
```

- 功能　删除本地的录制文件，上传中的文件会被取消上传
- 返回值　无
- 参数
  - filename 文件名

> 已上传完成的服务器文件不受影响

上传/取消上传录制文件

```
void uploadRecordFile(string filename)
```

- 功能 上传文件在默认位置
- 返回值 无
- 参数
  - filename 文件名

```
void uploadRecordFile2(string filename, string svrPathFileName)
```

- 功能 上传文件到服务器指定位置
- 返回值 无
- 参数
  - filename 文件名
  - svrPathFileName 文件存放在服务器上的相对路径文件名（如/AA/BB/CC/test.mp4）

```
void cancelUploadRecordFile(string filename)
```

- 功能 取消上传中的录制文件
- 返回值 无
- 参数
  - filename 文件名

回放录制文件

```
void playbackRecordFile(string filename)
```

- 功能 回放录制文件
- 返回值 无
- 参数
  - filename 文件名

> 可创建影音控件显示录制内容，功能同接口startPlayMedia，如果录制文件被加密，则只能使用playbackRecordFile来回放。

设置/获取会话内主功能页

```
void switchToPage(int mainPage, string pageID)
```

- 功能 功能切换
- 返回值 无
- 参数
  - mainPage 功能类型，数值参见MAIN_PAGE_TYPE
  - pageID 子页面标识（如创建白板时返回的boardID）

```
int getCurrentMainPage()
```

- 功能 获取当前主功能区
- 返回值
  - mainPage 数值参见MAIN_PAGE_TYPE定义
- 参数 无

```
string getCurrentSubPage()
```

- 功能 获取当前子页面
- 返回值 返回pageID
- 参数 无

获取/设置会议内视频分屏模式

```
int getVideoWallMode()
```

- 功能 获取视频墙当前分屏模式
- 返回值 分屏模式，类型见VIDEO_LAYOUT_MODE定义
- 参数 无

```
void setVideoWallMode(int videoWallMode)
```

- 功能 设置视频墙分屏模式
- 返回值 无
- 参数
  - videoWallMode 分屏模式

> videoWallMode数值参见VIDEO_LAYOUT_MODE定义

主视频

```
string mainVideo
```

- 功能 获取、设置当前哪个用户为主视频

> 读写属性

创建/关闭电子白板

```
string createBoard(string title, int width, int height, int pageCount)
```

- 功能 创建电子白板
- 返回值 返回boardID
- 参数
  - title 白板名称
  - width 白板宽度
  - height 白板高度
  - pageCount 白板内有多个页（一般空白板1页，文档白板为实际页数）

> 其他参会者会收到notifyCreateBoard事件；同时后台会记录下白板数据，新入会者会收到notifyInitBoards事件 注意：创建完白板后，一定要及尽快调用initBoardPageDat初始化各页数据；如果需要所有参会者同步切到此白板，请调用switchToPage

```
void closeBoard(string boardID)
```

- 功能 关闭电子白板
- 返回值 无
- 参数
  - boardID 白板标识

> 其他参会者将收到notifyCloseBoard事件；同时后台会移除对应白板的所有信息；

初始化白板图元数据

```
void initBoardPageDat(string boardID, int boardPageNo, string imgID, string elemets)
```

- 功能 初始化白板指定页数据
- 返回值 无
- 参数
  - boardID 白板标识
  - boardPageNo 白板第几页（0:代表第一页）
  - imgID 白板的背景图片标识（空代表无背影图）
  - elemets 白板的初始图元（空代表无图元，一般在导入历史文件才用到）

> 1. imgID非空时, 代表背景的图片ID(建议为uuid)。（对应的文件应通uploadNetDiskFile上传到服务器；）
> 2. 其他参会者将收到notifyInitBoardPageDat事件；
> 3. 后台会记录下白板的页数据，在新用户入会时，也会收到notifyInitBoardPageDat事件

生成白板图元ID

```
string createElementID()
```

- 功能 创建一个符合SDK要求的图元id
- 返回值 图元id
- 参数 无

> 所有白板图元id，必须由此接口创建；（历史文件存储的图元id，在会议内不能再使用，应重新创建）

添加/修改/删除白板图元

```
string addBoardElement(string boardID, int boardPageNo, string element)
```

- 功能 添加图元信息
- 返回值 elementID图元标识
- 参数
  - boardID 白板标识
  - boardPageNo 白板的页序号(0为第一页)
  - element 图元信息，参见json格式之BoardElementObj

> 其他参会者会收到：notifyAddBoardElement事件同时后台会保存图元，新入会者会在notifyInitBoardPageDat中得到这些图元

string modifyBoardElement(string boardID, int boardPageNo, string element)

- 功能 修改图元信息
- 返回值 elementID图元标识
- 参数

    - boardID 白板标识
    - boardPageNo 白板的页序号(0为第一页)
    - element 图元信息，参见json格式之BoardElementObj

> 其他参会者会收到：notifyModifyBoardElement事件，同时后台会覆盖对应图元的数据，新入会者会在notifyInitBoardPageDat中得到这些图元

void delBoardElement(string boardID, int boardPageNo, string elementIDs)

- 功能 删除图元
- 返回值 无
- 参数
    - boardID 白板标识
    - boardPageNo 白板的页序号(0为第一页)
    - elementIDs 图元id列表，多值时，以“;”分隔，如：“id1;id2”

> 其他参会者会收到notifyDelBoardElement事件，同时后台会移除这些图元，新入会者会在notifyInitBoardPageDat中将不包含这些图元

设置白板鼠标热点

```
void setMouseHotSpot(string boardID, int boardPageNo, int x, int y)
```

- 功能 设置鼠标热点信息
- 返回值 无
- 参数
    - boardID 白板标识
    - boardPageNo 白板的页序号(0为第一页)
    - x 屏幕横坐标
    - y 屏幕纵坐标

会议网盘容量

```
void getNetDiskSummary();
```

- 功能 获取会议网盘的容量信息
- 返回值 无
- 参数 无
- 回调函数 getNetDiskSummaryRslt

获取网盘文件列表

```
void getNetDiskFileList();
```

- 功能 获取网盘用户共享文件列表，即：使用makeNetDiskFileID中参数fileType为0的生成的fileID上传的文

件

- 返回值 无
- 参数 无
- 回调函数 getNetDiskFileListRslt

生成网盘文件ID

```
string makeNetDiskFileID(int fileType, string newFileName);
```

- 功能 生成网盘文件全局唯一ID
- 返回值 网盘文件ID，带相对路径前缀的字符串
- 参数
  - fileType 文件类型，0用户共享文件，1程序使用文件
  - newFileName 传入的全局唯一文件名，建议带文件后缀

> fileType等于0时，为会议网盘共享文件，上传的文件可通过getNetDiskFileList获取到文件列表详情；
> fileType等于1时，用户程序内资源文件，如白板的背景图片，无法获取文件详情；两种文件类型都要调用uploadNetDiskFile和downloadNetDiskFile进行上传和下载

上传/下载/删除网盘文件

```
void uploadNetDiskFile(string fileID, string localFilePath)
```

- 功能 上传文件到网盘
- 返回值 无
- 参数
  - fileID 网盘文件ID
  - localFilePath 本地文件路径，含文件名

```
void downloadNetDiskFile(string fileID, string localFilePath)
```

- 功能 从网盘中下载文件
- 返回值 无
- 参数
  - fileID 网盘文件ID
  - localFilePath 本地文件路径，全路径

```
void deleteNetDiskFile(string fileID)
```

- 功能 删除网盘文件
- 返回值 无
- 参数
  - fileID 网盘文件ID

取消网盘文件操作

```
void cancleNetDiskFile(string fileID)
```

- 功能 取消网盘文件操作（上传/下载）
- 返回值 无
- 参数
  - fileID 网盘文件ID

暂停/继续网盘文件传输

```
void setNetDiskTransportPause(string fileID, bool bTranPause)
```

- 功能 设置网盘文件传输暂停或继续
- 返回值 无
- 参数
  - fileID 网盘文件ID
  - bTranPause 是否暂停

影音播放配置

```
void setMediaCfg (string jsonCfg)
```

- 功能 配置远程影音共享时，图像质量参数
- 返回值 无
- 参数
  - jsonCfg json格式的字符串，详见VideoCfgOjb说明

开始/暂停/停止影音播放

```
void startPlayMedia(string filename, int bLocPlay, int bPauseWhenFinished)
```

- 功能 播放影音文件
- 返回值 无
- 参数
  - filename 文件名，全路径
  - bLocPlay 是否仅仅本地播放（1:本地播放，0：会议内播放）
  - bPauseWhenFinished 是否播放完毕自动暂停在最后一帧

> 如果播放成功，其他人收到notifyMediaOpened，如果播放失败，请关注通知事件notifyMediaStop

