众云视频连接平台

（Real-time Video Services - RVS)

视频采集端 SDK 开发指南

Linux

Version 3.2.1 Beta 1

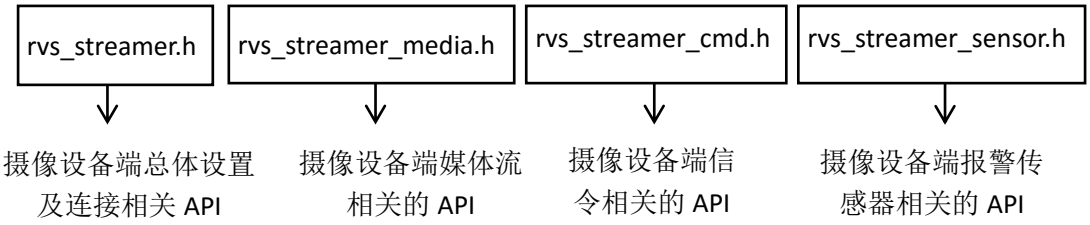
June 30, 2016

目录

1 概述

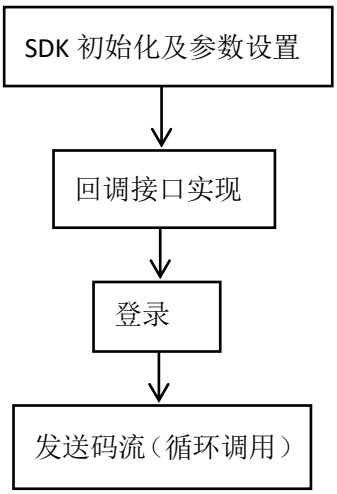
ICHANO SDK 为网络摄像机或智能设备方案商及生产商提供了基于互联网的多媒体连接服务，使设备方案商通过简明的 API 调用，就能将所采集到的音、视频等多媒体信息通过互联网传输到用户的手机、电脑上，满足用户的监控、直播、对讲等相关的各种需求。

ICHANO SDK 的头文件有 rvs_streamer_type.h 、 rvs_streamer_media_type.h 、rvs_streamer_cmd_type.h、rvs_streamer_sensor_type.h、rvs_streamer.h、rvs_streamer_media.h、rvs_streamer_sensor.h、rvs_streamer_cmd.h。其中文件名带 type 的头文件中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。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。[这四个头文件的定义，与 android 等其它平台的 ICHANO SDK 中的 streamer、media、cmd、sensor 这四个类的定义相对应]，应用层也可以理解为 ICHANO SDK 的 4 个模块，每个模块所对应的功能如下图所示：



使用 ICHANO SDK 编程，make 时需要链接这 5 个 ICHANO SDK 的库：libcos.a、libengine.a、libthirdparty.a、libbusiness.a、libsa.a。

下图中为本 SDK 的调用流程的示意图。较详细的 API 调用流程见本文档第 6 章。



下文中对 ICHANO SDK 的 API 进行详细说明时，每一章对应上图中的一个步骤。

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 6 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”或者“摄像设备端”。有时也简称为“采集端”。
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	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、设置采集端并实现各种监控相关的需求，也有时称为“客户端”。
观看端设备	安装了上述观看端 app 的手机或电脑。有时也直接简称为“观看端”。
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，须在观看端输入该采集端设备的 CID 号码，以区分于其它设备。 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	采集端设备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，通常会有主、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）	采集端允许多个观看端同时连接到同一个采集端设备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，该采集端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；ICHANO SDK 将屏幕翻转和镜像功能也一并定义到了 PTZ 控制相关的接口中。

3 修订记录

时间	文档版本	修订人	备注
2014-09-01	1.0.0	徐友仁	初稿
2015-01-22	1.0.1	张晓波	SDK 版本更新到 2.0.9
2015-05-25	2.1.0	徐友仁	新版本的 SDK
2016-06-30	3.2.1	张敏	SDK 版本更新到 3.2.1

4 接口说明

4.1 SDK 初始化及参数设置接口

4.1.1 SDK 初始化

此接口为 ICHANO SDK（采集端）的初始化接口。此接口须作为 ICHANO SDK 的第一个 API 进行调用。

接口定义	
RVS_INT Rvs_Streamer_Init(const RVS_UC* pucWorkPath, const RVS_UC* pucCachePath, const RVS_UC* pucAppVersion);	
参数名称	参数说明
pucWorkPath	工作目录，用于存放 ICHANO SDK 运行所需的配置文件等重要文件。
pucCachePath	缓存目录，用于存放录制文件、LOG 文件等允许应用层及用户删除的文件。
pucAppVersion	此版本号由应用层填写，当应用的版本有变化时，ICHANO SDK 可以搜集此信息并与应用的最新版本号进行比对，提醒用户进行版本升级。
返回结果	
成功返回 RVS_OK。RVS_OK 的值为 0，下同。	

4.1.2 设置本设备的能力属性

此接口设置本采集端设备所支持的能力属性，设置该能力集可以方便本地资源管理，观看端设备也可以获取到该设备的能力属性从而设计出更适合该设备的 UI 页面。

接口定义	
RVS_INT Rvs_Streamer_SetCapacity(const ST_RVS_STREAMER_CAPACITY* pstCapacity);	
参数名称	参数说明
pstCapacity	能力参数 typedef struct ST_RVS_STREAMER_CAPACITY { RVS_UI uiRunMode; /** 0x00 Manual Run; 0x01 Auto Run; 0x02 Background Run; 0x04 Suspend Run; */ RVS_UI uiRecordMode; /** 0x00 No Record; 0x01 Record by the third party; 0x02 Record by SDK; */ RVS_UI uiTimeZoneMode; /** 0x00 Not Support set timezone; 0x01 Support. */ RVS_UI uiEchoCancelMode; /** 0x00 Not support echo cancel, 0x01 Support. */ RVS_UI uiWlanMode; /** 0x00 NOT support; 0x01 Support Command setting; 0x02 Support Sound setting; 0x04 Support Broadcast setting; 0x08 Support AP setting; 0x10 Support QRCode setting; */ }ST_RVS_STREAMER_CAPACITY; 此结构体的成员变量的取值是按位定义的。比如 uiRunMode=3，则说明支持 0x01AutoRun 和 0x02Background Run 两个 runMode，即支持开机自动运行和后台运行这两个运行模式，不支持 SuspendRun，即不支持休眠唤醒运行模式。 录制模式参数 uiRecordMode 的取值，若取 0x01，则表示录制功能由第三方实现，若取 0x02，则表示录制功能由 sdk 内部实现。 uiEchoCancelMode 表示是否支持回音消除，若应用层支持回音消除，则对应的观看端设备可以做相应支持。 uiWlanMode 表示 wifi 设置方式，取值如上所示；
返回结果	
成功返回 RVS_OK。	

4.1.3 设置用户信息[登录名和密码]

此接口，将应用层所获取到的当前的最新的登录名和密码设置给 ICHANO SDK，供 ICHANO SDK 对观看端用户的权限进行验证。详见附录 7.1.1 的描述。

接口定义	
RVS_INT Rvs_Streamer_SetUserNameAndPwd(RVS_UC* pucUserName, RVS_UC* pucPwd);	
参数名称	参数说明
pucUserName	用户名
pucPwd	密码
返回结果	
成功返回 RVS_OK。	

4.1.4 设置采集端设备的名称

此接口允许为采集端设备设置较为个性化的名称，此名称将会显示在观看端的用户操作界面。注意，此接口需在 4.2.1 中的 pfunLoginResult 回调函数中的 enLoginState 返回成功 E_RVS_LOGIN_STATE_CONNED 后调用才有效。

接口定义	
RVS_INT Rvs_Streamer_SetDeviceName(const RVS_UC* pucDeviceName);	
参数名称	参数说明
pucDeviceName	采集端设备名称
返回结果	
成功返回 RVS_OK。	

4.1.5 设置会话数

本接口所设置的会话数，为同一时间该采集端设备所能支持的观看端连接数量的最大值。本 SDK 的使用者应根据采集端产品的软硬件能力自行确定。

接口定义	
RVS_INT Rvs_Streamer_SetSessionMaxNum(RVS_INT iMaxSessionNum);	
参数名称	参数说明
iMaxSessionNum	同一时间该采集端设备支持的观看端连接数量的最大值
返回结果	
成功返回 RVS_OK。	

4.1.6 设置镜头数量

一般情况下一个采集端设备只有 1 个镜头，此值设置为 1；少数设备可能会存在多个镜头的情况。

接口定义	
RVS_INT Rvs_Streamer_Media_SetCameraCount(RVS_UI uiCount);	
参数名称	参数说明
uiCount	采集端设备的镜头数量
返回结果	
成功返回 RVS_OK。	

