NetSDK 编程指导手册
（交通分册）

V1.0.4

前言

概述

欢迎使用 NetSDK（以下简称 SDK）编程指导手册。

SDK 是软件开发者在开发网络硬盘录像机、网络视频服务器、网络摄像机、网络球机和智能设备等产品监控联网应用时的开发套件。

本文档描述了 ITC（智能交通摄像机）、ITSE（智能交通终端管理设备）、IPMECK（控制机）的通用业务涉及的 SDK 接口以及调用流程，更多功能接口、结构体等说明请参见《网络 SDK 开发手册》。

本文档提供的示例代码仅为演示接口调用方法，不保证能直接拷贝编译。

读者对象

使用 SDK 的软件开发工程师、项目经理和产品经理。

符号约定

在本文档中可能出现下列标志，代表的含义如下。

符号	说明
窍门	表示能帮助您解决某个问题或节省您的时间。
说明	表示是正文的附加信息，是对正文的强调和补充。

修订记录

版本号	修订内容	发布日期
V1.0.4	修改登录设备和搜索设备接口函数	2020.02
V1.0.3	删除 2.3.7、3.3.7 章节，修改车辆位置事件联动道闸控制业务流程图名称	2019.12
V1.0.2	新增 2.3、3.3、4.8、4.9 章节	2019.10
V1.0.1	删除表 1-1 中的一部分内容	2019.01
V1.0.0	首次发布	2017.12

名词解释

以下对本文档中使用的专业名词分别说明，帮助您更好的理解各个业务功能。

ITC	智能交通摄像机，具有抓拍车辆图片并自动分析交通事件的功能。
ITSE	智能交通终端管理设备（俗称“智能盒子”），用于连接 ITC，具有图片和已分析过的数据存储的功能。
IPMECK	道闸控制设备（一般是出入口抓拍摄像机）输出开关量信号，用于控制道闸，可开启道闸和关闭道闸。
登录句柄	向 ITC、ITSE 和 IPMECK 设备建立连接对象的句柄。如与设备成功建立连接，句柄为非空（32 位 4 字节，64 位 8 字节）。该句柄在后续各个业务中会用到，直到登出该句柄才会被清空。
视频通道	ITC 或 ITSE 的每路视频抽象成通道号的概念，单目 ITC 只有一路通道，多目和 ITSE 有多路通道。
查询句柄	向 ITSE 请求查询信息对象的句柄。如请求成功，该句柄为非空（32 位 4 字节，64 位 8 字节）。该句柄在查询具体信息业务时会用到，用完后调用关闭查询该句柄才会被清空。
智能图片	智能交通摄像机抓拍到的图片，该图片会被进行自动识别和分析。
智能抓图	某些场景用户需要手动抓图，设备把抓到的图片通过智能分析，再把分析后的数据和图片发送给用户。
智能交通事件	在车辆通过交通路口或抓拍范围时，ITC 抓拍图片，对图片进行智能分析，并将分析结果数据和图片发送给用户。
卡口	指对每一辆行驶车辆进行过车抓拍的路口。设备会对在卡口抓拍的图片进行车辆的识别分析，ITC 会把分析后的数据和图片发送给用户。
开闸	安装有 IPMECK 和道闸的路口，通过控制 IPMECK 开启道闸让车辆通行。
关闸	安装有 IPMECK 和道闸的路口，通过控制 IPMECK 关闭道闸禁止车辆通行。

目录

第 1 章 内容简介

1.1 概述

本文档主要介绍 SDK 接口参考信息，包括主要功能、接口函数和回调函数。

主要功能包括：SDK 初始化、设备初始化、设备登录、实时监视、下载智能图片、智能交通手动抓图、智能交通事件上报、车流量统计和道闸控制。

根据环境不同，开发包包含的文件会不同，具体如下所示。

- Windows 开发包所包含的文件，请参见表 1-1。

表1-1 Windows 开发包包括的文件

库类型	库文件名称	库文件说明
功能库	dhnetsdk.h	头文件
	dhnetsdk.lib	Lib 文件
	dhnetsdk.dll	库文件
	avnetsdk.dll	库文件
配置库	avglobal.h	头文件
	dhconfigsdk.h	配置头文件
	dhconfigsdk.lib	Lib 文件
	dhconfigsdk.dll	库文件
播放（编码解码）辅助库	dhplay.dll	大华播放库
	fisheye.dll	鱼眼矫正库
avnetsdk.dll 的依赖库	Infra.dll	基础库
	json.dll	json 库
	NetFramework.dll	网络基础库
	Stream.dll	媒体传输结构体封装库
	StreamSvr.dll	流服务
dhnetsdk 辅助库	IvsDrawer.dll	图像显示库