```
void pausePlayMedia(int bPause)
```

- 功能 暂停播放影音
- 返回值 无
- 参数
  - bPause 0播放，1暂停

```
void stopPlayMedia()
```

- 功能 停止影音播放
- 返回值 无
- 参数 无

> 其他人收到notifyMediaStop

设置播放进度

```
void setMediaPlayPos(int pos)
```

- 功能 设置播放进度
- 返回值 无
- 参数
  - pos 设置播放位置，单位：毫秒

影音文件列表

```
string getAllFilesInMediaPath()
```

- 功能 取得影音文件夹下的所有可播放文件
- 返回值 文件名列表，以’\n’分割；
- 参数 无

> 影音文件夹位于方法init的第二个参数sdkFilePath，sdk会在此文件中建立media的子文件夹，即为影音文件夹

影音播放信息

```
string getMediaInfo()
```

- 功能 取得影音文件信息
- 返回值
  - json格式的字符串,详见MediaInfoObj说明
- 参数无

影音播放音量

```
int mediaVolume
```

- 功能 读取、设置影音播放的音量

> 读写属性，取值范围，音量等级（0-255）

获取影音图像数据

```
string getMediaImg(string userID)
```

- 功能 取得影音帧信息
- 返回值 json格式的字符串,详见VideoImgObj说明
- 参数
  - userID 用户id

开始/停止获取PCM音频数据

```
int startGetAudioPCM (int aSide,int getType, string jsonParam)
```

- 功能 开始获取语音pcm数据
- 返回值 整形数值，1：正常，0：失败
- 参数
  - aSide 声道类型，0:麦克风，1:扬声器
  - getType 获取方式，传0,1
    - 0 为回调方式， jsonParam可配置回调的数据大小(320-32000)，如: {"EachSize":320}；
    - 1 为文件保存， jsonParam可配置文件名,如: { "FileName" : "e:\test.pcm" }
  - jsonParam 参数字符串

> 1. 可同时启动获取"麦克风"、"扬声器"；
> 2. 一种声道，只能选一种获取方式，以最后的配置为准；
> 3. 采用回调方式后，将定期收到notifyAudioPCMDat事件；

```
void stopGetAudioPCM (int aSide);
```

- 功能 停止获取语音pcm数据
- 返回值 无
- 参数
  - aSide 声道类型，0麦克风，1扬声器

发送IM文本消息

```
string sendIMmsg(string text, string toUsrID, string cookie)
```

- 功能 发送IM消息
- 返回值 taskID 发送任务id
- 参数
  - text 发送的文本消息
  - toUsrID 目标用户，如果用户ID为空，消息发送给会议内所有用户
  - cookie 自定义用户数据
- 回调函数 sendIMmsgRlst

添加图片资源

```
void setPicResource(string picID, string jsonVal)
```

- 功能 将图片资源设置给sdk
- 返回值 无
- 参数
  - picID 资源唯一标识；（可以是guid，也可以序号方式）
  - jsonVal 资源内容，json格式，详见PicResourceObj说明。

> 注：如果jsonVal为空串，代表移除资源。

通知回调函数

进入/结束会议结果

```
void enterMeetingRslt(int sdkErr)
```

- 功能 自己进入会议的结果
- 参数
  - sdkErr 操作结果码, 0代表入会成功, 非0代表入会失败，取值参考CRVIDEOSDK_ERR_DEF

```
void stopMeetingRslt(int sdkErr)
```

- 功能 通知结束视频会议结果
- 参数
  - sdkErr 数值参考CRVIDEOSDK_ERR_DEF

有人进入/离开会议通知

```
void userEnterMeeting(string userID)
```

- 功能 某用户进入了会议
- 参数
  - userID 进入会议的用户id

```
void userLeftMeeting(string userID)
```

- 功能 某用户离开了会议
- 参数
  - userID 离开会议的用户id

会议掉线通知

```
void meetingDropped()
```

- 功能 通知从会议里掉线了，收到该通知后可以调用enterMeeting尝试重新入会
- 参数 无

> 如果用到了呼叫队列，掉线后不重新入会就必须调用hungupCall释放本次呼叫

会议被结束通知

```
void meetingStoped()
```

- 功能 会议已被结束
- 参数 无

网络状态变化通知

```
void netStateChanged(long level)
```

- 功能 网络变化通知
- 参数
  - level 网络状况等级(0~10，10分为最佳网络)

麦克风设备变化

```
void audioDevChanged()
```

- 功能 通知本地音频设备有变化
- 参数 无

麦克风状态变化

```
void audioStatusChanged(string userID, int oldStatus, int newStatus)
```

- 功能 用户的音频状态变化通知
- 参数
  - userID 会议中设备的所有者
  - oldStatus 旧状态，数值参考麦克风状态定义ASTATUS
  - newStatus 新状态，数值参考麦克风状态定义ASTATUS

麦克风声音变化

```
void micEnergyUpdate( string userID, int oldLevel, int newLevel)
```

- 功能 通知用户的说话声音强度更新
- 参数
  - userID 用户标识ID
  - oldLevel 原来的说话声音强度(0~10)
  - newLevel 现在的说话声音强度(0~10)

打开摄像头结果

```
void openVideoDevRslt(string devID, bool isSucceed)
```

- 功能 打开摄像头设备操作结果
- 参数
  - devID 摄像头设备ID
  - isSucceed 是否成功

视频状态变化

```
void videoStatusChanged(string userID, int oldStatus, int newStatus)
```

- 功能 用户的视频状态变化通知
- 参数
  - userID 会议中设备的所有者
  - oldStatus 旧状态，数值参考摄像头状态定义
  - newStatus 新状态，数值参考摄像头状态定义

摄像头设备变化

```
void videoDevChanged(string userID)
```

- 功能 通知用户的视频设备有变化
- 参数
  - userID 设备变化的用户ID

通知视频图像数据

```
void notifyVideoData(string userID, int videoID, int frameTime)
```

- 功能 通知用户有新的视频数据
- 参数
  - userID 用户标识ID
  - videoID 用户的摄像头ID
  - frmTime 图像的创建时间，可用作时间戳

> 收到此通知消息后，可通过getVideoImg获取图像显示；如果之前显示的帧时间和此值一致，说明此帧已显示过，直接忽略即可； 如果使用了成员视频UI显示组件，不再需要自己关注此事件并进行显示处理。

默认视频设备变化

```
void videoDevChanged(string userID, int videoID)
```

- 功能 通知用户的视频默认设备有变化
- 参数
  - userID 设备变化的用户ID
  - videoID 默认设备id

录制文件上传/取消上传错误

```
void uploadRecordFileErr(int sdkErr)
```

- 功能 上传录制文件错误通知
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF

录制错误通知

```
void recordErr(int sdkErr)
```

- 功能 录制异常，录制将自动停止
- 参数
  - sdkErr 错误信息，数值参考CRVIDEOSDK_ERR_DEF

录制状态变化通知

```
void recordStateChanged(int state)
```

- 功能 录制状态更改通知
- 参数
  - state 录制状态，数值请参考定义RECORD_STATE

开始/停止屏幕共享操作结果

```
void startScreenShareRslt(int sdkErr)
```

- 功能 开启屏幕共享的响应事件
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF

```
void stopScreenShareRslt (int sdkErr)
```

- 功能 停止屏幕共享的响应事件
- 参数
  - sdkErr 操作失败代码，数值参考CRVIDEOSDK_ERR_DEF

开始/停止屏幕共享通知

```
void notifyScreenShareStarted()
```

- 功能 通知他人开启了屏幕共享
- 参数 无

```
void notifyScreenShareStopped()
```

- 功能 通知他人停止了屏幕共享
- 参数 无

通知屏幕共享图像数据

```
void notifyScreenShareData(string userID, string datInfo)
```

- 功能 通知对端屏幕图像有变化
- 参数
  - userID 对端用户ID
  - datInfo 屏幕大小、变化区域信息

> datInfo，json格式，结构如下示例：

```
{"screenWidth":1366, "screenHeight":768, "changeLeft":50, "changeTop":50, "changeWidt
h":100 ,"changeHeight":100}
```

> 如果使用了屏幕共享UI显示组件，不再需要自己关注此事件并进行显示处理。

自定义抓屏通知

```
void notiyCatchScreen()
```

- 功能 自定义抓屏时，通知使用者抓屏
- 参数 无

> 在收到通知时， 一定要及时setCustomizeScreenImg，如果没图像时，可以先送入空图像；

通知赋予/收回屏幕共享操作权限

```
void notifyGiveCtrlRight(string operId, string targetId)
```

- 功能 通知被赋予了远程控制权限
- 参数
  - operId 操作的用户id
  - targetId 控制权限给予了谁