4.1.7 设置码流数量

同一个镜头，常常会录制为主、次两路码流。

接口定义	
RVS_INT Rvs_Streamer_Media_SetCameraStreamCount(RVS_INT iCamId, RVS_UI uiStreamCount);	
参数名称	参数说明
iCamId	镜头索引，取值从 0 计数。
uiStreamCount	当前的镜头支持几路码流
返回结果	
成功返回 RVS_OK。	

4.1.8 设置镜头相关的能力属性

此接口设置具体某一个镜头相关的能力属性。

接口定义	
RVS_INT Rvs_Streamer_Media_SetCameraCapacity(RVS_INT iCamId， ST_RVS_CAMERA_CAPACITY* pstCameraCapacity);	
参数名称	参数说明

pstCamdraCapacity	能力参数
	typedefstructstRVS_CAMERA_CAPACITY{
	RVS_UI uiPTZMoveMode; // 0x00 not support; 0x01 P control; 0x02 T control; 0x04 Z control; 0x08 Move X control; 0x10 Move Y control; 0x20 Move Z control
	RVS_UI uiTorchMode; // 0x00 Not support Torch; 0x01 Support TorchMode
	RVS_UI uiRotateMode; // 0x00 Not support Rotate; 0x01 Support Rotate
	RVS_UI uiStreamMode; // 0x00 One stream can used at one time; 0x01 Multi stream can used at the same time.
	}ST_RVS_CAMERA_CAPACITY;

返回结果
成功返回 RVS_OK。

4.1.9 设置一路视频码流的格式

应用层调用此接口告知 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 节中的回调接口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。

接口定义	
RVS_INT Rvs_Streamer_Media_SetCameraStreamProperty (RVS_UI uiCammeraId,RVS_UI uiStreamId, ST_RVS_STREAMER_CAMERASTREAM_INFO* pstVideoFormat);	
参数名称	参数说明
iCamid	镜头索引，取值从 0 计数。
iStreamid	码流索引，取值从 0 计数，小于 SetCameraStreamChannelCount 中所设置的码流数量。

stVideoFormat	typedefenumenum_RVS_VIDEO_TYPE
	{
	E_RVS_VIDEO_TYPE_NOVIDEO = 0,
	E_RVS_VIDEO_TYPE_JPEG = 1,
	E_RVS_VIDEO_TYPE_H264 = 2,
	E_RVS_VIDEO_TYPE_H265 = 3,
	}EN_RVS_VIDEO_TYPE;
	
	Typedef struct stRvsCameraStreamInfo
	{
	RVS_UI enVideoType; // 支持的编码类型 EN_RVS_VIDEO_TYPE
	RVS_UI uiWidth; //宽，以像素为单位
	RVS_UI uiHeight; // 高，以像素为单位。
	RVS_UI uiBitrate;
	//编码在外部完成的话,这几个值 SDK 仅仅是记录一下
	RVS_UI uiFramerate; //帧率。
	RVS_UI uiIFrameInterval; //I 帧间隔。
	RVS_UI uiImageQuality; //质量等级。
	}ST_RVS_STREAMER_CAMERASTREAM_INFO;

返回结果
成功返回 RVS_OK。

4.1.10设置麦克风数量

此接口用于决定采集几路音频码流。

接口定义	
RVS_INT Rvs_Streamer_Media_SetMicCount(RVS_UI uiCount);	
参数名称	参数说明
uiCount	用于采集音频的麦克风的数量。
返回结果	
成功返回 RVS_OK。	

4.1.11设置一路麦克风所采集码流的格式

应用层调用此接口告知 ICHANO SDK 当前麦克风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。

接口定义	
RVS_INT Rvs_Streamer_Media_SetMicProperty(RVS_UI iMicId, ST_RVS_STREAMER_MIC_INFOstAudioFormat);	
参数名称	参数说明
iMicId	麦克风索引，取值从 0 计数。

stVideoFormat	typedefenumenum_RVS_AUDIO_TYPE
	{
	E_RVS_AUDIO_TYPE_NOAUDIO = 0,
	E_RVS_AUDIO_TYPE_AAC = 1,
	E_RVS_AUDIO_TYPE_G711A = 2,
	E_RVS_AUDIO_TYPE_G711U = 3,
	E_RVS_AUDIO_TYPE_PCM16 = 4,
	}EN_RVS_AUDIO_TYPE;
	
	typedefstructstRvsMicInfo{
	RVS_UI enAudioType; // 支持的编码类型 EN_RVS_AUDIO_TYPE
	RVS_UI uiSampleRate;
	RVS_UI uiChannel;
	RVS_UI uiDepth;
	}ST_RVS_MIC_INFO;

返回结果	
成功返回 RVS_OK。	

4.1.12设置采集端的语言

此接口用于设置采集端的语言属性。

接口定义	
RVS_INT Rvs_Streamer_SetLanguage(RVS_UI uiLanguage);	
参数名称	参数说明
uiLanguage	/*0. 未知（en）; 1.CN; 2.EN; 3. 繁体中文; 4. 法语；5. 日语；6.西班牙语；7.韩语； */
返回结果	
成功返回 RVS_OK。	

4.1.13设置 sdk 打印信息开关

此接口用于设置 sdk 打印信息开关。

接口定义	
RVS_INT Rvs_SysSetDebugMode(RVS_BOOL bOpen);	
参数名称	参数说明
bOpen	bOpen 取值为 RVS_TRUE，表示开启 sdk 打印信息开关； bOpen 取值为 RVS_FALSE，表示关闭 sdk 打印信息开关；
返回结果	
成功返回 RVS_OK。	

4.1.14设置采集端设备能力

此接口用于设置采集端设备的能力。

接口定义	
RVS_INT Rvs_Streamer_SetDeviceAbility(EN_RVS_DEVICE_ABILITY enAbility);	
参数名称	参数说明
enAbility	typedef enum enum_RVS_DEVICE_ABILITY { EN_RVS_DEVICE_ABILITY_RICH =0, EN_RVS_DEVICE_ABILITY_MID =1, EN_RVS_DEVICE_ABILITY_POOR =2 }EN_RVS_DEVICE_ABILITY; enAbility 取值由设备可用内存情况而定，当采集端设备可用内存较小时，

	建议设置为 EN_RVS_DEVICE_ABILITY_POOR；
返回结果	
成功返回 RVS_OK。	

4.2 回调函数及其设置接口

本节所定义的回调函数接口由应用层实现，ICHANO SDK 将在执行过程中调用这些回调函数。

为了简化回调函数的设置接口，我们将一些具有较强的关联性或相似性的回调函数集中定义到了结构体中。[结构体的成员为回调函数指针]

4.2.1 设备端连接相关回调函数

本节所设置的为设备端与连接相关，或设备端名称相关的回调函数。

回调函数定义如下：

登录结果反馈接口

回调函数定义	
typedef RVS_VOID (*PFUN_ONLOGINRESULT)(EN_RVS_LOGIN_STATE enLoginState, RVS_UI uiProgressRate, EN_RVS_LOGIN_ERR enErrCode);	
参数名称	参数说明
enLoginState	当登陆状态为 E_RVS_LOGIN_STATE_CONNING 时，用 uiProgressRate 来查看登陆的进展。 当登陆状态为 E_RVS_LOGIN_STATE_DISCONNED 时，通过 EN_RVS_LOGIN_ERR 来查看错误码。 当登陆状态为 E_RVS_LOGIN_STATE_CONNED 时表示登陆成功。
uiProgressRate	登陆过程中的每个登陆步骤的完成情况。
enErrCode	登陆失败的错误码。
返回结果	
无	

CID 号更新通知接口

回调函数定义	
typedef RVS_VOID (*PFUN_ONUPDATECID)(RVS_UC* pucLocalCID);	
参数名称	参数说明
pucLocalCID	CID 号此处是以字符串的形式通知给应用层的。
返回结果	
无	

设备名称变化通知接口

回调函数定义	
typedef RVS_VOID (*PFUN_ONDEVICENAMECHANGE)(RVS_UC* pucDeviceName);	
参数名称	参数说明
pucDeviceName	观看端所设置的该设备的设备名称
返回结果	
无	

用户名密码更新回调接口

回调函数定义	
typedef RVS_INT (*PFUN_ONUPDATEUSERNAME)(RVS_UC* pucUserName, RVS_UC* pucPwd);	
参数名称	参数说明
pucUserName	观看端所设置的该设备的登陆用户名
pucPwd	观看端所设置的该设备的登陆密码
返回结果	
无	

会话状态变化通知

回调函数定义	
typedef RVS_VOID (*PFUN_ONSESSIONSTATECHANGE)(RVS_ULL ullRemoteCID, EN_RVS_SESSION_STATE enState);	
参数名称	参数说明
ullRemoteCID	该会话所对应的观看端的 CID 号
enState	该会话的状态
返回结果	
无	

