众云视频连接平台

（Real-time Video Services - RVS)

视频采集端 **SDK** 开发指南

Windows

Version 3.0.1

Nov 09, 2015

目录

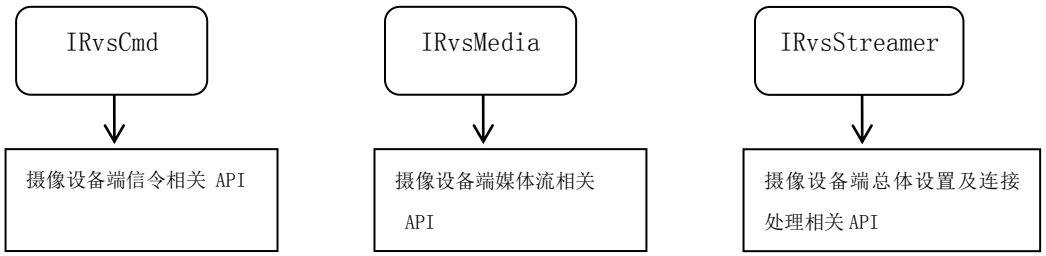
1. 概述

ICHANO SDK 为网络摄像机或智能设备方案商及生产商提供了基于互联网的多媒体连接服务，使设备方案商通过简明的 API 调用，就能将所采集到的音、 视频等多媒体信息通过互联网传输到用户的手机、电脑上，满足用户的监

控、 直播、对讲等相关的各种需求。

ICHANO SDK 分为采集端和观看端两部分，并为不同的操作系统平台提供相应的版本，本文档为 Windows 平台上观看端的 SDK 应用指南。不同操作系统的 ICHANO SDK 的 API 接口的定义相似，工作流程基本相同，可以互为参照。采集端的 SDK 和观看端的 SDK 有较多的关联性，故开发者也可以将观看端文档与采集端文档互为参照。

ICHANO SDK 采集端的头文件主要有 rvs_api.h、rvs_define.h、rvs_cmd_api.h(IRvsCmd)、rvs_media_api.h(IRvsMedia)、rvs_streamer_api.h(IRvsStreamer)。其中 rvs_define.h 头文件为数据类型和数据结构的定义，其余为提供给使用者调用的 API 的接口定义，但是使用者只要包含 rvs_api.h 文件即可。我们根据这些 API 的不同作用， 将这些 API 进行了归类，放到了不同的头文件的文件名中。应 用层也可以理解为 ICHANO SDK 的 3 个模块，每个模块所对应的功能如下图所示：



IRvsStreamer 类是 iChano SDK 的采集端的总体处理类，负责登录鉴权、修改用户名、处理 消息回调通知上层应用。该类是单例，应调用 RvsQueryInterface 来获得该类的实例。 此外，IRvsMedia、IRvsCmd 两个类也是调用 RvsQueryInterface 来获得该类的实例。其中 IRvsMedia 用于对音、视频码流相关的处理进行设置； IRvsCmd 用于采集端与观看端之间的信令交互处理。

2. 名词解释

本文档中所出现的容易引起歧义的名词，解释如下：

应用层	调用本 SDK 接口的代码，在本文档中称之为应用层。本文档第 5 章示例代码，即为应用层代码的示例。
采集端设备	本文档用采集端设备来表示带有视频采集功能和网络传输功能的智能设备，即上文所述“网络摄像机或智能设备”。
用户	本文档中出现的用户，并不表示使用 ICHANO SDK 的客户，而是指购买了采集端设备的使用者个人。
观看端 app	基于 ICHANO SDK 观看端所开发的，安装在用户的手机或电脑上的应用软件，用户可使用该软件来远程连接、设置采集端设备并实现各种监控相关的需求。在 ICHANO SDK 的旧版本文档里也称之为客户端。
观看端设备	也就是安装了上述观看端 app 的手机或电脑。在 ICHANO SDK 的旧版本文档里也称之为客户端设备。