- Linux 开发包所包含的文件，请参见表 1-2。

表1-2 Linux 开发包包括的文件

库类型	库文件名称	库文件说明
功能库	dhnetsdk.h	头文件
	libdhnetsdk.so	库文件
	libavnetsdk.so	库文件
配置库	avglobal.h	头文件
	dhconfigsdk.h	配置头文件
	libdhconfigsdk.so	配置库
libavnetsdk.so 的依赖库	libInfra.so	基础库
	libNetFramework.so	网络基础库
	libStream.so	媒体传输结构体封装库
	libStreamSvr.so	流服务

说明

- SDK 的功能库和配置库是必备库。

- 功能库是设备网络 SDK 的主体，主要用于网络客户端与各类产品之间的通讯交互，负责远程控制、查询、配置及码流数据的获取和处理等。
- 配置库针对配置功能的结构体进行打包和解析。
- 推荐使用播放库进行码流解析和播放。
- 辅助库用于监视、回放、对讲等功能的音视频码流解码以及本地音频采集。
- 如果功能库包含 avnetsdk.dll 或 libavnetsdk.so，则对应依赖库是必备的。

1.2 适用性

- 推荐内存：不低于 512M。
- SDK 支持的系统如下：
  ◇ Windows
  Windows 10/Windows 8.1/Windows 7 以及 Windows Server 2008/2003。
  ◇ Linux
  Red Hat/SUSE 等通用 Linux 系统。
- 通用和卡口设备：
  ITSE1604-GN5A-D 系列、ITSE0400-GN5A-B 系列、ITSE0804-GN5B-D 系列。
- 停车场设备：
  出入口相机：ITC215-PW4I 系列、ITC215-PW5H 系列。
  出入口一体机：IPMECS-2201D、IPMECS-2001B 系列。
  车位检测相机：ITCXX4-PH 系列。