```
void notifyReleaseCtrlRight(string operId, string targetId)
```

- 功能 通知被收回了远程控制权限
- 参数
  - operId 操作的用户id
  - targetId 收回了谁的控制权限

通知屏幕共享区域变化

```
void notifyShareRectChanged(int x, int y, int w, int h)
```

- 功能 通知屏幕共享区域变化
- 参数
  - x，y 位置
  - w 宽度
  - h 高度

发送IM消息结果

```
void sendIMmsgRlst(string taskID, int sdkErr, string cookie)
```

- 功能 发送IM消息时，通知使用者发送结果

- 参数
  - taskID 发送任务id
  - sdkErr 数值参考CRVIDEOSDK_ERR_DEF
  - cookie 用户自定义数据

通知收到IM消息

```
void notifyIMmsg(string fromUserID, string text, int sendTime)
```

- 功能 通知收到IM消息
- 参数
  - fromUserID 消息来源
  - text 消息内容
  - sendTime 消息发送时间戳，从1970开始算起

会话内主功能页切换通知

```
void notifySwitchToPage(int mainPage, string pageID)
```

- 功能 通知功能切换
- 参数
  - mainPage 功能类型，参见MAIN_PAGE_TYPE定义
  - pageID 子页面标识

会话内视频分屏模式通知

```
void notifyVideoWallMode(int model)
```

- 功能 通知视频分屏模式切换
- 参数
  - model 分屏模式，参见VIDEO_LAYOUT_MODE定义

会话内主视频变化通知

```
void notifyMainVideoChanged()
```

- 功能 通知主视频更改
- 参数 无

通知初始化电子白板列表

```
void notifyInitBoards(string BoardObjs)
```

- 功能 SDK入会后通知会议中已经存在的白板列表
- 参数
  - BoardObjs 已经创建好的白板列表, json结构请参见BoardObjs说明

通知初始化白板内图元数据

```
void notifyInitBoardPageDat(string boardID, int boardPageNo, string imgID, string ele
mentDatas,string operatorID)
```

- 功能 初始化白板页数据
- 参数
  - boardID 白板标识
  - boardPageNo 白板页序号
  - imgID 页背景文件ID（空代表无背景）
  - elementDatas 此页的所有图元, 详见json结构之BoardElementObjs
  - operatorID 初始化用户（为空时，代表入会时后台事件）

通知创建/关闭白板

```
void notifyCreateBoard(string jsonBoard, string operatorID)
```

- 功能 通知创建白板
- 参数
  - jsonBoard 白板信息，详见json格式之BoardObj
  - operatorID 创建白板的用户ID

```
void notifyCloseBoard(string boardID, string operatorID)
```

- 功能 通知关闭白板
- 参数
  - boardID 白板标识
  - operatorID 关闭白板的用户ID

通知添加/修改/删除白板图元

```
void notifyAddBoardElement(string boardID, int boardPageNo, string element, string op
eratorID)
```

- 功能 通知添加图元信息
- 参数
  - boardID 白板标识
  - boardPageNo 白板页序号

    - element 图元信息, 详见json结构之BoardElementObj
    - operatorID 添加图元的用户ID

```
void notifyModifyBoardElement(string boardID, int boardPageNo, string element, string
 operatorID)
```

- 功能 通知图元信息被修改
- 参数
    - boardID 白板标识
    - boardPageNo 白板页序号
    - element 图元信息, 详见json结构之BoardElementObj
    - operatorID 添加图元的用户ID

> 应从页内找到旧的图元并替换;

```
void notifyDelBoardElement(string boardID, int boardPageNo, string elementIDs, string
 operatorID)
```

- 功能 通知图元被删除
- 参数
    - boardID 白板标识
    - elementIDs 图元id列表，以 “;”分隔
    - operatorID 删除图元的用户ID

通知白板鼠标热点

```
void notifyMouseHotSpot(string boardID, int boardPageNo, int x, int y, string operato
rID)
```

- 功能 通知鼠标热点消息
- 参数
    - boardID 白板标识
    - x 屏幕横坐标
    - y 屏幕纵坐标
    - operatorID 操作者的用户ID

获取网盘容量信息结果

```
void getNetDiskSummaryRslt(unsigned int diskLimit, unsigned int diskUsed)
```

- 功能 通知获取网盘容量信息结果
- 参数
    - diskLimit 网盘总容量
    - diskUsed 网盘已用容量

获取网盘文件列表结果

```
void getNetDiskFileListRslt(string fileList);
```

- 功能 通知获取网盘文件列表
- 参数
  - fileList 网盘文件列表，json格式，定义见NetFileObjs

删除网盘文件结果

```
void notifyNetDiskFileDeleteRslt(string fileID, int isSucceed)
```

- 功能 通知删除网盘文件结果
- 参数
  - fileID 网盘文件id
  - isSucceed 是否成功，1成功，0失败

通知网盘文件传输进度

```
void notifyNetDiskTransforProgress(string fileID, int percent, int isUpload)
```

- 功能 通知网盘上传或下载进度
- 参数
  - fileID 网盘文件id
  - percent 进度0-100
  - isUpload 是否是上传，1上传，0下载

通知录制文件状态变化

```
void notifyRecordFileStateChanged(string fileName, int state)
```

- 功能 通知录制文件状态更改
- 参数
  - fileName 本地文件路径
  - state 状态，0未上传，1上传中，2已上传，3上传失败

通知录制文件上传进度

```
void notifyRecordFileUploadProgress(string fileName, int percent)
```

- 功能 通知录制文件上传进度
- 参数

  - fileName 本地文件路径
  - percent 进度0-100

通知影音打开/播放/暂停/停止

```
void notifyMediaOpened(int totalTime, int width, int height);
```

- 功能 通知影音文件打开
- 参数
  - totalTime 影音时长(毫秒)
  - width 宽度
  - height 高度

```
void notifyMediaStart(string userid)
```

- 功能 通知影音开始播放
- 参数
  - userid 操作者的用户id

```
void notifyMediaPause(string userid, int bPause)
```

- 功能 通知设置鼠标热点消息
- 参数
  - userid 操作者的用户id
  - bPause 是否暂停，1暂停，0播放

```
void notifyMediaStop(string userid, int reason)
```

- 功能 通知影音播放停止
- 参数
  - userid 操作者的用户id
  - reason 播放停止原因，数值参考MEDIA_STOP_REASON

通知更新影音播放进度

```
void notifyPlayPosSetted(int setPTS)
```

- 功能 通知播放进度已设置完成
- 参数
  - setPTS 播放进度

通知影音播放图像数据

```
void notifyMemberMediaData(string userid, int curPos)
```

- 功能 通知影音帧数据已解码完毕
- 参数
  - userid 操作者的用户id
  - curPos 该影音帧的时间，毫秒为单位

> 收到此通知消息后，可通过getMediaImg获取图像显示；但如果之前显示的帧时戳更大，说明此通知消息已过时，直接忽略即可； 如果使用了影音共享UI显示组件，不再需要自己关注此事件和进行显示处理。

通知语音PCM数据

```
void notifyAudioPCMDat(int aSide, string base64PcmDat)
```

- 功能 通知语音PCM数据
- 参数
  - aSide 声道类型
  - base64PcmDat PCM数据(base64格式)

会议内可视化UI显示组件

成员视频UI显示组件

> CloudroomVideoUI 是视频显示控件，它显示设定的用户的视频，是由视频会议组件内相关接口的封装而成，如果此组件不满足需求，可以使用视频会议组件内相关的接口自行实现。 多实例组件，在会话中可以为每一个视频用户创建一个用来显示其画面

- 设置视频源
- 获取用户/摄像头编号
- 保存成员画面截图
- 获取图像时间戳
- 清空成员画面内容
- 是否保持显示比例
- 是否显示昵称
- 成员是否有图像

设置视频源

```
void setVideo(string userID, int videoID)
```

- 功能 设置显示的目标用户视频
- 返回值 无
- 参数
  - userID 目标用户ID
  - videoID 用户的指定视频设备（-1，代表用户的默认视频设备）

> 从会议内可观看摄像头列表中获取userID和videoID

获取用户/摄像头编号

```
string getUserID()
```

- 功能 获取当前显示的用户
- 返回值 用户ID
- 参数 无

```
string getVideoID()
```

- 功能 获取当前显示的用户的视频设备
- 返回值 用户的摄像头ID

- 参数 无

保存成员画面截图

```
int savePic(string pathFileName)
```

- 功能 截图拍照
- 返回值 0成功，非0为保存遇到的错误码；
- 参数
  - pathFileName 保存为本地路径文件名