Streamer	ICHANO SDK 采集端的总体处理的类名，与观看端的 Viewer 相对应。在本文档中，若没有特别强调，一个 Streamer 即表示一个采集端。
CID 号	ICHANO SDK 的云端为每个采集端设备分配的一个识别号码，由数字组成。用户第一次使用某个采集端设备时，须在观看端 app 输入该设备的 CID 号码，以区分于其它采集端设备。 当观看端设备与采集端设备处在同一个局域网内时，用户可以在观看端 app 进行“局域网内搜索”操作，来获得局域网内的采集端设备的 CID 号。 应用层若需要及时获取 CID 号，则可在会话状态[见下文]为 E_RVS_SESSION_STATE_CONNECTED 之后调用 CID 号获取接口。 相似地，ICHANO SDK 的云端也为每个观看端设备分配了一个 CID 号，采集端设备中可用此 CID 号来区分前来访问的不同观看端。
设备 ID	生产商为每个采集端设备写入的用于识别的字符串，可由任意数字字母组成。ICHANO SDK 的云端将为每一个设备 ID 产生一个独有的 CID 号供用户使用。 [设备 ID 号需要事先向南京云恩通信有限公司报备以获得授权，否则无法获得有效的 CID 号，也无法正常使用本 SDK 的接口]
镜头数	一般情况下，一个采集端设备只装有一个镜头，镜头数为 1。ICHANO SDK 考虑到支持一些特殊设备可能装有 2 个或以上的镜头，故需要应用层对所支持的镜头数量进行设置。
镜头索引	一般情况下，一个采集端设备内只有一个镜头，镜头索引[Camera Id]取值为 0。当出现多个镜头时，应用层使用该索引来区分不同的镜头。
StreamChannel	一个镜头，常有主、次两路码流[也叫主、辅码流]，两路码流的视频的分辨率、帧率、比特率等属性都不同，有时两路码流对应的音频属性也有区分，ICHANO SDK 以 StreamChannel 的概念来表示同一个镜头的不同码流。并以 StreamCount 和 Stream Id 来表示码流的数量和索引。
会话（SESSION）	ICHANO SDK 允许多个观看端同时连接到同一个采集端进行实时观看和操作。采集端为每一个实时连接、观看和操作的观看端建立一个会话，当观看端退出实时观看页面后，该会话结束。 由于建立会话会增加采集端设备的 CPU 负荷及内存消耗，所以需要应用层根据采集端设备的实际能力来指定最大支持多少会话。（相应地，该设备最多支持多少个观看端同时进行实时连接、观看和操作） 此外，应用层也可以根据会话的状态，来管理该会话所占用的应用层资源。
PTZ	一般指 Pan/Tilt/Zoom，表示云台的全方位移动及镜头控制；。

3. 修订记录

时间	文档版本	修订人	备注
2015-11-09	V3.0.1	印体亮	初稿

4. 接口说明

4.1. IRvsStreamer

4.1.1. 获取 IRvsStreamer 单例

方法： RvsQueryInterface(IRvsStreamer::IID, RVS_VOID** ppClass);
说明： 获取 IRvsStreamer 接口的唯一实例。

4.1.2. IRvsStreamer 的初始化

方法： RVS_INT InitStreamer(const RVS_CHAR* szInitPath);
说明： IRvsStreamer 的初始化就是整个 ICHANO SDK 的初始化工作。
参数：
szInitPath ： 设置配置，日志等工作路径。

4.1.3. 设置开发者密钥

方 法 ： RVS_INT SetLoginInfo(const RVS_UC* pucAppVersion , const RVS_UC* pucCompanyID,RVS_ULL ullCompanyKey,const RVS_UC* pucAppID,const RVS_UC* pucLisence);
说明： 设置 ICHANO SDK 登录服务器所需权限的密钥。
参数： 所需权限的密钥需要联系南京云恩通讯科技有限公司得到并妥善保管。若密钥有误会导致登录服务器失败。

4.1.4. 采集端能力描述

方法： RVS_INT SetCapacity(const ST_RVS_STREAMER_CAPACITY* tCapacity);
说明： 此接口启动了采集端与服务器的连接，连接结果会在回调 StreamerCallback。onLoginResult 接口中反馈给应用层。连接成功后调用 getCID( )可得到一个 CID 号，并具 备了被 iChano 的观看端访问的能力。

4.1.5. 获取 CID 号

方法： RVS_ULL GetStreamerCID();
说明： 获取当前设备的 CID 号，程序第一次 login()成功后，会从 iChano 服务器获得一个唯一的 CID 号。

可在回调接口 onLoginResult 的返回 CONNECTED 之后， 调用此方法获取 CID 号。

4.1.6. 设置用户名和密码

方法：RVS_INT SetUserNameAndPwd(const RVS_UC *pucUsername,const RVS_UC *pucPassword);