会话连接数统计

回调函数定义	
typedef RVS_VOID (*PFUN_ONWATCHSTATECHANGE)(RVS_ULL ullRemoteCID, RVS_UI uiConnectCount);	
参数名称	参数说明
ullRemoteCID	该会话所对应的观看端的 CID 号
uiConnectCount	会话连接个数
返回结果	
无	

蜂蜜器状态设置

回调函数定义	
typedef RVS_VOID (*PFUN_ONBEEPSTATECHANGE)(RVS_UI uiBeepState);	
参数名称	参数说明
uiBeepState	uiBeepState 为 1 表示蜂鸣器开启，为 0 表示关闭；
返回结果	

无

报警录像设置

回调函数定义	
typedef RVS_VOID (*PFUN_ONALARMRECORDSTATECHANGE)(RVS_UI uiRecState);	
参数名称	参数说明
uiRecState	uiRecState 为 1 表示报警录像开启，为 0 表示关闭；（注意：此回调函数在报警录像由第三方实现时使用，如果报警录像由 sdk 实现，无需调用）
返回结果	
无	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_SetCallback(ST_RVS_STREAMER_CALLBACK stRvsStreamerCallback);	
参数名称	参数说明

stVideoFormat	typedefstructstRVS_STREAMER_CALLBACK
	{
	PFUN_ONLOGINRESULT pfunLoginResult; //登录结果
	PFUN_ONUPDATECID pfunOnUpdateCID; //CID 信息更新
	PFUN_ONDEVICENAMECHANGE pfunOnDeviceNameChangeByRemote;
	//设备名称有变化时通知
	PFUN_ONUPDATEUSERNAME pfunOnUpdateUserNameByRemote;
	//用户名与密码更新
	PFUN_ONSESSIONSTATECHANGE pfunOnSessionStateChange;
	//会话状态更新
	PFUN_ONWATCHSTATECHANGE pfunOnWatchStateChange;
	//会话连接数统计;
	PFUN_ONBEEPSTATECHANGE pfunOnBeepStateChange;
	//蜂鸣器设置;
	PFUN_ONALARMRECORDSTATECHANGE pfunOnAlarmRecordStateChange;
	//报警录像设置;
	}ST_RVS_STREAMER_CALLBACK;

返回结果	
无	

4.2.2 设备端媒体相关回调函数

本节所设置的为设备端与音、视频媒体相关的回调函数。

音视频相关回调函数定义如下：
视频码流采集任务发生变化时通知上层,上层可以暂停不必要的采集工作

回调函数定义	
typedef RVS_VOID (*PFUN_VIDEOCOLLECTFLAGNOTIFY)(RVS_UI uiCammeraId, RVS_UI uiStreamId, EN_RVS_VIDEO_COLLECT_FLAG nFlag);	
参数名称	参数说明
uiCammeraId	镜头索引，取值从 0 计数
uiStreamId	码流索引，取值从 0 计数
EN_RVS_VIDEO_COLLECT_FLAG	视频采集开关
返回结果	
无	

音频码流采集任务发生变化时通知上层,上层可以暂停不必要的采集工作

回调函数定义	
typedef RVS_VOID (*PFUN_AUDIOCOLLECTFLAGNOTIFY)(RVS_UI iMicId, EN_RVS_AUDIO_COLLECT_FLAG nFlag);	
参数名称	参数说明
iMicId	麦克风索引
EN_RVS_AUDIO_COLLECT_FLAG	音频采集开关
返回结果	
无	

通知应用层，需要编码一个关键帧

回调函数定义	
typedef RVS_VOID (*PFUN_ONKEYFRAMEREQUIRED)(RVS_UI uiCammeraId, RVS_UI uiStreamId);	
参数名称	参数说明
uiCammeraId	镜头索引，取值从 0 计数
uiStreamId	码流索引，取值从 0 计数
返回结果	
无	

从应用层获取一帧 YUV 数据

回调函数定义	
typedef RVS_VOID (*PFUN_ONGETYUV420FRAME)(RVS_UI uiCammeraId, RVS_UI uiStreamId, RVS_UC** pucFrameBuf);	
参数名称	参数说明
uiCammeraId	镜头索引，取值从 0 计数
uiStreamId	码流索引，取值从 0 计数
pucFrameBuf	YUV 数据
返回结果	
无	

观看端请求从应用层获取一个 JPEG 图片

回调函数定义

typedef RVS_VOID (*PFUN_ONGETJPEGFRAME)(RVS_UI uiCammeraId, RVS_UI uiStreamId, EN_RVS_STREAMER_JPEG_TYPE enJpegType, RVS_UC** pucFrameBuf, RVS_INT* piBufLen);	
参数名称	参数说明
uiCammeraId	镜头索引，取值从 0 计数
uiStreamId	码流索引，取值从 0 计数
enJpegType	抓图类型
pucFrameBuf	JPEG 数据
piBufLen	数据大小
返回结果	
无	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_Media_SetAVCallback(ST_RVS_STREAMER_MEDIA_AVCALLBACK stRvsAVCallback);	
参数名称	参数说明

stRvsAVCallback	typedefstructstRVS_STREAMER_MEDIA_AV_CALLBACK
	{
	PFUN_VIDEOCOLLECTFLAGNOTIFY pfunVCollectFlagNotify; //视频码流采集任务发生变化时通知上层,上层可以暂停不必要的采集工作
	PFUN_AUDIOCOLLECTFLAGNOTIFY pfunACollectFlagNotify; //音频码流采集任务发生变化时通知上层,上层可以暂停不必要的采集工作
	PFUN_ONKEYFRAMEREQUIRED pfunOnKeyFrameRequired; //需要关键帧
	PFUN_ONGETYUV420FRAME pfunOnGetOneYUV420Frame;
	PFUN_ONGETJPEGFRAME pfunOnGetOneJpegFrame;
	}ST_RVS_STREAMER_MEDIA_AVCALLBACK;

返回结果	
无	

录像文件相关回调函数定义如下：

获取录像列表

回调函数定义	
typedef RVS_INT (* GETRECORDFILELIST)(RVS_INT iCamid, RVS_INT recordType, RVS_UI tmbegin, RVS_UI tmend, RVS_INT* filecount, ST_RVS_RECORDFILELIST** recordFList);	
参数名称	参数说明
iCamid	镜头索引，取值从 0 计数
recordType	录像类型，取值 0x01 表示定时录像，取值 0x02 表示报警录像
tmbegin	录像查询开始时间
tmend	录像查询结束时间
filecount	查询到的录像文件个数
recordFList	查询到的录像文件列表

返回结果
成功返回 RVS_OK。

打开录像文件

回调函数定义	
typedef RVS_INT (* OPENRECORDFILE)(RVS_CHAR* pcFileName, void** phFileHandler, RVS_INT* pRecordedMaxFrameSize);	
参数名称	参数说明
pcFileName	文件名
phFileHandler	录像文件句柄
pRecordedMaxFrameSize	录像文件大小
返回结果	
成功返回 RVS_OK。	

关闭录像文件

回调函数定义	
typedef RVS_INT (* CLOSERECORDFILE)(RVS_VOID* phFileHandler);	
参数名称	参数说明
phFileHandler	文件句柄
返回结果	
成功返回 RVS_OK。	

获取一帧录像文件数据

回调函数定义	
typedef RVS_INT (* GETONEFRAMEFROMRECORDFILE)(void* phFileHandler, RVS_CHAR** ppcStreamBuf, RVS_INT* streamBufLen, RVS_INT* vaDatatype, RVS_UI* ptimestamp, RVS_INT* offset1, RVS_INT* offset2);	
参数名称	参数说明
phFileHandler	文件句柄
ppcStreamBuf	数据缓冲区
streamBufLen	数据长度
vaDatatype	数据类型，1 表示视频，0 表示音频
ptimestamp	帧时间戳
offset1	偏移 1
offset2	偏移 2
返回结果	
成功返回数据长度。	

获取录像文件的音视频媒体信息

回调函数定义	
typedef RVS_INT (*GETRECORDFILEMEDIAFORMAT)(void* phFileHandler, ST_RVS_STREAMER_MEDIA_AVDESC* pstMediaFormat);	
参数名称	参数说明

phFileHandler	文件句柄

pstMediaFormat	typedef struct stRVS_STREAMER_MEDIA_AVDESC
	{
	EN_RVS_STREAMER_VIDEO_TYPE enVideoType;
	RVS_INT iWidth;
	RVS_INT iHeight;
	