```
string savePicToBase64(string format);
```

- 功能 截图拍照
- 返回值 以base64编码的图片数据，不成功返回空
- 参数
  - format 图片格式，支持bmp, png, gif, jpg, jpeg

```
SAFEARRAY(BYTE) savePicToArray(string format)
```

- 功能 截图拍照
- 返回值 图片数据的safe array，不成功返0长度的array
- 参数
  - format 图片格式，支持bmp, png, gif, jpg, jpeg

获取图像时间戳

```
long long getPicFrameTime();
```

- 功能 获取图像时间戳
- 返回值 当前图像数据对应的时间戳
- 参数 无

清空成员画面内容

```
void clear()
```

- 功能 清空缓存的图像
- 返回值 无
- 参数 无

是否保持显示比例

```
bool keepAspectRatio
```

- 功能 设置显示的视频画面是否保持比例

> 属性，为true时保持比例不拉伸，UI组件大小不变，图像数据保持比例居中显示，空余区域黑色填充， false 不保持比例进行拉伸

是否显示昵称

```
int visibleNickName
```

- 功能 检查昵称是否可见

> 属性，0可见，1不可见

成员是否有图像

```
int isPicEmpty
```

- 功能 检查图像是否为空

> 属性，只读，用于检查当前是否有图像，0: 有图像， 1:无图像

屏幕共享UI显示组件

> CloudroomScreenShareUI 是屏幕共享开启后远端显示控件，是由视频会议组件内相关接口的封装而成，如果此组件不满足需求，可以使用视频会议组件内相关的接口自行实现。 单例组件，整个程序的生命过程中只能有有一个实例。

- 保存屏幕截图
- 获取图像时间戳
- 清空共享屏幕内容
- 屏幕共享是否有图像
- 是否允许他人屏幕标注
- 设置画笔风格
- 是否显示保持比例
- 是否允许他人远程控制

保存屏幕截图

```
int savePic(string pathFileName)
```

- 功能 拍照
- 返回值 0成功，非0为保存遇到的错误码；

- 参数
  - pathFileName 保存到本地的路径文件名

```
string savePicToBase64(string format);
```

- 功能 截图拍照
- 返回值 以base64编码的图片数据，不成功返回空
- 参数
  - format 图片格式，支持bmp, png, gif, jpg, jpeg

```
SAFEARRAY(BYTE) savePicToArray(string format)
```

- 功能 截图拍照
- 返回值 图片数据的safe array，不成功返0长度的array
- 参数
  - format 图片格式，支持bmp, png, gif, jpg, jpeg

获取图像时间戳

```
long long getPicFrameTime();
```

- 功能 获取图像时间戳
- 返回值 当前图像数据对应的时间戳
- 参数 无

清空共享屏幕内容

```
void clear()
```

- 功能 清空缓存的图像
- 返回值 无
- 参数 无

共享是否有图像

```
int isPicEmpty
```

- 功能 检查图像是否为空

> 属性，只读，0有图像, 1无图像

是否允许他人屏幕标注

```
bool enableMarked
```

- 功能 是否允许别人进行标注

> 属性

设置画笔风格

```
void setPenStyle(int intRgb, int penWidth)
```

- 功能 设置标注画笔样式
- 参数
  - intRgb 颜色，排列方式bgr(8:8:8)
  - penWidth 画笔宽度

是否显示保持比例

```
bool keepAspectRatio
```

- 功能 绘制模式，是否拉伸绘制

> 属性，为true时UI组件大小不变，图像数据保持比例居中显示，空余区域黑色填充

是否允许他人远程控制

```
int ctrlOpen
```

- 功能 是否允许他人进行远程控制

> 属性

白板显示UI显示组件

> CloudroomBoardUI 白板UI显示组件，是由视频会议组件内相关接口的封装而成，如果此组件不满足需求，可以使用视频会议组件内相关的接口自行实现。 单例组件，整个程序的生命过程中只能有有一个实例。

- 创建电子白板
- 白板工具条是否可见
- 获取白板ID
- 删除白板

创建电子白板

```
string createBoard(int width, int height, string fileName)
```

- 功能 创建组件内的电子白板
- 返回值 返回值白板id字符串，否则是空字符串；
- 参数
  - width 白板宽度
  - height 白板高度
  - fileName 白板背景图片文件，传空值则创建空白板

白板工具条是否可见

```
void setBoardToolBarVisible(bool isVisible)
```

- 功能 设置白板工具条是否可见
- 返回值 无
- 参数
  - isVisible true可见，false不可见

获取白板ID

```
string getBoardID()
```

- 功能 获取当前电子白板的ID
- 返回值 白板的ID
- 参数 无

删除白板

```
void deleteBoard()
```

- 功能 关闭并删除当前白板
- 返回值 无
- 参数 无

影音共享UI显示组件

> CloudroomMediaUI 影音显示控件，是由视频会议组件内相关接口的封装而成，如果此组件不满足需求，可以使用视频会议组件内相关的接口自行实现功能。 单例组件，整个程序的生命过程中只能有有一个实例。

- 保存影音画面截图
- 获取图像时间戳
- 隐藏/显示播放工具条上的UI组件
- 显示比例保持
- 是否禁用工具条

保存影音画面截图

```
int savePicToFile(string pathFileName);
```

- 功能 保存播放影音画面到图片文件
- 返回值 0成功，非0失败
- 参数
  - pathFileName 本地绝对路径文件名

> 支持格式：bmp, png, gif, jpg, jpeg

```
string savePicToBase64(string format);
```

- 功能 保存播放截图为base64数据字符串
- 返回值 以base64编码的图片数据，不成功返回空
- 参数
  - format 支持格式：bmp, png, gif, jpg, jpeg

```
SAFEARRAY(BYTE) savePicToArray(string format)
```

- 功能 截图拍照
- 返回值 图片数据的safe array，不成功返0长度的array
- 参数
  - format 图片格式，支持bmp, png, gif, jpg, jpeg

获取图像时间戳

```
long long getPicFrameTime();
```

- 功能 获取图像时间戳
- 返回值 当前图像数据对应的时间戳
- 参数 无

隐藏/显示播放工具条上的UI组件

```
void setToolBarUIElementVisible(int UIElement, bool isVisible);
```

- 功能 显示隐藏播放工具条上的界面元素
- 返回值 无
- 参数
  - UIElement 界面元素，定义见ToolBarUI
  - isVisible 是否可见

> 影音控件默认工具条可用

显示比例保持

```
bool keepAspectRatio
```

- 功能 绘制模式，是否拉伸绘制

> 属性，为ture时UI组件大小不变，图像数据保持比例居中显示，空余区域黑色填充

是否禁用工具条

```
void disableToolBar(int bDisable)
```

- 功能 工具条是否可用
- 返回值 0可用，非0:不可用
- 参数 无

> 工具条默认不禁用

队列管理组件

> CloudroomQueue 是队列控件，实现队列功能，目的是为了实现用户自动分配。 单例组件，整个程序的生命过程中只能有有一个实例。

接口函数

初始化队列

```
void initQueueDat(string cookie)
```

- 功能 初始化用户队列功能数据
- 返回值 无
- 参数
  - cookie 自定义数据(在响应消息中回传给调用者)
- 说明 在响应回调initQueueDatRslt初始化成功后，才可获取队列队列相关信息；
- 回调函数 initQueueDatRslt

刷新所有队列状态

```
void refreshAllQueueStatus()
```

- 功能 刷新所有队列状态信息
- 返回值 无
- 参数 无
- 说明 当前排队的队列或服务的队列，sdk自动有状态变化回调；其它队列则需要此函数来查询
- 回调函数 触发 queueStatusChanged 通知刷新的结果

查询队列

```
string getAllQueueInfo()
```

- 功能 获取队列信息
- 返回值 返回json对象，结构定义见QueueObjs
- 参数 无

获取队列状态

```
string getQueueStatus(int queID)
```

- 功能 获指定取队列状态
- 返回值 返回json对象，结构定义见QueueStatusObj
- 参数
  - queID 队列id

获取我的排队信息

```
string getQueuingInfo()
```

- 功能获取我的排队信息
- 返回值 返回json对象，结构定义见QueuingObj
- 参数 无

获取我服务的队列信息

```
string getServingQueues()
```

- 功能 获取我服务的所有队列
- 返回值 队列ID列表，以’\n’分割；

获取我的会话信息

```
string getSessionInfo()
```

- 功能 获取我的会话信息
- 返回值 返回json对象，结构定义见SesssionObj

开始 / 停止排队

```
void startQueuing(int queID, string cookie)
```

- 功能 客户开始排队
- 返回值 无
- 参数
  - queID 排队的队列ID
  - cookie 用户自定义数据(在响应消息中回传给调用者)
- 回调函数 startQueuingRslt