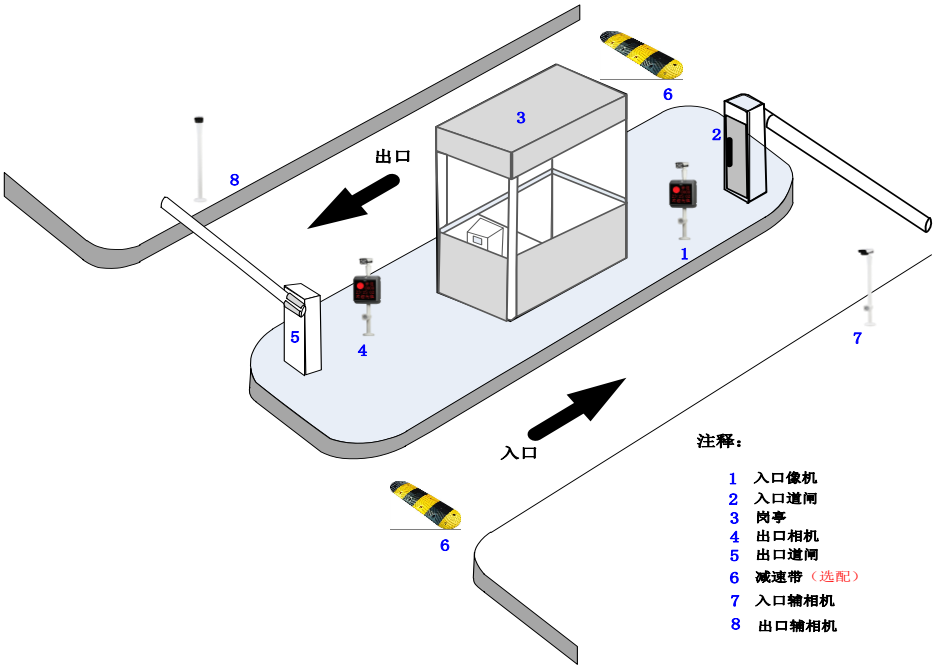
1.3 应用场景

- ITC 与 ITSE 在交通路口的应用，用于抓拍交通违章行为及车辆流量统计，如图 1-1 所示：

图1-1 ITC 与 ITSE 在交通路口的应用

- ITC、ITSE 和 IPMECK 在停车场出入口的应用，用于控制车辆进出停车场及监控车位是否有空余，如图 1-2 所示：

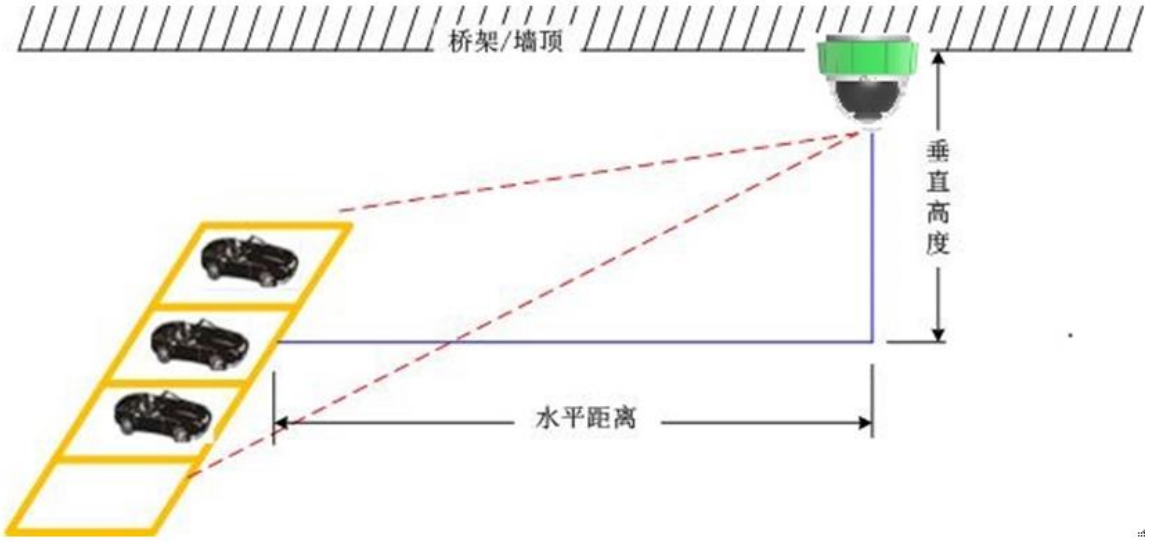
图1-2 ITC、ITSE 和 IPMECK 在停车场出入口的应用



一般情况下，用于出入口的相机（集合了 ITC 和 IPMECK 的功能），既要处理抓拍业务，又作为道闸控制设备

- ITC、ITSE 和 IPMECK 在停车场业务的应用，用于抓拍进出场车辆、监控、显示车位当前状态，如图 1-3 所示：

图1-3 ITC、ITSE 和 IPMECK 在停车场内部的应用



第 2 章 主要功能

2.1 通用

2.1.1 SDK 初始化

2.1.1.1 简介

初始化是 SDK 进行各种业务的第一步。初始化本身不包含监控业务，但会设置一些影响全局业务的参数。

- SDK 的初始化将会占用一定的内存。
- 同一个进程内，只有第一次初始化有效。
- 使用完毕后需要调用 CLIENT_Cleanup 释放资源。

2.1.1.2 接口总览

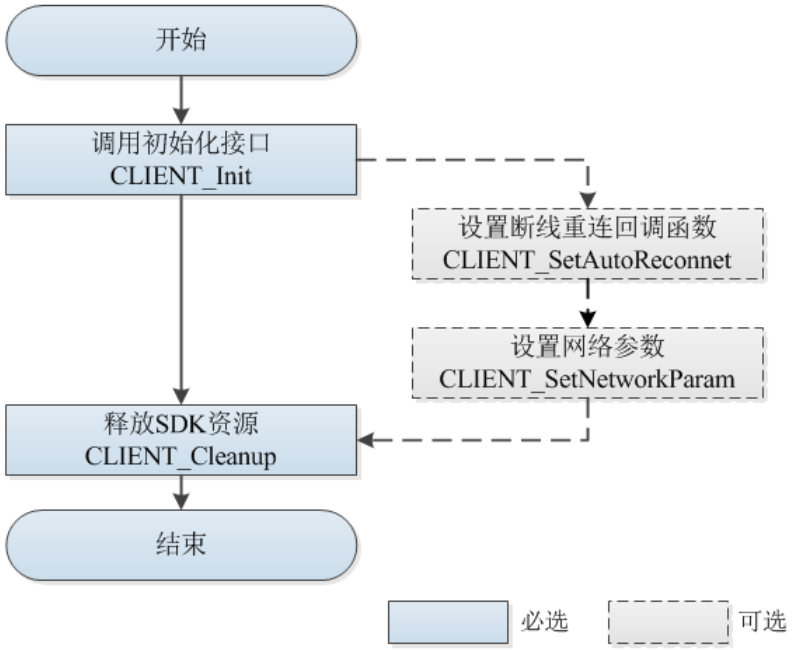
表2-1 SDK 初始化接口信息

接口	说明
CLIENT_Init	SDK 初始化接口
CLIENT_Cleanup	SDK 清理接口
CLIENT_SetAutoReconnect	设置断线重连回调接口
CLIENT_SetNetworkParam	设置网络环境接口

2.1.1.3 流程说明

SDK 初始化业务流程如图 2-1 所示。

图2-1 SDK 初始化业务流程