	EN_RVS_STREAMER_AUDIO_TYPE enAudioType;
	RVS_UI uiSampleRate;
	RVS_UI uiChannel;
	RVS_UI uiDepth;
	}ST_RVS_STREAMER_MEDIA_AVDESC;
	
	typedef enum enum_RVS_STREAMER_VIDEO_TYPE
	{
	E_RVS_STREAMER_VIDEO_TYPE_NOVIDEO = 0,
	E_RVS_STREAMER_VIDEO_TYPE_JPEG = 1,
	E_RVS_STREAMER_VIDEO_TYPE_H264 = 2,
	}EN_RVS_STREAMER_VIDEO_TYPE;
	
	typedef enum enum_RVS_STREAMER_AUDIO_TYPE
	{
	E_RVS_STREAMER_AUDIO_TYPE_NOAUDIO = 0,
	E_RVS_STREAMER_AUDIO_TYPE_AAC = 1,
	E_RVS_STREAMER_AUDIO_TYPE_G711A = 2,
	E_RVS_STREAMER_AUDIO_TYPE_G711U = 3,
	E_RVS_STREAMER_AUDIO_TYPE_PCM16 = 4,
	//用户传此格式的话，由 SDK 强制压缩成 AAC 格式传递给 CLIENT
	}EN_RVS_STREAMER_AUDIO_TYPE;

返回结果	
成功返回 RVS_OK。	

按照时间删除录像

回调函数定义	
typedef RVS_INT (* DELETERECORDFILEBYTIME)(RVS_INT recordType, RVS_INT iCamid, RVS_UI tmbegin, RVS_UI tmend);	
参数名称	参数说明
recordType	录像类型
iCamid	镜头索引值
tmbegin	开始时间
tmend	结束时间
返回结果	
成功返回 RVS_OK。	

按照录像文件名删除录像

回调函数定义	
typedef RVS_INT (* DELETERECORDFILEBYNAME)(const RVS_CHAR* filename);	
参数名称	参数说明
filename	录像文件名
返回结果	
成功返回 RVS_OK。	

录像拖动进度条

回调函数定义	
typedef RVS_INT (* SETRECORDTIME)(void* phFileHandler, RVS_UI uiSetTime);	
参数名称	参数说明
phFileHandler	文件句柄
uiSetTime	跳转时间
返回结果	
成功返回 RVS_OK。	

释放录像列表

回调函数定义	
typedef RVS_INT (* CLEARRECORDFILELIST)(ST_RVS_RECORDFILELIST** recordFList);	
参数名称	参数说明
recordFList	录像列表
返回结果	
成功返回 RVS_OK。	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_Media_SetRecordFilesCallback (ST_RVS_STREAMER_MEDIA_RECORDFILESCALLBACK* pstRvsMediaRecordFilesCallback)	
参数名称	参数说明

pstRvsMediaRecordFilesCallback	typedef struct stRVS__STREAMER_MEDIA_RECORDFILES_CALLBACK
	{
	GETRECORDFILELIST pfunGetRecordFileList; //获取录像列表
	OPENRECORDFILE pfunOpenRecordedFile; // 打开录像文件
	
	GETONEFRAMEFROMRECORDFILE
	pfunGetOneFrameFromRecordedFile; // 获取一帧录像文件数据
	CLOSERECORDFILE pfunCloseRecordedFile; // 关闭录像文件
	
	GETRECORDFILEMEDIAFORMAT
	pfunGeFileMediaFormatCB; // 获取录像文件的音视频媒体信息

	DELETERECORDFILEBYTIME
	pfunDeletRecordFileByTime; // 按照时间删除
	DELETERECORDFILEBYNAME
	pfunDeletRecordFileByName; // 按照录像名删除
	
	SETRECORDTIME pfunSetRecordTime; // 录像拖动进度条
	CLEARRECORDFILELIST pfunClearRecordFileList; //释放录像列表
	}ST_RVS_STREAMER_MEDIA_RECORDFILESCALLBACK;

返回结果	
无	

定时录制时间段设置回调

回调函数定义	
typedef RVS_INT (*PFUN_ONTIMERECORDSETTINGUPDATE)(RVS_UI uiCammeraId, RVS_UI uiScheduleCount, ST_RVS_STREAMER_TIMERECORD_SETTING* pstValue);	
参数名称	参数说明
uiCammeraId	镜头索引
uiScheduleCount	时间段个数

pstValue	typedef struct stRvsStreamerTimeRecordSetting
	{
	ST_RVS_STREAMER_SCHEDULE_SETTING schedule;
	RVS_UI uiParam;
	}ST_RVS_STREAMER_TIMERECORD_SETTING;
	
	typedef struct stRvsStreamerScheduleSetting
	{
	RVS_UI uiSeq; //定时设置序号；
	RVS_BOOL bEnable; //是否开启
	RVS_UC ucRes[3];
	RVS_UI uiStartPoint; //开始时间，换算成秒计算；
	RVS_UI uiEndPoint; //结束时间，换算成秒；
	RVS_UI uiWeekFlag; //周一到周日开启标志位；
	}ST_RVS_STREAMER_SCHEDULE_SETTING;

返回结果	
成功返回 RVS_OK。	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_Media_SetRecordSettingsCallback (PFUN_ONTIMERECORDSETTINGUPDATE pfunOnScheduledRecordSetting);	
参数名称	参数说明

pfunOnScheduledRecordSetting	
返回结果	
无	

运动侦测时间段设置回调

回调函数定义	
typedef RVS_INT (*PFUN_ONMOTIONDETECTSETTINGUPDATE)(RVS_UI uiCameraID, RVS_UI uiScheduleCount, ST_RVS_STREAMER_MOTION_DETECT_SETTING* pstValue);	
参数名称	参数说明
uiCammeraId	镜头索引
uiScheduleCount	时间段个数

pstValue	typedef struct stRvsStreamerMotionDetectSetting
	{
	ST_RVS_STREAMER_SCHEDULE_SETTING schedule;
	RVS_UI uiSensitive;
	}ST_RVS_STREAMER_MOTION_DETECT_SETTING;
	
	typedef struct stRvsStreamerScheduleSetting
	{
	RVS_UI uiSeq; //定时设置序号；
	RVS_BOOL bEnable; //是否开启
	RVS_UC ucRes[3];
	RVS_UI uiStartPoint; //开始时间，换算成秒计算；
	RVS_UI uiEndPoint; //结束时间，换算成秒；
	RVS_UI uiWeekFlag; //周一到周日开启标志位；
	}ST_RVS_STREAMER_SCHEDULE_SETTING;

返回结果	
成功返回 RVS_OK。	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_Media_SetMotionDetectSettingsCallback (PFUN_ONMOTIONDETECTSETTINGUPDATE pfunOnMotionDetectSetting);	
参数名称	参数说明
pfunOnMotionDetectSetting	
返回结果	
无	

定时录制状态回调

回调函数定义
typedef RVS_INT (*PFUN_ONTIMERECORDCHANGE)(RVS_UI uiRecordType, RVS_UI uiAlarmId, ST_RVS_RECORD_STATUS enStatus);

参数名称	参数说明
uiRecordType	录像类型
uiAlarmId	索引值
enStatus	取值为 1 表示录像开启，取值为 0 表示录像关闭
返回结果	
成功返回 RVS_OK。	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_Media_TimeRecordChangeCallback (PFUN_ONTIMERECORDCHANGE pfunOnTimeRecordChange)	
参数名称	参数说明
pfunOnTimeRecordChange	
返回结果	
无	

定制化录制开始接口：

接口定义	
RVS_INT Rvs_Streamer_StartCustomRecord(RVS_INT iCamId, RVS_INT iStreamId);	
参数名称	参数说明
iCamId	镜头索引
iStreamId	码流索引
返回结果	
成功返回 RVS_OK。	

定制化录制结束接口：

接口定义	
RVS_INT Rvs_Streamer_StopCustomRecord(RVS_INT iCamId, RVS_INT iStreamId);	
参数名称	参数说明
iCamId	镜头索引
iStreamId	码流索引
返回结果	
成功返回 RVS_OK。	

获取定制化录制时间接口：

接口定义	
RVS_INT Rvs_Streamer_GetCustomRecordTime(RVS_INT iCamId,RVS_INT *puiRecordTime);	
参数名称	参数说明
iCamId	镜头索引
puiRecordTime	任务开始到当前的时间
返回结果	
成功返回 RVS_OK。	

获取报警录制时间接口：

接口定义	
RVS_INT Rvs_Streamer_GetAlarmRecordTime(RVS_INT iCamId,RVS_INT *puiRecordTime);	
参数名称	参数说明
iCamId	镜头索引
puiRecordTime	任务开始到当前的时间
返回结果	
成功返回 RVS_OK。	

获取云存储时间接口：

接口定义	
RVS_INT Rvs_Streamer_GetCloudRecordTime(RVS_INT iCamId,RVS_INT *puiRecordTime);	
参数名称	参数说明
iCamId	镜头索引
puiRecordTime	任务开始到当前的时间
返回结果	
成功返回 RVS_OK。	