```
void startQueuing2(int queID, string usrExtDat, string cookie)
```

- 功能 客户开始排队
- 返回值 无

- 参数
  - queID 排队的队列ID
  - usrExtDat 用户排队扩展数据
  - cookie 用户自定义数据(在响应消息中回传给调用者)
- 回调函数 startQueuingRslt

```
void stopQueuing(string cookie)
```

- 功能 客户停止排队
- 返回值 无
- 参数
  - cookie 用户自定义数据(在响应消息中回传给调用者)
- 回调函数 stopQueuingRslt

开始 / 停止服务队列

```
void startService(int queID, string cookie)
```

- 功能 开始服务某个队列(可以多次调用，开启对多个队列的服务)
- 返回值 无
- 参数
  - queID 队列ID
  - cookie 用户自定义数据(在响应消息中回传给调用者)
- 回调函数 startServiceRslt

> 开启队列服务成功后：
>
> 1. 如果没有开启免打扰，那么系统会自动分配客户：将收到事件autoAssignUser
> 2. 如果开启免打扰，系统就不会分配客户，如需服务客户可调用reqAssignUser

```
void stopService(int queID, string cookie)
```

- 功能 停止服务某个队列
- 返回值 无
- 参数
  - queID 队列ID
  - cookie 用户自定义数据(在响应消息中回传给调用者)
- 回调函数 stopServiceRslt

请求分配用户

```
void reqAssignUser(string cookie)
```

- 功能 请求分配一个客户

- 返回值 无
- 参数
  - cookie 用户自定义数据(在响应消息中回传给调用者)
- 回调函数 reqAssignUserRslt

> 1. 当关闭免打扰时，系统将自动分配客户，无需调用此函数
> 2. 当开启免打扰时，系统不再自动分配客户，座席如需服务客户可使用此函数分配

接受 / 拒绝分配的用户

```
void acceptAssignUser(int queID, string userID, string cookie)
```

- 功能 接受系统安排的客户
- 返回值 无
- 参数
  - queID 排队的队列ID
  - userID 队列中的用户ID
  - cookie 用户自定义数据(在响应消息中回传给调用者)

```
void rejectAssignUser(int queID, string userID, string cookie)
```

- 功能 拒绝系统安排的客户
- 返回值 无
- 参数
  - queID 排队的队列ID
  - userID 队列中的用户ID
  - cookie 用户自定义数据(在响应消息中回传给调用者)

> 被拒绝的客户将重新回到队列的最前端。

通知回调函数

初始化队列结果

```
void initQueueDatRslt(int sdkErr, string cookie)
```

- 功能 队列初始化操作结果
- 参数
  - sdkErr 操作结果代码，数值参考CRVIDEOSDK_ERR_DEF，成功为CRVIDEOSDK_NOERR
  - cookie 自定义用户数据

队列状态变化通知

```
void queueStatusChanged(string jsonQueStatus)
```

- 功能 队列状态变化通知
- 参数
  - jsonQueStatus 新的队列状态，json对象，结构定义见QueueStatusObj

> 1. 在排队的队列、或服务的队列发生变化时，将有队列状态变化通知到来
> 2. 在调用refreshAllQueueStatus时，查询到的队列数据有变化时，会有通知到来

排队信息变化通知

```
void queuingInfoChanged(string jsonQueuingInfo)
```

- 功能 排队信息变化通知
- 参数
  - jsonQueuingInfo json对象，结构定义见QueuingObj

开始 / 停止排队操作结果

```
void startQueuingRslt(int sdkErr, string cookie)
```

- 功能 *startQueuing* 开始排队操作结果
- 参数
  - sdkErr 操作结果代码，数值参考CRVIDEOSDK_ERR_DEF，CRVIDEOSDK_NOERR为成功操作
  - cookie 自定义用户数据

```
void stopQueuingRslt(int sdkErr, string cookie)
```

- 功能 *stopQueuing* 停止排队操作结果
- 参数
  - sdkErr 操作结果代码，数值参CRVIDEOSDK_ERR_DEF，CRVIDEOSDK_NOERR为成功操作
  - cookie 自定义用户数据

开始 / 停止队列服务结果

```
void startServiceRslt(int queID, int sdkErr, string cookie)
```

- 功能 *startService* 开始服务队列操作结果
- 参数
  - queID 服务的队列ID
  - sdkErr 操作结果代码，参考CRVIDEOSDK_ERR_DEF，CRVIDEOSDK_NOERR为成功操作
  - cookie 自定义用户数据

```
void stopServiceRslt(int queID, int sdkErr, string cookie)
```

- 功能 *stopService* 停止服务队列操作结果
- 参数
  - queID 服务的队列ID
  - sdkErr 操作结果代码，参考CRVIDEOSDK_ERR_DEF，CRVIDEOSDK_NOERR为成功操作
  - cookie 自定义用户数据

请求分配用户结果

```
void reqAssignUserRslt(int sdkErr, string jsonUsr, string cookie)
```

- 功能 请求分配客户操作结果
- 参数
  - sdkErr 操作结果代码，参考CRVIDEOSDK_ERR_DEF，CRVIDEOSDK_NOERR为成功操作
  - jsonUser 请求到的队列用户，json对象，结构定义见QueueUser
  - cookie 自定义用户数据

自动分配用户通知

```
void autoAssignUser(string jsonUser)
```

- 功能 队列系统自动分配客户
- 参数
  - jsonUser 安排的用户，json对象，结构定义见QueueUser

> 如果想停止系统的自动分配，请调用CloudroomVideoMgr中的 setDNDStatus 设置免打扰功能。

自动分配用户被取消

```
void cancelAssignUser(int queID, string userID)
```

- 功能 队列系统取消之前自动分配的的客户
- 参数
  - queID 服务的队列
  - userID 用户id

> 队列系统通过接口autoAssignUser给开始队列服务的人自动推送用户，收到系统分配的用户后，如果队列服务者还未决定接受acceptAssignUser还是拒绝rejectAssignUser推送的用户，系统可用取消本次推送，并通过本接口通知队列服务者。

Http文件传输管理组件

> CloudroomHttpFileMgr是http文件上传下载、及文件管理控件

接口函数

启动 / 停止传输管理器

```
void startMgr()
```

- 功能 启动Http文件文件管理组件运行
- 返回值 无
- 参数 无

> 在调用CloudroomVideoSDK的 init 初始化SDK后后方可调用

```
void stopMgr()
```

- 功能 停止Http文件文件管理组件运行
- 返回值 无
- 参数 无

获取所有的Http传输信息

```
string getAllTransferInfos()
```

- 功能 获取本地上传、下载文件信息
- 返回值 json格式的文件信息，详见说明 HttpFileInfoObjs

开始 / 取消传输文件

```
void startTransferFile(string jsonFileInfo)
```

- 功能 开始下载/上传文件
- 返回值 无
- 参数
  - jsonFileInfo json格式的文件信息，详见说明 HttpFileInfoObj

> 说明：上传不支持断点续传； 下载支持断点续传；如果文件传输完成，且fileVersion一致，下次再请求时会立即报告完成。

```
void cancelFileTransfer(string fileName)
```

- 功能 取消传输
- 返回值 无
- 参数
  - fileName 本地路径文件名

> 说明：取消时，只是停止了传输任务，不清理记录及断点文件。

删除传输记录

```
void rmTransferInfo(string fileName, int bRemoveLocFile)
```

- 功能 删除传输记录及相关文件
- 返回值 无
- 参数
  - fileName 文件名
  - bRemoveLocFile 是否移除本地文件

> 说明 此接口将文件从管理器中移除（getAllTransferInfos将不再返回相关信息），如果bRemoveLocFile为1时，那么上传的源始文件、下载的临时文件或结果文件都将被移除。

通知回调函数

文件状态变化

```
void fileStateChanged(string fileName, int state)
```

- 功能 通知用户文件状态更改
- 参数
  - fileName 文件名
  - state 状态，详见 HTTP_TRANSFER_STATE

文件传输(上传/下载)进度

```
void fileProgress(string fileName, int finisedSize, int totalSize)
```

- 功能 通知用户文件的传输进度
- 参数
  - fileName 文件名
  - finisedSize 已传输大小
  - totalSize 文件总大小

传输完成