说明：设置用户名和密码，用于对观看端进行校验和授权。用户在掌上看家观看端连接当前设备时，所输入的用户名密码必须与这里所设置的用户名密码完全一致，才可以 获得本地设备提供的音视频服务，否则 iChano SDK 会主动断开与观看端的连接。

此外，允许用户在掌上看家观看端对该用户名密码进行设置。观看端设置了此用户名密 码之后，采集端会通过回调接口 StreamerCallback.onUpdateUserNameByRemote 来通知 应用层。 此接口须在回调接口 onLoginResult 的返回值 loginState 为 CONNECTED 之 后调用。

4.1.7. 获取用户名和密码

方法：RVS_INT GetUserNameAndPwd(RVS_UC **ppucUsername, RVS_UC **ppucPassword);

说明：用于获取当前所设置的用户名和密码。

参数：返回值为 String 数组，[0]为用户名，[1]为密码。 此接口须在回调接口 onLoginResult 的返回值 loginState 为 CONNECTED 之 后调用。

4.1.8. 设置采集端设备的名称

方法： RVS_INT SetDeviceName(const RVS_UC *pucDeviceName);

说明：设置设备名称。所设置的设备名称可以被掌上看家观看端获取，从而显示给用户。

此外，允许用户在掌上看家观看端对该名称进行设置。观看端设置了此用户名密码之后， 采集端会通过回调接口 StreamerCallback.onDeviceNameChange 来通知应用层。 此接口须在回调接口 onLoginResult 的返回值 loginState 为 CONNECTED 之 后调用。

4.1.9. 获取采集端设备的名称

方法：RVS_INT GetDeviceName(RVS_UC **ppucDeviceName);

说明：本地[指同一个局域网内]搜索采集端，搜索结果将通过 IRvsStreamerEvent 回调方法返回给应用层。具体回调方法的定义见 OnLanSearchStreamerResult 的定义。

4.1.10. 设置 Streamer 的回调接口

方法： RVS_INT SetStreamerEvent(IRvsStreamerEvent* pStreamerEvent);

说明：设置回调方法，具体回调方法的定义见 IRvsStreamerEvent。

4.1.11. 回调接口 IRvsStreamerEvent

定义：

```
class IRvsStreamerEvent
{
public:
    //login status notification.
    virtual RVS_VOID OnLoginResult(EN_RVS_LOGIN_STATE enLoginState, RVS_UI
uiProgressRate, EN_RVS_LOGIN_ERR enErrCode) = 0;

    //remote client connect status notification.
    virtual    RVS_VOID    OnSessionStatusChanged(RVS_ULL    ullRemoteCID,
EN_RVS_STREAMER_SESSION_STATE enState)= 0;

    //streamer cid  update message notification.
    virtual RVS_INT OnUpdateStreamerCID( const RVS_UC *pucStreamerCID) = 0;

    //streamer name changed message notification.
    virtual RVS_INT OnStreamerNameChanged(const RVS_UC *pucDeviceName) = 0;

    //username or password changed message notification.
    virtual  RVS_INT  OnUserNamePwdChanged(const  RVS_UC  *pucUserName,const
RVS_UC *pucPwd) = 0;

    //streamer notify message.
    virtual        RVS_INT        OnStreamerInfoNotify(EN_RVS_STRAMER_INFO
enStreamerInfoType , RVS_INT iStatus ) = 0;
};
```

4.2. IRvsMedia 接口

Media 是 iChano SDK 中对音、视频码流进行处理的类，该接口实现类是单例，应调用 RvsQueryInterface()来获得该类的实例。

4.2.1. 获取 IRvsMedia 单例

方法： RvsQueryInterface(IRvsMedia::IID, RVS_VOID** ppClass);
说明： 获取 IRvsMedia 接口的唯一实例。

4.2.2. 设置采集端的镜头相关的能力

方 法 ： RVS_INT SetCameraCapacity(RVS_UI uiCameraId, ST_RVS_CAMERA_CAPACITY* pstCameraCapacity);

说明：此接口设置的能力包括：视频流是一路还是多路，镜头是否有闪光灯或红外灯，镜头的转动或移动能力。

4.2.3. 设置视频码流的格式

方法：RVS_INT SetCameraStreamProperty(RVS_UI uiCammeraId,
RVS_UI uiStreamId,
ST_RVS_VIDEO_FORMAT stVideoFormat);