4.2.3 设备端反向码流接收回调函数

本节所设置的为设备端与反方向音、视频媒体相关的回调函数。

当观看端发送反向音频码流前，会先发送一个请求，询问采集端是否同意接受此码流

回调函数定义	
typedef RVS_INT (*PFUN_REVSTREAMSTATUS)(RVS_ULL ullPeerCid, RVS_UC* pucUrl, RVS_UI uiStatus);	
参数名称	参数说明
ullPeerCid	观看端设备的 CID 号
pucUrl	请求时携带的信息，这个用于描述此反向码流，由观看端的应用层自行定义，用于跟采集端应用层协商使用。ICHANO SDK 对具体内容不做规定。
uiStatus	0x02 表示准备发送，其他值表示结束发送；
返回结果	
若本地接受并处理此码流，则返回 0，否则返回非 0，表示不处理此码流，则观看端将不发送此路反向码流。	
重要备注	
当应用层同意了此反向码流的请求之后，调用如下接口来获取此码流的具体格式： RVS_INT Rvs_Streamer_Media_GetRevProperty(RVS_ULL ullPeerCID, _OUT ST_RVS_MEDIA_AVDESC** ppstMediaDesc);	

反向音频码流的接收回调接口

回调函数定义	
typedef RVS_VOID (*PFUN_REVSTREAMDATATOPLAY)(RVS_UC* pucRevData, RVS_INT iLength, RVS_ULL ullPeerCid, EN_RVS_MEDIA_REVDATATYPE enRevDataType);	
参数名称	参数说明
pucRevData	当前的码流数据帧
iLength	当前帧的长度
ullPeerCid	观看端 CID 号
enRevDataType	码流类型是音频还是视频 typedefenumenum_RVS_MEDIA_REVDATATYPE{ E_RVS_MEDIA_REVDATATYPE_AUDIO = 1, E_RVS_MEDIA_REVDATATYPE_VIDEO = 2 }EN_RVS_MEDIA_REVDATATYPE;
返回结果	
无	

设置接口：

接口定义	
RVS_INT Rvs_Streamer_Media_SetRevCallback(ST_RVS_STREAMER_MEDIA_REVSTREAMCALLBACK stRvsRevStreamCallback);	
参数名称	参数说明

stRvsRevStreamCallback	typedef struct stRVS_STREAMER_MEDIA_REVSTREAM_CALLBACK
	{
	PFUN_REVSTREAMSTATUS pfunOnRevStreamStatus;
	PFUN_REVSTREAMDATATOPLAY pfunOnRevStreamDataRecv;
	PFUN_REVSTREAMSTATUS pfunOnRevVideoStatus;
	//反向视频码流状态回调，该版本暂不支持直接置 NULL；
	PFUN_REVSTREAMDATATOPLAY pfunOnRevVideoDataRecv;
	//反向视频码流数据接收回调，该版本暂不支持直接置 NULL；
	}ST_RVS_STREAMER_MEDIA_REVSTREAMCALLBACK;

返回结果	
无	

获取反向码流的描述信息：

接口定义	
RVS_INT Rvs_Streamer_Media_GetRevProperty(RVS_ULL ullPeerCID, _OUT ST_RVS_STREAMER_MEDIA_AVDESC** ppstMediaDesc);	
参数名称	参数说明
ullPeerCID	输入参数，此反向码流所来自的观看端的 CID 号。
ppstMediaDesc	输出参数，此参数对反向码流中的数据进行描述。
返回结果	

无

4.2.4 设备端信令相关回调函数

本节所设置的回调接口，用于处理来自观看端的用户设置操作及信令相关的接口。
回调函数定义如下：
PTZ 控制及设备位移控制接口

ICHANO SDK 会调用此接口来将用户操作所产生的 PTZ 控制信息传给应用层，应用层应在此回调接口中实现根据 PTZ 控制信息来驱使采集端设备完成 PTZ 动作。

此接口还兼容了设备位移控制的功能，当此值取 EN_RVS_STREAMER_PTZMOVE_CTRL_MOVE 时，则后 3 个参数用于表示采集端设备移动到指定地点的空间相对坐标值。

回调函数定义	
typedef RVS_INT (*PFUN_ONPTZMOVECTRL)(RVS_ULL ullIdCid, RVS_UI uiCamIndex, EN_RVS_STREAMER_PTZMOVE_CTRL enPTZMoveCtrl, RVS_INT iPRDirect,RVS_INT iTVDirect, RVS_INT iZFDirect);	
参数名称	参数说明
ullIdCid	此操作指令所来自的观看端的 CID 号。
iCamid	镜头索引，一般情况下，一个采集端设备内只有一个镜头，此值取 0。
enPTZMoveCtrl	当此值取 EN_RVS_STREAMER_PTZMOVE_CTRL_MOVE 时，则后 3 个参数用于表示采集端设备移动到指定地点的空间相对坐标值。 当此值取 EN_RVS_STREAMER_PTZMOVE_CTRL_PTZ 时，则后 3 个参数用于表示采集端设备的 PTZ 控制参数。
iPRDirect	PTZ 控制时表示水平方向旋转幅度。位移控制时表示 X 轴参数。
iTVDirect	PTZ 控制时表示竖直方向旋转幅度。位移控制时表示 Y 轴参数。
iZFDirect	PTZ 控制时表示镜头 zoom 幅度。位移控制时表示 Z 轴参数。
返回结果	
无	

获取设备本地时间

回调函数定义	
typedef RVS_INT (*PFUN_ONGETLOCALTIME)(RVS_INT *ibsync, RVS_INT* iZone, RVS_UC* pucDatetime);	
参数名称	参数说明
ibsync	是否设置了标准时区
iZone	按照标准时区的时候，设置为哪个时区
pucDatetime	时间格式"2006-4-20-20:30"
返回结果	
无	

设置设备本地时间

ICHANO SDK 会调用此接口把用户的观看端的时间传给应用层，用于设置采集端的系统时间，应用层可根据需要进行此接口的实现和设置

回调函数定义	
typedef RVS_INT (*PFUN_ONSETLOCALTIME)(RVS_INT ibsync, RVS_INT iZone, RVS_UC* pucDateTime);	
参数名称	参数说明
ibsync	是否设置了标准时区
iZone	按照标准时区的时候，设置为哪个时区
pucDatetime	时间格式"2006-4-20-20:30"
返回结果	
无	

获取 SD 卡信息

回调函数定义	
typedef RVS_INT (*PFUN_ONGETSDCARDINFO)(RVS_INT* piTotalSize, RVS_INT* piFreeSize);	
参数名称	参数说明
piTotalSize	SD 卡总空间
piFreeSize	可用空间
返回结果	
无	

格式化 SD 卡

回调函数定义	
typedef RVS_INT (*PFUN_ONFORMATSDCARD)();	
参数名称	参数说明
返回结果	
无	

采集端闪光灯或红外灯开关

回调函数定义	
typedef RVS_INT (*PFUN_ONSWITCHTORCH)();	
参数名称	参数说明
无	
返回结果	
无	

报警输出开关是否打开

回调函数定义	
typedef RVS_INT (*PFUN_ONSWITCHALARMOUT)(RVS_BOOL bEnableAlarmOut, RVS_UI AlarmOutIndex);	
参数名称	参数说明
bEnableAlarmOut	报警开关打开还是关闭

AlarmOutIndex	报警器的索引
返回结果	
无	

邮件设置接口

回调函数定义	
typedef RVS_INT (*PFUN_ONEMAILSETTING)(RVS_BOOL bEnableEmail, RVS_UC* pucEmailAddr);	
参数名称	参数说明
bEnableEmail	邮件功能是否打开
pucEmailAddr	邮件地址
返回结果	
无	

WIFI 设置

当观看端已经连接上采集端设备[采集端设备通过有线已经连上网络]时，可以通过此信令来设置 WIFI

回调函数定义	
typedef RVS_INT (*PFUN_ONSETWIFI)(RVS_UC* pucSSID, RVS_UC* pucPwd, RVS_UI cMode, RVS_UI EncrptyType);	
参数名称	参数说明
pucSSID	WIFI 名
pucPwd	WIFI 密码
cMode	WIFI 的加密模式
EncrptyType	WIFI 的辅助加密模式
返回结果	
无	

查询当前 WIFI 状态

回调函数定义	
typedef RVS_INT (*PFUN_ONGETWIFISTATUS)(RVS_UI* puiWifiStatus);	
参数名称	参数说明
返回结果	
无	

收取自定义信令

ICHANO SDK 允许应用透传自定义信令数据。

回调函数定义	
typedef RVS_INT (*PFUN_ONRECVDATA)(RVS_ULL ullidCid, RVS_UC* pucData, RVS_UI uiLen);	
参数名称	参数说明
ullidCid	观看端的 CID 号

pucData	自定义数据内容
uiLen	自定义数据长度
返回结果	
无	

摄像头屏幕翻转设置

回调函数定义	
typedef RVS_INT (*PFUN_ONCAMERAROTATE)(RVS_ULL ullPeerCID, RVS_UI uiCamIndex, RVS_UI uiRotateType);	
参数名称	参数说明
ullPeerCID	观看端的 CID 号
uiRotateType	摄像头屏幕翻转 0x01 垂直 0x02 水平 0x03 垂直+水平
返回结果	
无	