```
void fileFinished(string fileName, int rslt)
```

- 功能 通知用户文件传输结束
- 参数
  - fileName 文件名
  - rslt 传输结果，详见 HTTP_TRANSFER_RESULT

常量定义

> 相关数值及含义定义

错误码定义

> CRVIDEOSDK_ERR_DEF

代码	数值	含义
CRVIDEOSDK_NOERR	0	没有错误
CRVIDEOSDK_UNKNOWERR	1	未知错误
CRVIDEOSDK_OUTOF_MEM	2	内存不足
CRVIDEOSDK_INNER_ERR	3	sdk内部错误
CRVIDEOSDK_MISMATCHCLIENTVER	4	不支持的sdk版本
CRVIDEOSDK_MEETPARAM_ERR	5	参数错误
CRVIDEOSDK_ERR_DATA	6	无效数据
CRVIDEOSDK_ANCTPSWD_ERR	7	帐号密码不正确
CRVIDEOSDK_SERVER_EXCEPTION	8	服务异常
CRVIDEOSDK_LOGINSTATE_ERROR	9	登录状态错误
CRVIDEOSDK_USER_BEEN_KICKOUT	10	登录用户被踢下线
CRVIDEOSDK_NETWORK_INITFAILED	200	网络初始化失败
CRVIDEOSDK_NO_SERVERINFO	201	没有服务器信息
CRVIDEOSDK_NOSERVER_RSP	202	服务器没有响应
CRVIDEOSDK_CREATE_CONN_FAILED	203	创建连接失败
CRVIDEOSDK_SOCKETEXCEPTION	204	socket异常
CRVIDEOSDK_SOCKETTIMEOUT	205	网络超时
CRVIDEOSDK_FORCEDCLOSECONNECTION	206	连接被关闭
CRVIDEOSDK_CONNECTIONLOST	207	连接丢失
CRVIDEOSDK_QUE_ID_INVALID	400	队列ID错误
CRVIDEOSDK_QUE_NOUSER	401	没有用户在排队
CRVIDEOSDK_QUE_USER_CANCELLED	402	排队用户已取消
CRVIDEOSDK_QUE_SERVICE_NOT_START	403	队列服务还未开启
CRVIDEOSDK_ALREADY_OTHERQUE	404	已在其它队列排队(客户只能在一个队列排队)
CRVIDEOSDK_INVALID_CALLID	600	无效的呼叫ID
CRVIDEOSDK_ERR_CALL_EXIST	601	已在呼叫中
CRVIDEOSDK_ERR_BUSY	602	对方忙
CRVIDEOSDK_ERR_OFFLINE	603	对方不在线
CRVIDEOSDK_ERR_NOANSWER	604	对方无应答
CRVIDEOSDK_ERR_USER_NOT_FOUND	605	用户不存在

CRVIDEOSDK_ERR_REFUSE	606	对方拒接
CRVIDEOSDK_MEETNOTEXIST	800	会议不存在或已结束
CRVIDEOSDK_AUTHERROR	801	会议密码不正确
CRVIDEOSDK_MEMBEROVERFLOWERROR	802	会议终端数量已满（购买的license不够)
CRVIDEOSDK_RESOURCEALLOCATEERROR	803	分配会议资源失败
CRVIDEOSDK_MEETROOMLOCKED	804	会议已加锁
CRVIDEOSDK_CATCH_SCREEN_ERR	900	抓屏失败
CRVIDEOSDK_RECORD_MAX	901	单次录制达到最大时长(8h)
CRVIDEOSDK_RECORD_NO_DISK	902	磁盘空间不够
CRVIDEOSDK_SENDFAIL	1000	发送失败
CRVIDEOSDK_CONTAIN_SENSITIVEWORDS	1001	有敏感词语
CRVIDEOSDK_SENDCMD_LARGE	1100	发送信令数据过大
CRVIDEOSDK_SENDBUFFER_LARGE	1101	发送数据过大
CRVIDEOSDK_SENDDATA_TARGETINVALID	1102	目标用户不存在
CRVIDEOSDK_SENDFILE_FILEINERROR	1103	文件错误
CRVIDEOSDK_TRANSID_INVALID	1104	无效的发送id
CRVIDEOSDK_RECORDFILE_STATE_ERR	1200	状态错误不可上传/取消上传
CRVIDEOSDK_RECORDFILE_NOT_EXIST	1201	录制文件不存在

麦克风（音频）状态

> ASTATUS

代码	数值	含义
AUNKNOWN	0	麦克风状态未知
ANULL	1	没有麦克风设备
ACLOSE	2	麦克风处于关闭状态
AOPEN	3	麦克风处于打开状态
AOPENING	4	向服务器发送打开消息中

视频尺寸定义

> VIDEO_SHOW_SIZE

代码	数值	分辨率及码率
VIDEO_SZ_80	1	144*80, 最大码率：56kbps
VSIZE_SZ_128	2	224*128, 最大码率：72kbps
VSIZE_SZ_160	3	288*160, 最大码率：100kbps
VSIZE_SZ_192	4	336*192, 最大码率：150kbps
VSIZE_SZ_256	5	448*256, 最大码率：200kbps
VSIZE_SZ_288	6	512*288, 最大码率：250kbps
VSIZE_SZ_320	7	576*320, 最大码率：300kbps

VSIZE_SZ_360	8	640*360, 最大码率：350kbps
VSIZE_SZ_400	9	720*400, 最大码率：420kbps
VSIZE_SZ_480	10	848*480, 最大码率：500kbps
VSIZE_SZ_576	11	1024*576, 最大码率：650kbps
VSIZE_SZ_720	12	1280*720, 最大码率：1mbps
VSIZE_SZ_1080	13	1920*1080, 最大码率：2mbps

摄像头（视频）状态定义

> VSTATUS

代码	数值	含义
VUNKNOWN	0	摄像头状态未知
VNULL	1	没有摄像头设备
VCLOSE	2	摄像头处于关闭状态
VOPEN	3	摄像头处于打开状态
VOPENING	4	向服务器发送打开消息中

视频图像格式

> VIDEO_FORMAT

代码	数值	含义
VFMT_YUV420P	0	yuv420p
VFMT_ARGB32	1	32-bit ARGB format (0xAARRGGBB)

录制内容类型

> REC_VCONTENT_TYPE

代码	数值	含义
RECVTP_VIDEO	0	摄像头
RECVTP_PIC	1	图片
RECVTP_SCREEN	2	屏幕
RECVTP_MEDIA	3	影音共享
RECVTP_TIMESTAMP	4	时间戳

录制状态

> RECORD_STATE

代码	数值	含义
NO_RECORD	0	录制未启动

STARTING	1	录制正在开启
RECORDING	2	正在录制
PAUSED	3	录制已暂停
STOPPING	4	录制正在结束

录制文件上传状态

> RECORD_FILE_STATE

代码	数值	含义
RFS_NoUpload	0	未上传
RFS_Uploading	1	上传中
RFS_Uploaded	2	已上传
RFS_UploadFail	3	上传失败

屏幕共享编码类型

> ENCODE_TYPE

代码	数值	含义
ENC_CLOUDROOM	0	云屋科技私有编码格式（清晰度更高、带宽大)
ENC_H264	1	(清晰度差一些，带宽小)

鼠标事件类型

> MOUSE_MSG_TYPE

代码	数值	含义
MOUSE_MOVE	0	鼠标移动
MOUSE_DOWN	1	鼠标键按下
MOUSE_UP	2	鼠标键弹起
MOUSE_DBCLICK	3	鼠标双击

鼠标键类型

> MOUSE_KEY_TYPE

代码	数值	含义
MOUSEKEY_NULL	0	无
MOUSEKEY_L	1	鼠标左键
MOUSEKEY_M	2	鼠标中键
MOUSEKEY_R	3	鼠标右键
MOUSEKEY_WHEEL	4	鼠标滚轮

MOUSEKEY_X	5	鼠标侧键

键盘事件类型

> KEY_MSG_TYPE

代码	数值	含义
KEYT_DWON	0	键值按下
KEYT_UP	1	键值弹起

主功能页类型

> MAIN_PAGE_TYPE

代码	数值	含义
MAINPAGE_VIDEOWALL	0	视频墙
MAINPAGE_SHARE	1	屏幕共享
MAINPAGE_WHITEBOARD	2	电子白板
MAINPAGE_MEDIASHARE	3	影音共享

影音结束原因

> MEDIA_STOP_REASON

代码	数值	含义
MEDIA_CLOSE	0	文件关闭
MEDIA_FINI	1	播放到文件尾部
MEDIA_FILEOPEN_ERR	2	打开文件失败
MEDIA_FORMAT_ERR	3	文件格式错误
MEDIA_UNSUPPORT	4	影音格式不支持
MEDIA_EXCEPTION	5	其他异常

视频墙分屏模式

> VIDEO_LAYOUT_MODE

代码	数值	含义
VLO_1v1_M	0	互看
VLO_WALL1_M	1	1分屏
VLO_WALL2	2	2分屏
VLO_WALL4	3	4分屏
VLO_WALL5_M	4	5分屏
VLO_WALL6_M	5	6分屏

VLO_WALL9	6	9分屏
VLO_WALL13_M	7	13分屏
VLO_WALL16	8	16分屏
VLO_WALL25	9	25分屏

录制的类型

> REC_DATA_TYPE

代码	数值	含义
REC_AV_DEFAULT	0	录制所有语音和视频
REC_AUDIO_LOC	1	录制本地语音
REC_AUDIO_OTHER	2	录制其他人语音
REC_VIDEO	3	录制视频(内容由setRecordVideos设定)

> 用户可以自由组合，如REC_AUDIO_LOC|REC_VIDEO，表示录制本地语音和视频；
> REC_AUDIO_LOC|REC_AUDIO_OTHER，录制双方语音

Http文件传输状态

> HTTP_TRANSFER_STATE

代码	数值	含义
HTTPFS_NULL	0	未开始
HTTPFS_QUEUE	1	排队中
HTTPFS_TRANSFERING	2	传输(上传/下载)中
HTTPFS_FINISHED	3	传输完成

Http文件传输结果

> HTTP_TRANSFER_RESULT

代码	数值	含义
HTTPR_Success	0	成功
HTTPR_InnerErr	1	内部错误
HTTPR_ParamErr	2	参数错误
HTTPR_NetworkFail	3	网络不通/地址不对
HTTPR_NetworkTimeout	4	超时失败
HTTPR_FileOperationFail	5	文件操作失败
HTTPR_PathNotSupprot	6	不支持的路径
HTTPR_FileTransfering	7	文件正在传输
HTTPR_HTTPERR_BEGIN	1000	http错误码启始
HTTPR_HTTPERR_END	1999	http错误码结束

用户状态

> CLIENT_STATUS

代码	数值	含义
OFFLINE	0	离线状态
ONLINE	1	在线，且空闲状态
BUSY	2	在线，忙状态
MEETING	3	在线，会议中状态

影音播放工具条UI组件

> ToolBarUI

代码	数值	含义
BTN_Pause	0	暂停
BTN_Stop	1	停止

对象结构定义

> 相关数据结构定义

会议对象

> MeetInfoObj