说明：应用层调用此方法告知 iChano SDK 当前所采集的视频码流的格式参数，包括分辨率、编码格式等。 当采集端设备的压缩方式采用定码率的方式时，还需要设置格式参数中的帧率、比特率、 I 帧间隔。当采集端设备的压缩方式采用变码率的方式时，则还需要设置格式参数中的 质量等级，SDK 在需要调用 4.2 节中的回调方法来调节该码流采集的质量等级时，以此 作为初始参照值进行调节。

参数：ST_RVS_VIDEO_FORMAT 视频码流属性，具体见此类的定义。。

4.2.4. 设置音频码流的格式

方法：RVS_INT SetAudioProperty(RVS_UI iMicId, ST_RVS_AUDIO_FORMAT stAudioFormat);

说明：应用层调用此接口告知 iChano SDK 当前所采集的音频码流的格式参数，包括音频采样率、位深、声道数、编码格式等。 当编码格式为 E_RVS_AUDIO_TYPE_PCM16 时，音频将由 SDK 压缩成 AAC 格式传递给 Client。

参数：ST_RVS_AUDIO_FORMAT 音频码流属性，具体见此类的定义。

4.2.5. 视频码流传输接口

方法：RVS_INT WriteVideoStream(RVS_UI uiCammeraId,
RVS_UI uiStreamId,
RVS_UC* pucData,
RVS_UI uiLen,
RVS_UI uiTimestamp,
RVS_BOOL bIFrame);

说明：设置采集端定时录制的时间计划。

4.2.6. 音频码流传输接口

方 法 ： RVS_INT WriteAudioData(RVS_UI iMicId, RVS_UC* pucData, RVS_UI uiLen, RVS_UI uiTimestamp);

说明：采集到的音频码流数据，通过调用此方法传输给 iChano SDK 以发往观看端。

4.2.7. 获取逆向码流的描述信息

方法：RVS_INT GetReverseMediaDesc(RVS_ULL uiHandle, RVS_REV_MEDIA_DESC* pDesc);

说明：此方法用于获取来自观看端的媒体码流的数据格式的描述信息。 具体见 RVS_REV_MEDIA_DESC 的定义。

4.2.8. 回调接口 IRvsMediaEvent

定义：

```
class IRvsMediaEvent
{
public:
    virtual ~IRvsMediaEvent(){}
    ////初始化编码库;
    //virtual    RVS_VOID*    InitEncoder(RVS_UI    uiVideoTypeIn,    RVS_UI
uiVideoWidth, RVS_UI uiVideoHeight) = 0;

    ////开始编码;
    //virtual  RVS_INT     DoEncode(RVS_VOID*  hHandle,RVS_UC  ucNeedIFrame,
RVS_UC *pucInFrame,  RVS_UC **ppucOutFrame,RVS_UI   *puiOutLen) = 0;

    ////释放编码库资源;
    //virtual RVS_INT   FreeEncoder(RVS_VOID* hHandle ) = 0;

    ////调节编码库参数;
    //virtual RVS_INT     AjustEncoder(RVS_VOID*              hHandle,RVS_UC
ucAdjustType) = 0;

    ////开始压缩成 jpeg;
    //virtual  RVS_INT     EncodeJpg(RVS_VOID*  hHandle,   RVS_UC  *pucInYUV,
RVS_UI uiInLenYUV, RVS_UC **ppucOutJpg,RVS_UI   *puiOutLenJpg) = 0;

    //其他事件消息通知;
    virtual RVS_INT  OnEventNotify(RVS_UI  uiCammeraId,  RVS_UI  uiStreamId,
RVS_UI uiMsg) = 0;

    //事件通知，具体消息见 T_EVENT_FLAG 定义;
    virtual  RVS_INT  OnMediaEncodeNotify(T_ENCODE_EVENT  eventFlag,RVS_UI
uiCamID,RVS_UI uiValue ) = 0;
```

```
        //事件通知，具体消息见 T_EVENT_FLAG 定义；
        virtual   RVS_INT   OnMediaStatusNotify(T_EVENT_FLAG   eventFlag,RVS_UI
uiCamID,RVS_UI uiValue,RVS_CHAR* strvalue) = 0;

        //报警设置时间段变化通知；
        virtual     RVS_INT     OnMotionSettingChanged(RVS_UI     uiCamID,RVS_UI
uiSchduledCount,const T_MOTION_SETTING* tValue) = 0;

        //定时录制时间段变化通知；
        virtual    RVS_INT    OnScheduledSettingChanged(RVS_UI    uiCamID,RVS_UI
uiSchduledCount,const T_SCHEDULED_SETTING* tValue) = 0;


        virtual       RVS_INT       OnVideoYuvOutput(RVS_UI       uiCamID,RVS_UI
uiStreamID,RVS_UC** ppucYuvData,RVS_UI *puiYuvLen) = 0;


        virtual  RVS_INT  OnVideoImageOutput(RVS_UI  uiCamID,RVS_UI  uiStreamID,
RVS_UC **ppucJpgData, RVS_UI* puiJpgLen,T_IMAGE_SIZE uiImageFlag) = 0;


        virtual    RVS_VOID    OnMediaChannelStateChanged(RVS_UI    uiChannelID,
EN_RVS_STREAMER_MEDIASTREAM_STATE enState) = 0;


        virtual RVS_INT  OnReverseMediaStatus(RVS_ULL ullHadnle,RVS_UI uiStatus,
RVS_ULL ullPeerCid, RVS_CHAR*  pucUrl, EN_RVS_MEDIA_TYPE enType) = 0;


        virtual   RVS_INT     OnReverseMediaDataCB(RVS_ULL   ullHadnle,   RVS_UC
*pucOutStream, RVS_UI uiDatalen, RVS_UI uiTimeStamp, EN_RVS_MEDIA_TYPE enType)
= 0;


    };
```