设置接口：

接口定义	
RVS_INTRvs_Streamer_Cmd_setCallback(ST_RVS_STREAMER_CMD_CALLBACK stRvsStreamerCmdCallback);	
参数名称	参数说明

stVideoFormat	typedefstructstRVS_Streamer_Cmd_CALLBACK
	{
	PFUN_ONPTZCONTROL pfunOnPtzMoveControl;
	PFUN_ONEMAILSETTING pfunOnEmailSetting;
	//EMAIL 设置变更【含开关和 EMAIL 两个变量】
	PFUN_ONSWITCHALARMOUT pfunOnSwitchAlarmOut; //报警输出开关
	PFUN_ONGETLOCALTIME pfunOnGetLocalTime; //获取本地时间
	PFUN_ONSETLOCALTIME pfunOnSetLocalTime; //设置本地时间
	PFUN_ONGETSDCARDINFO pfunOnGetSDCardInfo; //获取 SDK 信息
	PFUN_ONFORMATSDCARD pfunOnFormatSDCard; //格式化 sdcard
	PFUN_ONSWITCHTORCH pfunOnSwitchTorch;
	//对红外灯或闪光灯进行控制的接口
	PFUN_ONSETWIFI pfunOnSetWifi; //WIFI 配置
	PFUN_ONGETWIFISTATUS pfunOnGetWifiStatus; //获取当前的 wifi 状态
	PFUN_ONRECVDATA pfunOnRecvData; //收到自定义的控制指令
	PFUN_ONCAMERAROTATE pfunOnCameraRotate; // 摄像头屏幕翻转
	}ST_RVS_STREAMER_CMD_CALLBACK;

返回结果	
无	

4.2.5 设备端移动侦测相关回调函数

本节所设置的为设备端移动侦测相关设置接口。

移动侦测处理开始回调函数。

回调函数定义	
typedef RVS_INT (* PFUN_RVSMOTIONSTART)(RVS_UI uiSensorId, _OUT RVS_HANDLE *PhHandle);	
参数名称	参数说明
uiSensorId	索引
PhHandle	此句柄由应用层自行定义,其值将会在 PFUN_RVSMOTIONPROCESS 和 PFUN_RVSMOTIONSTOP 中传递回应用层,允许为空
返回结果	
成功返回 RVS_OK。	

移动侦测处理回调函数。

回调函数定义	
typedef RVS_INT (* PFUN_RVSMOTIONPROCESS)(RVS_UI uiSensorId, RVS_HANDLE hHandle, RVS_UI uiThreshold, _OUT RVS_UI *uiOutResult);	
参数名称	参数说明
uiSensorId	索引
hHandle	PFUN_RVSMOTIONSTART 中由应用层产生的句柄,传回给应用层
uiThreshold	处理阈值
uiOutResult	处理结果
返回结果	
成功返回 RVS_OK。	

移动侦测停止回调函数。

回调函数定义	
typedef RVS_INT (* PFUN_RVSMOTIONSTOP)(RVS_UI uiSensorId, RVS_HANDLE hHandle);	
参数名称	参数说明
uiSensorId	索引
hHandle	PFUN_RVSMOTIONSTART 中由应用层产生的句柄,传回给应用层
返回结果	
成功返回 RVS_OK。	

此接口用于注册移动侦测回调。

接口定义	
RVS_INT Rvs_Streamer_Media_AddMotionPlug(ST_RVS_MOTION_PLUG stMotionPlug);	
参数名称	参数说明

stMotionPlug	typedef struct stRVS_MOTION_PLUG
	{
	RVS_CHAR cMotionName[128];//可选设置
	RVS_UI uiMotionType; //类型，一般取 EN_RVS_SENSOR_TYPE 所定义的值,支持扩展
	RVS_UI uiMotionId; //索引

	PFUN_RVSMOTIONSTART pFunStart; //开始处理函数
	PFUN_RVSMOTIONPROCESS pFunProcess; //处理函数
	PFUN_RVSMOTIONSTOP pFunStop; //结束处理函数
	
	}ST_RVS_MOTION_PLUG;

返回结果	
成功返回 RVS_OK。	

4.2.6 设备端报警传感器相关回调函数

本节所设置的为设备端报警传感器相关设置接口。

报警传感器处理开始回调函数。

回调函数定义	
typedef RVS_INT (* PFUN_RVSSENSORSTART)(RVS_UI uiSensorId, _OUT RVS_HANDLE *PhHandle);	
参数名称	参数说明
uiSensorId	传感器索引
PhHandle	此句柄由应用层自行定义,其值将会在 PFUN_RVSSENSORPROCESS 和 PFUN_RVSSENSORSTOP 中传递回应用层,允许为空
返回结果	
成功返回 RVS_OK。	

报警传感器处理回调函数。

回调函数定义	
typedef RVS_INT (* PFUN_RVSSENSORPROCESS)(RVS_UI uiSensorId, RVS_HANDLE hHandle, RVS_UI uiThreshold, _OUT RVS_UI *uiOutResult);	
参数名称	参数说明
uiSensorId	传感器索引
hHandle	PFUN_RVSSENSORSTART 中由应用层产生的句柄,传回给应用层
uiThreshold	处理阈值
uiOutResult	处理结果
返回结果	
成功返回 RVS_OK。	

报警传感器停止回调函数。

回调函数定义	
typedef RVS_INT (* PFUN_RVSSENSORSTOP)(RVS_UI uiSensorId, RVS_HANDLE hHandle);	
参数名称	参数说明
uiSensorId	传感器索引
hHandle	PFUN_RVSSENSORSTART 中由应用层产生的句柄,传回给应用层
返回结果	

成功返回 RVS_OK。

此接口用于设置报警传感器个数。

接口定义	
RVS_INT Rvs_Streamer_Sensor_SetCount(RVS_UI uiAlarmCount);	
参数名称	参数说明
uiAlarmCount	报警传感器个数
返回结果	
成功返回 RVS_OK。	

此接口用于添加一个报警传感器。

接口定义	
RVS_INT Rvs_Streamer_Sensor_AddSensorPlug(ST_RVS_SENSOR_PLUG stSensorPlug);	
参数名称	参数说明

stSensorPlug	typedef struct stRVS_SENSOR_PLUG
	{
	RVS_CHAR cSensorName[128];//可选设置
	RVS_UI uiSensorType; //传感器类型,一般取 EN_RVS_SENSOR_TYPE 所定义的值,支持扩展
	RVS_UI uiSensorId; //传感器索引，需要注意与观看端的传感器页面的对应关系
	PFUN_RVSSENSORSTART pFunStart; //开始处理函数
	PFUN_RVSSENSORPROCESS pFunProcess; //处理函数
	PFUN_RVSSENSORSTOP pFunStop; //结束处理函数
	}ST_RVS_SENSOR_PLUG;
	
	//传感器类型
	typedef enum enum_RVS_Sensor_TYPE
	{
	E_RVS_TYPE_MOTION = 1, //动作检测，由 sdk 实现
	E_RVS_TYPE_SMOG = 2, //烟雾报警
	E_RVS_TYPE_INFRARED = 3, //红外检测
	E_RVS_TYPE_DOORSENSOR = 4, //门磁检测
	E_RVS_TYPE_USER_MOTION = 5, //私有动作侦测，由第三方实现
	E_RVS_TYPE_SENSOR1 = 6, //探头 1
	E_RVS_TYPE_SENSOR2 = 7, //探头 2
	E_RVS_TYPE_SENSOR3 = 8, //探头 3
	E_RVS_TYPE_SENSOR4 = 9, //探头 4
	
	E_RVS_TYPE_INVALID = 99
	}EN_RVS_SENSOR_TYPE;

返回结果	
成功返回 RVS_OK。	

此接口用于删除一个报警传感器。

接口定义	
RVS_INT Rvs_Streamer_Sensor_DelSensorPlug(RVS_UI uiSensorType, RVS_UI uiSensorId);	
参数名称	参数说明
uiSensorType	传感器类型,一般取 EN_RVS_SENSOR_TYPE 所定义的值,支持扩展
uiSensorId	传感器索引，需要注意与观看端的传感器页面的对应关系
返回结果	
成功返回 RVS_OK。	