```
{"ID":100,"pswd":"","subject":"test","pubMeetUrl":"www.cloudroom.com/auzjie","creator"
:"testuser", "memberCount":4, "startTime": 123455}
```

- **ID:** 会议号，数值0代表会议信息无效
- **pswd:** 会议密码；（空代表会议无密码）
- **subject:** 会议主题
- **pubMeetUrl:** 会议公共链接
- **creator:** 会议创建者
- **memberCount:** 会议内人数
- **startTime:** 会议开始时间(从1970年1月1日00:00:00起)

会议对象列表

> MeetInfoObjs

```
[
{"ID":100,"pswd":"","subject":"test1","pubMeetUrl":"www.cloudroom.com/auzjie", "creat
or":"testuser", "memberCount":4},
{"ID":123,"pswd":"","subject":"test2","pubMeetUrl":"www.cloudroom.com/sdfsdd", "creat
or":"ddd", "memberCount":10},
{"ID":456,"pswd":"","subject":"test3","pubMeetUrl":"www.cloudroom.com/swerwe", "creat
or":"eee", "memberCount":3}
]
```

会议成员

> MemberObj

```
{"userID":"111","nickName":"aaa","audioStatus": 1,"videoStatus": 1}
```

- **userID** 用户ID
- **nickname** 用户昵称
- **audioStatus** 音频状态，数值参考麦克风状态ASTATUS

- **videoStatus** 视频状态，数值参考视频状态定义VSTATUS

会议成员列表

> MemberObjs

```
[
{"userID":"111","nickName":"aaa","audioStatus": 1,"videoStatus": 1},
{"userID":"222","nickName":"bbb","audioStatus": 1,"videoStatus": 1}
]
```

音频配置

> AudioCfgObj

```
{"micName": "aaa", "speakerName":"aaa", "privEC": 0, "privAgc": 0}
```

- **micName** 麦克风设备名称(空代表系统默认设备)
- **speakerName** 扬声器名称(空代表系统默认设备)
- **privEC** 是否开启云屋私有回声抑制
  - **0**： 不开启
  - **1**： 开启(缺省建议不开启)
- **privAgc** 是否开启云屋私有语音自动增益
  - **0**： 不开启
  - **1**： 开启(缺省建议不开启)

视频配置

> VideoCfgObj

```
{"sizeType": 1, "fps": 12}
```

- **sizeType** 视频尺寸，请参见：VIDEO_SHOW_SIZE
- **fps** 视频帧率(5~30)
- **maxbps** 视频码率（1~100*1000*1000）(未配置则使用内部默认值，请参见VIDEO_SHOW_SIZE)
- **qp_min** 最佳质量(18~51，越小质量越好) (未配置则使用内部默认值25)
- **qp_max** 最差质量(18~51， 越大质量越差) (未配置则使用内部默认值36)
- **wh_rate** 视频宽高比,取值如下：
  - **0** 为16:9(未配置时内部默认值)
  - **1** 为4:3
  - **2** 为1:1

> 我们采用的是vbr编码（由质量+码率，双重控制）：
>
> 1. qp范围：质量参数，为的是达到目标质量后，无需花费更大码率提高质量

> 2. maxbps码率控制，是为了确保结果一定不大于“目标码率”（体积受控）
> 3. 当要超出码率控制时，自动降低质量；当质量达到目标时，自动减少码率甚至无码率输出

视频帧图像

> VideoImgObj

```
{ "format":1, "dat":"FKLE34O123F12JX345KFD13…","width":1024, "height":768, "frameTime"
:100}
```

- **format** 图像格式,数值参考视频图像格式VIDEO_FORMAT
- **dat** 图像数据Base64编码
- **width** 图像宽度
- **height** 图像高度
- **frameTime** 图像的时间戳

用户视频信息

> VideoInfoObj

```
{"userID":"111", "videoID":2, "videoName":"camera2"}
```

- **userID** 用户id
- **videoID** 设备id
- **videoName** 设备名称

用户视频信息列表

> VideoInfoObjs

```
[
{"userID":"111","videoID":1,"videoName":"camera1"},
{"userID":"111","videoID":2,"videoName":"camera2"}
]
```

用户视频列表

> VideoIDObjs

```
[
{"userID":"111","videoID":1},
{"userID":"222","videoID":2},
...
]
```

- **userID** 用户id
- **videoID** 设备id

屏幕共享配置

> ScreenShareCfgObj

```
{"encodeType":0, "maxFPS":8, "maxKbps":800, "catchWnd":0, "catchRect":{"left":0, "top"
:0,  "width":1920, "height":1080}}
```

- **encodeType** 编码类型,详见屏幕共享的编码类型ENCODE_TYPE
- **maxFPS** 最大帧率, 缺省为8 (当网络发不动时，帧率会自动下降)
- **maxKbps** 最大码率，缺省800kbps
- **catchRect** 用于实现区域共享，如{"left":10,"top":10,"width":1000,"height":800}
- **catchWnd** 共享窗口的窗口句柄，用于实现窗口共享

录制文件配置

> RecordCfgObj

```
{"filePathName":"D:\\1.mp4","recordWidth":640,"recordHeight":320,"frameRate":8,
"bitRate":500000,"defaultQP":28,"recDataType":1,"isUploadOnRecording":0}
```

- **filePathName** 录像存储的路径文件名,使用完整路径
- **recordWidth** 录制结果中视频文件宽度
- **recordHeight** 录制结果中视频文件高度
- **frameRate** 录制视频文件的帧率，取值范围:1-30(值越大,cpu要求更高，推荐15帧)
- **bitRate** 录制视频文件的最高码率，当图像变化小时，实际码率会低于此值。建议：
  - 640*360: 500000; (500kbps)
  - 1280*720: 1000000; (1mbps)
  - 1920*1080: 2000000; (2mbps)
- **defaultQP** 录制视频文件的缺省质量，缺省值：28

> 取值范围：0~51，0表示完全无损, 51表示质量非常差，推荐高质量取值18，中质量28， 低质量36。

- **recDataType** 内容类型，值参考定义REC_DATATYPE
- **isUploadOnRecording** 上传类型，是否边录边上传，0：手动上传；1：边录制边上传

> 开始录制时 startRecording 传入的参数，目的是为了确定此次录制的文件规格以及此文件的上传方式。

录制内容配置

> RecordContentObj