4.3. IRvsCmd 类

IRvsCmd 类是在采集端与观看端之间进行信令交互处理的类，该类是单例，调用 RvsQueryInterface 来获得该类的实例。用于发送自定义数据到观看端，响应观看端发送来的控制采集端设备的 PTZ 动作等命令，响应观看端发送来的查询采集端设备信息的命令。

4.3.1. 设置 IRvsCmd 类的回调处理接口

方法： RVS_INT SetStreamCmdEvent(IRvsCmdEvent* streamEvent);

说明： 采集端接收到观看端发送来的操作信令后，调用 IRvsCmdEvent 中的回调方法来处理。具体见 IRvsCmdEvent 的定义。

4.3.2. 发送自定义信令

方法： RVS_INT SendCustomData(RVS_ULL ullStreamerCID,
RVS_UC* pucDataBuffer,
RVS_UC* pucBufLen)

说明： 发送自定义的信令给采集端。用户可以通过 IRvsCmdEvent 接口中的回调函数 OnRecvCustomData 来收取来自观看端的自定义信令数据。

4.3.3. 回调接口 IRvsCmdEvent

定义：

```
class IRvsCmdEvent
{
public:
  virtual ~IRvsCmdEvent(){}
    //云台控制通知；tPtzType：具体类型见定义；point：表示使用相对控制，表示绝对数值控制，一般选，value：表示控制数值大小；
    virtual RVS_INT OnPtzControl(RVS_UI uiCamID,T_PTZ_CTRL tPtzType,RVS_UI point, RVS_UI value  ) = 0;
    //收到其他控制数据，可自定义控制消息；
    virtual RVS_INT OnRecvCustomData(RVS_ULL ullCID ,RVS_CHAR* data,RVS_UI uiLen) = 0;
    virtual RVS_INT OnChangeStreamQuality(RVS_UI uiCamID,RVS_UI uiStreamId,RVS_UI uiFramerate,RVS_UI uiBitrate,RVS_UI uiIFrameInterval,RVS_UI uiQuality) = 0;
};
```

5. 调用流程

1、对 IRvsStreamer、IRvsMedia、IRvsCmd 三个接口调用 RvsQueryInterface 获取相应实例，并进行初始化。

2、实现 IRvsStreamer、IRvsMedia、IRvsCmd 三个接口对应的回调通知基类的方法，创建实现类的实例，并通过类的设置事件通知类方法设置事件通知类。

3、IRvsStreamer 登录 StartLogin() 。

4、登录成功以后调用等待观看端的请求；

5、采集端收到观看端的播放请求后，通过 IRvsMedia 接口 WriteVideoStream 和 WriteAudioData，循环写入 H264 和 AAC 数据，发送到观看端。

6、退出并销毁

依次调用 IRvsStreamer 接口的 StopLogin、DestroyStreamer 和 Release 函数，关闭采集端 sdk 资源，并释放申请的资源。

6. 备注

略。