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

步骤2　（可选）调用 CLIENT_SetAutoReconnect 设置断线重连回调函数，设置后 SDK 内部断线自动重连。

步骤3　（可选）调用 CLIENT_SetNetworkParam 设置网络登录参数，参数中包含登录设备超时时间和尝试次数。

步骤4　SDK 所有功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- SDK 的 CLIENT_Init 和 CLIENT_Cleanup 接口需成对调用，支持单线程多次成对调用，但建议全局调用一次。
- 初始化：CLIENT_Init 接口内部多次调用时，仅在内部用做计数，不会重复申请资源。
- 清理：CLIENT_Cleanup 接口内会清理所有已开启的业务，如登录、实时监视和报警订阅等。
- 断线重连：SDK 可以设置断线重连功能，当遇到一些特殊情况（例如断网、断电等）设备断线时，在 SDK 内部会定时持续不断地进行登录操作，直至成功登录设备。断线重连后可以恢复实时监视、报警和智能图片订阅业务，其他业务无法恢复。

2.1.1.4 示例代码

```
// 通过 CLIENT_Init 设置该回调函数，当设备出现断线时，SDK 通过该函数通知用户
void CALLBACK DisConnectFunc(LLONG lLoginID, char *pchDVRIP, LONG nDVRPort, DWORD dwUser)
{
    printf("Call DisConnectFunc: lLoginID[0x%x]\n", lLoginID);
}
```

```
// 初始化 SDK
CLIENT_Init(DisConnectFunc, 0);

// .... 调用功能接口处理业务

// 清理 SDK 资源
CLIENT_Cleanup();
```