```
[
{"left":426,"top":124,"width":409,"height":231,"type":0, "keepAspectRatio":1,
"param": {"camid":"testuser.1", "camid":"guestUser.1" }}
]
```

- **left，top，width，height** 在录制画面中的区域（相对于录像尺寸）
- **type** 录制类型REC_VCONTENT_TYPE

  1. 当type=RECVTP_VIDEO时，表示录制的是摄像头区域，param必须包含camid
  2. 当type=RECVTP_PIC时，表示指定的图片，param必须包含resourceid
  3. 当type=RECVTP_SCREEN时，表示录制的是屏幕，可以增加附加参数screenid
  4. 当type= RECVTP_TIMESTAMP时，不用附加任何参数
- **camid** "用户id.摄像头id"
- **resourceid** xxx，调用setPicResource后得到此资源ID
- **screenid** 屏幕序号，-1表示主屏
- **pid** 进程号，0表示未指定进程
- **area** x，y，w，h，抓屏区域，无此参数时，代表抓全屏
- **keepAspectRatio** 内容保持原始比例，0不保持，1保持
- **param** 具体值与type相关

> 录制中调用录制内容配置接口传入的参数，可重复调用以调整录制内容。

录制文件列表

> RecordFileObjs

```
[
{"fileName":"D:\\1.mp4","size":10240,"state":1,"uploadPercent":100},
{"fileName":"D:\\2.mp4","size":13140,"state":2,"uploadPercent":80}
]
```

- **fileName** 文件名，全路径
- **size** 文件大小
- **state**
  - **0** 没有上传
  - **1** 上传中
  - **2** 上传完毕
- **uploadPercent** 录制结果中视频尺寸高度上传进度，*state*为*1*时此字段有效

影音文件

> MediaInfoObj

```
{ "userID": "111", "state":1, mediaName:"D:\1.mp4"}
```

- **userID** 用户id
- **state** 播放状态
  - **0** 播放
  - **1** 暂停
  - **2** 未播放
- **mediaName** 影音文件名

图片资源

> PicResourceObj

```
{"fmt":"picfile","dat":"c:\test.jpg"}
```

- **fmt** 资源格式，可取值："yuv420p"，"rgb32"，"picfile"，"picdat"
- **dat** 资源数据，不同格式时，要提供的数据各不一样：

  > fmt为"yuv420p"时： dat存放的是base64(yuv420p数据)； fmt为"rgb32"时： dat存放的是base64(rgb32数据)； fmt为"picfile"时： dat存放的是“本地文件名”； fmt为"picdat "时：dat存放的是base64(图片文件内容)；

- **width** 图像宽度(像素), 在fmt为"yuv420p"，"rgb32"时，需要此参数
- **height** 图像高度(像素), 在fmt为"yuv420p"，"rgb32"时，需要此参数

白板

> BoardObj

```
{ "boardID": "xx", "title": "board_1", "width": 1024, "height": 768, "pageCount": 1}
```

- **boardID** 白板标识
- **title** 白板的名字
- **width** 宽
- **height** 高
- **pageCount** 页数

白板列表

> BoardObjs

```
[
{ "boardID": "xx","title": "board1","width": 1024,"height": 768,"pageCount": 1},
{ "boardID": "yy","title": "board2","width": 1024,"height": 768,"pageCount": 1}
]
```

白板图元

> BoardElementObj

```
{ "elementID": "xx", "type":100, "left":0, "top":0, …}
```

- **elementID** 图元id
- **type** 图元的类型，值100以下为云屋保留值，100及以上为自定义值
- **left**、**top** 图元在页内的左上角位置
- **…** 可自由扩展

> 说明：
>
> - elementID必须调用createElementID获取, 即使是曾经调此接口创建的然后存入了磁盘文件，再次读入会议时，所有elementID也需要重新生成
> - 如果要和云屋产品互通，那就需要按云屋的定义取值，可以联系云屋获取相关文档

白板图元列表

> BoardElementObjs

```
[
{ "elementID": "xx", "type":100, "orderID":0, "left":0, "top":0, …}
{ "elementID": "yy", "type":100, "orderID":0, "left":100, "top":100, …}
]
```

网盘文件

> NetFileObj

```
{"ownerID":"00112","ownerName":"test","name":"ddd","orgFileName":"dddd.xxx","md5":"sdf234sdf2sfas2","ctime":"2017-12-6 12:13:14",
"size":12345678,"orgSize":123456999,"status":5}
```

- **ownerID** 文件所有者的ID
- **ownerName** 文件所有者的名称
- **name** 在服务器上的文件名
- **orgFileName** 用户的原始文件名
- **md5** 压缩后的md5(如果不压缩，则原文件md5)
- **ctime** 上传时间，字符串，格式"yyyy-MM-dd hh:mm:ss"
- **size** 在服务器上的大小(单位B，类型ulong)
- **orgSize** 原始文件大小(单位B，类型ulong)
- **status** 文件状态

网盘文件列表

> NetFileObjs


```
[{"ownerID":"","ownerName":"","name":"","orgFileName":"","md5":"","ctime":"","size":""
,"orgSize":"","status":""},
{"ownerID":"","ownerName":"","name":"","orgFileName":"","md5":"","ctime":"","size":"",
"orgSize":"","status":""},…]
```


队列信息

> QueueObj


```
{"queID":0,"name":"aaa","desc":"this is desc","prio":1}
```


- **queID** 队列ID
- **name** 队列名称
- **desc** 队列描述
- **prio** 优先级，值越小优先级越高

队列列表

> QueueObjs


```
[
{"queID":0,"name":"aaa","desc":"this is desc","prio":1},
{"queID":0,"name":"bbb","desc":"this is desc","prio":2},
…
]
```


队列状态

> QueueStatusObj


```
{"queID":0,"agent_num":12,"wait_num":3,"srv_num":11}
```


- **queID** 队列id
- **agent_num** 坐席数量
- **wait_num** 排队客户数量
- **srv_num** 正在服务的客户数量

排队信息

> QueuingObj

```
{"queID":0,"position":3,"queuingTime":17}
```

- **queID** 队列ID 我排的队列(-1:代表我没有排队； -2:代表我正在会话中,通过GetSessionInfo可获取相关信息)
- **position** 我的位置
- **queuingTime** 我排队的时长(单位s)

会话信息

> SesssionObj

```
{"callID":"0000123","peerID":"123","peerName":"aaa", "bCallAccepted":1,"meetingID":32
1321,"meetingPswd":"123456","duration":120}
```

- **callID** 会话中的呼叫ID
- **peerID** 会话中的目标用户ID
- **peerName** 会话中的目标用户昵称
- **bCallAccepted** 呼叫是否被对方接受 0:暂未接受，1:已接受
- **meetingID** 会话中分配的会议ID
- **meetingPswd** 会议密码
- **duration** 会话持续的时长(秒)

队列用户

> QueueUser

```
{"queID":1, "usrID":"31231231", "name":"aaa", "queuingTime":10, "usrExtDat":""}
```

- **queID** 队列ID
- **usrID** 用户ID
- **name** 用户昵称
- **queuingTime** 用户排队的时长(单位s)
- **usrExtDat** 用户排队时传入的扩展参数

Http文件传输对象

> HttpFileInfoObj

```
{ "bUploadType" : 1, "filePathName" : "D:/CloudroomVideoSDK_file/test.log",
"fileVersion" : "V1.0", "httpUrl" : "http://10.0.7.130:8080/Upload/", "params" : {},
"fileSize" : 53788511, "finishedSize" : 53788511,  "state" : 3 }
```

- **bUploadType** 传输类型，0:下载类型，1:上传类型
- **filePathName** 本地完整路径文件名(路径中要求有“CloudroomVideoSDK”)

- **fileVersio** 文件版本（可以填版本号，也可以md5，也可以为空）
- **httpUrl** 目标URL
- **params** 特殊参数，字典数据。

  > params详细说明：decodeCREEFile：取值0或1。此参数仅上传有效，0：上传原始文件，1：上传解密的文件

- **fileSize** 文件大小
- **finishedSize** 已传输大小
- **state** 文件传输状态，详见HTTP_TRANSFER_STATE

注意：fileName路径中要求有“CloudroomVideoSDK”，目的是防止利用sdk上传用户隐私文件、或下载恶意文件到系统目录等。

Http文件传输对象列表

> HttpFileInfoObjs

```
[ { "bUploadType" : true, "filePathName" : "D:/CloudroomVideoSDK_file/test.log",
"fileVersion" : "V1.0", "httpUrl" : "http://10.0.7.130:8080/Upload/", "params" : {},
"fileSize" : 53788511, "finishedSize" : 53788511,  "state" : 3 }
]
```

用户状态

> UserStatusObj

```
{"userID":"xxx","userStatus":0,"DNDType":0}
```

- **userID** 用户id
- **userStatus** 用户的在线状态，参见CLIENT_STATUS
- **DNDType** 用户免打扰状态，参见setDNDStatus

用户状态列表

```
[
{"userID":"xxx","userStatus":0,"DNDType":0}
{"userID":"yyy","userStatus":0,"DNDType":0}
]
```