此接口用于发送 push。

接口定义	
RVS_INT Rvs_Streamer_Sensor_SendPush(RVS_UI uiAlarmId,RVS_UI uiAlarmType, RVS_UI uiStatus);	
参数名称	参数说明
uiAlarmId	传感器索引，与接口 Rvs_Streamer_Sensor_AddSensorPlug 中的 uiSensorId 值对应
uiAlarmType	传感器类型，一般取 EN_RVS_SENSOR_TYPE 所定义的值,支持扩展
uiStatus	报警传感器检测到的状态值（拿移动侦测来说，如果灵敏度是 50，移动侦测触发，上层检测到的侦测值可能是 65），目前是预留参数，没什么用，直接赋值为 0
返回结果	
成功返回 RVS_OK。	

4.3 登录接口

4.3.1 登录信息设置

接口定义	
_RVS_API RVS_INT Rvs_Streamer_SetLoginInfo(const RVS_UC* pucCompanyID, const RVS_UC* pucCompanyKey, const RVS_UC* pucAppID, const RVS_UC* pucLicense);	
参数名称	参数说明
pucCompanyID	加密信息，具体联系南京云恩通讯科技有限公司索取。
pucCompanyKey	加密信息，具体联系南京云恩通讯科技有限公司索取。
pucAppID	加密信息，具体联系南京云恩通讯科技有限公司索取。
pucLicense	加密信息，具体联系南京云恩通讯科技有限公司索取。
返回结果	
无	

4.3.2 启动登录

登陆的结果会在 4.2.1 的回调接口中通知给应用层。

接口定义	
RVS_INT Rvs_Streamer_Login();	
参数名称	参数说明
无	
返回结果	
无	

4.3.3 退出登录

接口定义	
RVS_INT Rvs_Streamer_Logout();	
参数名称	参数说明
无	
返回结果	
无	

4.4 发送码流接口

4.4.1 视频码流传输接口

采集端设备所采集到的视频码流，以数据帧为单位，通过调用此接口传输给 ICHANO SDK 以发往观看端。
视频码流一次传输一个视频帧，以 0001 开头，对于关键帧，则包含了 sps、pps 及关键帧的内容。
视频码流的格式应该与 4.1.9 节 Rvs_Streamer_Media_SetCameraStreamProperty 所描述的格式相符。

接口定义	
RVS_INT Rvs_Streamer_Media_WriteVideoData(RVS_UI uiCammeraId, RVS_UI uiStreamId, RVS_UC* pucData, RVS_UI uiLen, RVS_UI uiTimestamp, RVS_BOOL bIFrame);	
参数名称	参数说明

iCamid	镜头索引，取值从 0 计数。
iStreamid	码流索引，取值从 0 计数，小于 SetCameraStreamChannelCount 中所设置的码流数量。
pucData	码流数据
uiLen	码流数据长度，单位为字节。
uiTimestamp	当前数据帧的时间戳，单位毫秒
bIFrame	当前数据帧是否是 I 帧。
返回结果	
无	

4.4.2 音频码流传输

音频码流的格式应与 4.1.11 节 Rvs_Streamer_Media_SetMicProperty 所描述的格式相符。当音频码流格式为 PCM16 时，所发送的码流将会被自动压缩为 AAC 格式。

接口定义	
RVS_INT Rvs_Streamer_Media_WriteAudioData(RVS_UI iMicId, RVS_UC* pucData, RVS_UI uiLen, RVS_UI uiTimestamp);	
参数名称	参数说明
iMicId	麦克风索引
pucData	码流数据
uiLen	码流数据长度，单位为字节。
uiTimestamp	当前数据的时间戳
返回结果	
无	

4.5 其它重要接口

4.5.1 自定义信令数据发送

接口定义	
RVS_INT Rvs_Streamer_Cmd_SendData(RVS_ULL llidCid, RVS_UC *pucData, RVS_UI uiLen);	
参数名称	参数说明
pucData	自定义数据内容
uiLen	自定义数据的长度
返回结果	
无	

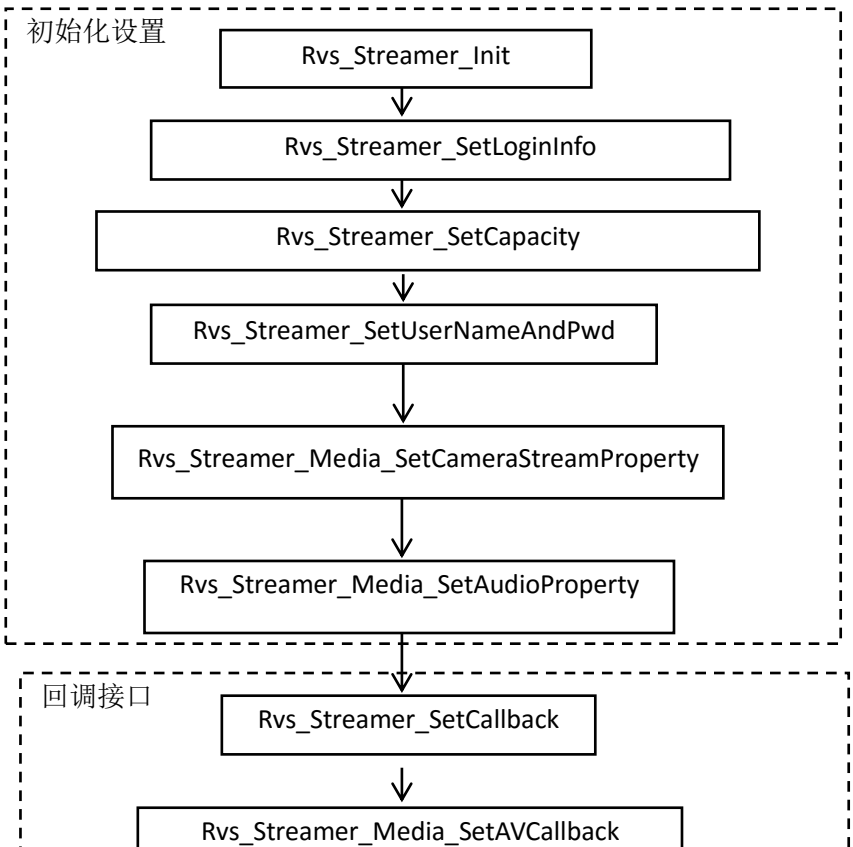
5 重要数据类型定义

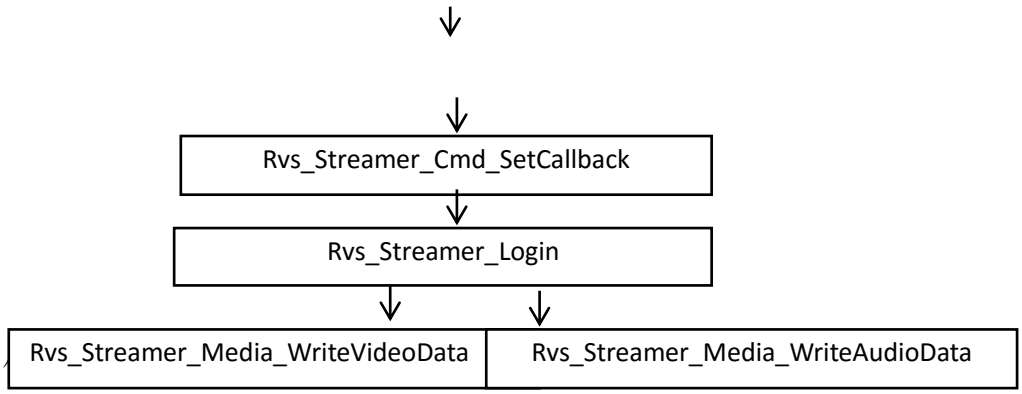
5.1.1 登录状态错误码

```
typedefenumenum_RVS_LOGIN_ERR
{
    E_RVS_LOGIN_ERR_          = 0,

    E_RVS_LOGIN_ERR_SERVICEGETERR,
    E_RVS_LOGIN_ERR_AUTH_ERRCOMPANYINFO,
    E_RVS_LOGIN_ERR_AUTH_ERRLICENSE,
    E_RVS_LOGIN_ERR_AUTH_TIMEOUT,
    E_RVS_LOGIN_ERR_CONNECT_ERR,
    E_RVS_LOGIN_ERR_REGISTER_ERR,
    E_RVS_LOGIN_ERR_ALLOCATE_ERR,
    E_RVS_LOGIN_ERR_GETSYSCONFIG_ERR,
    E_RVS_LOGIN_ERR_UPLOADINFO_ERR,
    E_RVS_LOGIN_ERR_CONNECT_INTERUPT

}EN_RVS_LOGIN_ERR;
```

6 示例代码

应用层使用 ICHANO SDK 接口，按下图所示调用步骤即可。





示例代

```
#include "rvs_streamer_type.h"
#include "rvs_streamer.h"
#include "rvs_streamer_media_type.h"
#include "rvs_streamer_media.h"
#include "rvs_streamer_cmd_type.h"
#include "rvs_streamer_cmd.h"
#include "rvs_streamer_sensor_type.h"
#include "rvs_streamer_sensor.h"
```