2.1.2 设备初始化

2.1.2.1 简介

设备在出厂时处于未初始化的状态，使用设备前需要初始化设备。

- 未初始化的设备不能登录。
- 初始化相当于给默认的 admin 帐户设置一个密码。
- 当忘记密码时，也可以重置密码。

2.1.2.2 接口总览

表2-2 设备初始化接口信息

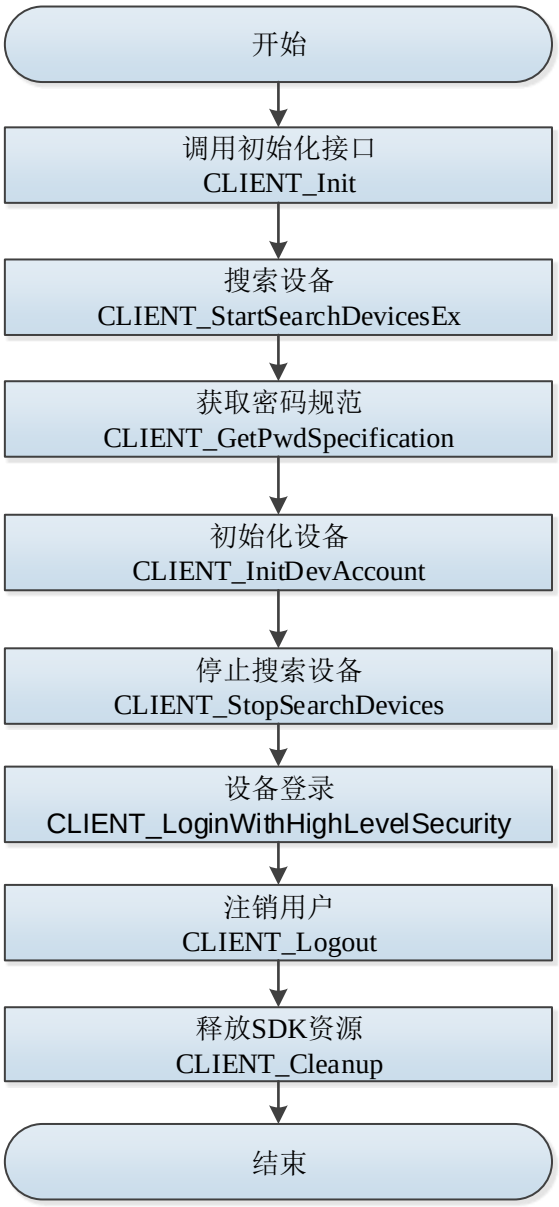
接口	说明
CLIENT_StartSearchDevicesEx	搜索局域网内的设备，找到未初始化设备
CLIENT_InitDevAccount	设备初始化接口
CLIENT_GetDescriptionForResetPwd	获取密码重置信息：手机号、邮箱和二维码信息
CLIENT_CheckAuthCode	校验安全码是否有效
CLIENT_ResetPwd	重置密码
CLIENT_GetPwdSpecification	获取密码规则
CLIENT_StopSearchDevices	停止搜索设备