/*以下接口,此处只进行了部分声明，实际代码中需要由应用层来进行定义和实现，这也是应用层所需要完成的主要工作*/

```
voidmyApp_Send_TalkData(RVS_UC* pucRevData, RVS_INT iLength, RVS_ULL ullPeerCid,
EN_RVS_STREAMER_MEDIA_REVDATATYPE enRevDataType);

intmyfuncPTZControlCallback RVS_ULL ullIdCid, RVS_UI uiCamIndex,
  EN_RVS_STREAMER_PTZMOVE_CTRL enPTZMoveCtrl,
  RVS_INT iPRDirect, RVS_INT iTVDirect,RVS_INT iZFDirect);

intmyfuncWifiConfigCallback(RVS_UC* pucSSID, RVS_UC* pucPwd, RVS_UI cMode,
                                    RVS_UI EncrptyType);
intmyGetWifiStatus(RVS_UI* puiWifiStatus);
intmyfuncUserPwdSetCallback(RVS_UC* pucUserName, RVS_UC* pucPwd);
RVS_INT myRevStreamStatus(RVS_ULL ullPeerCid, RVS_UC* pucUrl, RVS_UI uiStatus);

int main(void)
{
    ST_RVS_STREAMER_CAPACITY stCapacity;
```

```
    ST_RVS_STREAMER_CALLBACK stRvsStreamerCallback;
    ST_RVS_STREAMER_CAMERASTREAM_INFO stCameraStreamInfo;
    ST_RVS_STREAMER_MIC_INFOstMicInfo;
    ST_RVS_STREAMER_MEDIA_REVSTREAMCALLBACKstRvsCallback;
    ST_RVS_STREAMER_CMD_CALLBACK        stRvsStreamerCmdCallback;
    ST_RVS_STREAMER_MEDIA_RECORDFILESCALLBACKstRvsMediaRecordFilesCallback;
    ST_RVS_SENSOR_PLUGstSensorPlug;
    ST_RVS_CAMERA_CAPACITYstCameraCapacity;

     Rvs_Streamer_Init();
stCapacity.uiRunMode = 3;
Rvs_Streamer_SetCapacity(&stCapacity);

Rvs_Streamer_SetLoginInfo((const RVS_UC*)"",(const RVS_UC*)"",
                                  (const RVS_UC*)"",(const RVS_UC*)"",(const RVS_UC*)"");

Rvs_Streamer_Login();

stRvsStreamerCallback.pfunLoginResult                          = RVS_NULL;
stRvsStreamerCallback.pfunOnSessionStateChange          = mySessionStateChange;
stRvsStreamerCallback.pfunOnUpdateCID                       = RVS_NULL;
stRvsStreamerCallback.pfunOnDeviceNameChangeByRemote = RVS_NULL;
stRvsStreamerCallback.pfunOnUpdateUserNameByRemote     = RVS_NULL;

Rvs_Streamer_SetCallback(&stRvsStreamerCallback);

    /*最大支持会话数*/
Rvs_Streamer_SetSessionMaxNum(4);
Rvs_Streamer_Media_SetCameraCount(1);

    //镜头有几路码流
Rvs_Streamer_Media_SetCameraStreamCount(0, 2);

stVideoFormat.uiWidth           = stVideoConfig.stMainVEncCtrl.Width;
stVideoFormat.uiHeight          = stVideoConfig.stMainVEncCtrl.Height;
stVideoFormat.enVideoType     = E_RVS_VIDEO_TYPE_H264;
stVideoFormat.uiBitrate         = stVideoConfig.stMainVEncCtrl.Bitrate;
stVideoFormat.uiFramerate     = stVideoConfig.stMainVEncCtrl.FrameRate;
stVideoFormat.uiIFrameInterval = stVideoConfig.stMainVEncCtrl.KeyInterval;

    //设置镜头每路码流的视频格式
Rvs_Streamer_Media_SetCameraStreamProperty(0,    0, stVideoFormat);

stVideoFormat.uiWidth               = stVideoConfig.stMinVEncCtrl.Width;
```

```
stVideoFormat.uiHeight = stVideoConfig.stMinVEncCtrl.Height;
stVideoFormat.enVideoType          = E_RVS_VIDEO_TYPE_H264;
stVideoFormat.uiBitrate = stVideoConfig.stMinVEncCtrl.Bitrate;
stVideoFormat.uiFramerate          = stVideoConfig.stMinVEncCtrl.FrameRate;
stVideoFormat.uiIFrameInterval = stVideoConfig.stMinVEncCtrl.KeyInterval;

Rvs_Streamer_Media_SetCameraStreamProperty(0,    1, &stVideoFormat);

    //设置音频格式
Rvs_Streamer_Media_SetAudioProperty(0, &stAudioFormat);

stRvsCallback.pfunOnRevStreamStatus = myRevStreamStatus;
stRvsCallback.pfunOnRevStreamDataRecv = myApp_Send_TalkData;

Rvs_Streamer_Media_SetRevCallback(&stRvsCallback);

stRvsStreamerCmdCallback.pfunOnPtzControl = myfuncPTZControlCallback;
stRvsStreamerCmdCallback.pfunOnMotionControlVDirection = RVS_NULL;
stRvsStreamerCmdCallback.pfunOnSwitchAlarmOut       = myOnSwitchAlarmOut;
stRvsStreamerCmdCallback.pfunOnGetLocalTime          = myGetSysTime;
stRvsStreamerCmdCallback.pfunOnSetLocalTime          = mySetSysTime;
stRvsStreamerCmdCallback.pfunOnGetSDCardInfo    = myfuncGetDiscInfoCallback;
stRvsStreamerCmdCallback.pfunOnFormatSDCard= myFormatDisk;
stRvsStreamerCmdCallback.pfunOnSetWifi = myfuncWifiConfigCallback;
stRvsStreamerCmdCallback.pfunOnGetWifiStatus          = myGetWifiStatus;
stRvsStreamerCmdCallback.pfunOnRecvData               = RVS_NULL;

Rvs_Streamer_Cmd_setCallback(&stRvsStreamerCmdCallback);

    Rvs_Streamer_Media_SetRecordSettingsCallback(myScheduleRecordTimeCallback);

    Rvs_Streamer_Media_SetMotionDetectSettingsCallback(myfuncAlarmSettingCallback);

stSensorPlug.pFunStart        = mySensorAlarmStart;
stSensorPlug.pFunProcess      = mySensorAlarmProcess;
stSensorPlug.pFunStop         = mySensorAlarmStop;
stSensorPlug.uiSensorId       = 0;
stSensorPlug.uiSensorType     = E_RVS_TYPE_INFRARED;
Rvs_Streamer_Sensor_AddSensorPlug(stSensorPlug);

Rvs_Streamer_SetUserNameAndPwd(ucUserName, ucPwd);
```

/*此处由应用层来实现，启动音频、视频采集和编码功能，并调用

Rvs_Streamer_Media_WriteVideoData 和 Rvs_Streamer_Media_WriteAudioData 将采集到的码流传给 ICHANO SDK。[应用层也可以在回调接口中实现启动或关闭音频、视频采集和编码功能。]*/

```
    while(1)
    {
        usleep(2000*10000);
    }
}
```

7 备注

7.1.1 登录名和密码的管理

用户在观看端上以 CID 号连接到采集端设备时，需要输入登录名和密码， ICHANO SDK 将会验证该登录名和密码，只有验证正确的情况下，才会给观看端发送所需的多媒体码流。

ICHANO SDK 用于验证的登录名和密码，由应用层通过设置给 ICHANO SDK，有两种具体情况：

1、 对于有显示和操作界面的采集端设备，用户可以在采集端设备的操作界面对登录名和密码进行设置和修改，设置和修改时，应用层调用 Rvs_Streamer_SetUserNameAndPwd 即可。[这种情况下，应用层可以不实现 4.2.1 节 pfunOnUpdateUserNameByRemote。]

2、 对于不带有显示和操作界面的采集端设备，用户无法在采集端设备上对登录名和密码进行设置或修改。这就需要采集端设备端具有两个功能：

a、支持用户在观看端进行登录名和密码的远程设置；[这就需要应用层能够提供修改登录名和密码的回调接口供 ICHANO SDK 调用，即应用层必须实现 4.2.1 节 pfunOnUpdateUserNameByRemote]

b、用户忘记登录名和密码时，可以通过恢复出厂设置的方式，恢复出厂默认的登录名和密码。[这一功能须要应用层提供]

应用层应当能获取当前最新的登录名和密码，并在 ICHANO SDK 初始化时调用 Rvs_Streamer_SetUserNameAndPwd，将所获取到的当前的登录名和密码设置给 ICHANO SDK。