2.1.2.3 流程说明

2.1.2.3.1 设备初始化

设备初始化业务流程如图 2-2 所示。

图2-2 设备初始化流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

步骤2　调用 CLIENT_StartSearchDevicesEx 搜索局域网内的设备，获取设备信息（不支持多线程调用）。

步骤3　调用 CLIENT_GetPwdSpecification 接口获取设备的密码规则，依照规则确定需要设置的密码格式。

步骤4　调用 CLIENT_InitDevAccount 初始化设备。

步骤5　调用 CLIENT_StopSearchDevices 停止设备的搜索。

步骤6　调用 CLIENT_LoginWithHighLevelSecurity，使用 admin 帐户和设置的密码登录设备。

步骤7　业务使用完后，调用 CLIENT_Logout 登出设备。

步骤8　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

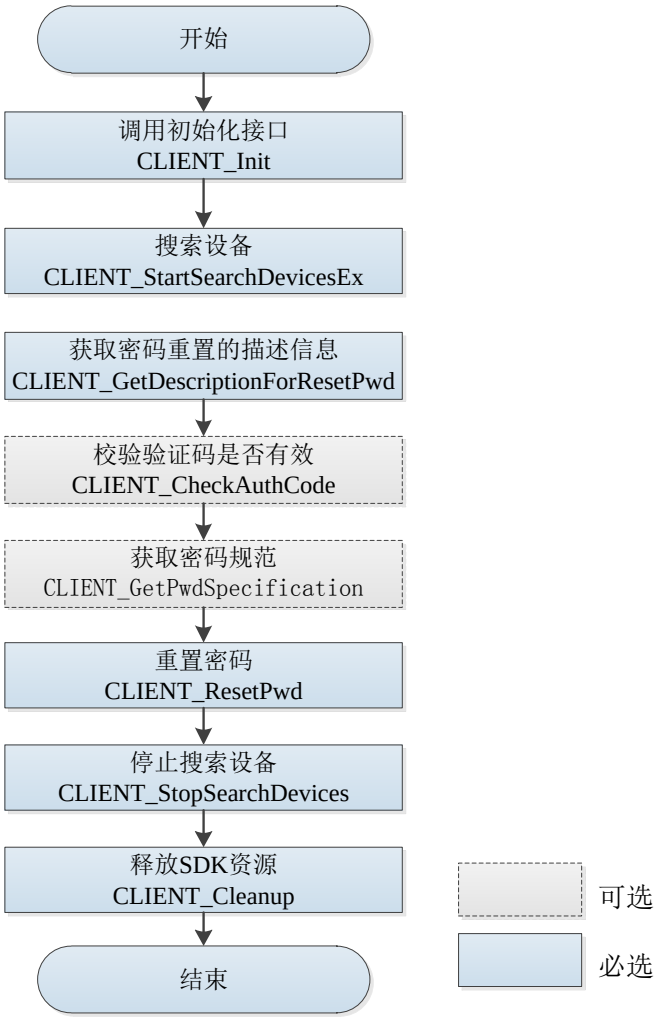
注意事项

此接口的工作方式为组播，因此主机和设备必须在同一个组播组。

2.1.2.3.2 重置密码

重置密码流程如图 2-3 所示。

图2-3 重置密码及验证流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

步骤2　调用 CLIENT_StartSearchDevicesEx 搜索局域网内的设备，获取设备信息（不支持多线程调用）。

步骤3　调用 CLIENT_GetDescriptionForResetPwd 获取重置密码的描述信息。

步骤4　（可选）指定方式扫描上一步骤中获取的二维码，获取重置密码的安全码，通过 CLIENT_CheckAuthCode 校验安全码。

步骤5　（可选）使用 CLIENT_GetPwdSpecification 获取密码规则。

步骤6　使用 CLIENT_ResetPwd 重置密码。

步骤7　调用 CLIENT_StopSearchDevices 停止设备的搜索。

步骤8　调用 CLIENT_LoginWithHighLevelSecurity，使用 admin 帐户和已重置的密码登录设备。

步骤9　业务使用完后，调用 CLIENT_Logout 登出设备。

步骤10　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

此接口的工作方式为组播，因此主机和设备必须在同一个组播组。

2.1.2.4 示例代码

2.1.2.4.1 设备初始化示例代码

```
//首先调用接口 CLIENT_StartSearchDevicesEx ，在回调函数中获取设备信息
//获取密码规则
NET_IN_PWD_SPECI stIn = {sizeof(stIn)};
strncpy(stIn.szMac, szMac, sizeof(stIn.szMac) - 1);
NET_OUT_PWD_SPECI stOut = {sizeof(stOut)};
CLIENT_GetPwdSpecification(&stIn, &stOut, 3000, NULL);//在单网卡的情况下最后一个参数可以不填；在多网卡的情况下，最后一个参数填主机 IP。可根据已获取的设备密码规则，设置符合规则的密码，此步骤主要是防止客户设置一些设备不支持的密码格式。

//设备初始化
NET_IN_INIT_DEVICE_ACCOUNT sInitAccountIn = {sizeof(sInitAccountIn)};
NET_OUT_INIT_DEVICE_ACCOUNT sInitAccountOut = {sizeof(sInitAccountOut)};
sInitAccountIn.byPwdResetWay = 1;//1 为手机号重置方式，2 为邮箱重置方式
strncpy(sInitAccountIn.szMac, szMac, sizeof(sInitAccountIn.szMac) - 1);//设置 mac
strncpy(sInitAccountIn.szUserName, szUserName, sizeof(sInitAccountIn.szUserName) - 1);//设置用户名
strncpy(sInitAccountIn.szPwd, szPwd, sizeof(sInitAccountIn.szPwd) - 1);//设置密码
strncpy(sInitAccountIn.szCellPhone, szRig, sizeof(sInitAccountIn.szCellPhone) - 1);//由于 byPwdResetWay 设置为 1，此处需要设置 szCellPhone 字段；如果 byPwdResetWay 设置为 2，则需要设置 sInitAccountIn.szMail。
CLIENT_InitDevAccount(&sInitAccountIn, &sInitAccountOut, 5000, NULL);
```

2.1.2.4.2 重置密码示例代码

```
//首先调用接口 CLIENT_StartSearchDevicesEx，在回调函数中获取设备信息
//获取密码重置的描述信息
NET_IN_DESCRIPTION_FOR_RESET_PWD stIn = {sizeof(stIn)};
strncpy(stIn.szMac, szMac, sizeof(stIn.szMac) - 1); //设置 mac 值
strncpy(stIn.szUserName, szUserName, sizeof(stIn.szUserName) - 1);//设置用户名
stIn.byInitStatus = bStstus; //bStstus 为搜索设备接口(CLIENT_SearchDevices、CLIENT_StartSearchDevices、CLIENT_StartSearchDevicesEx 的回调函数和 CLIENT_SearchDevicesByIPs)返回字段 byInitStatus 的值
NET_OUT_DESCRIPTION_FOR_RESET_PWD stOut = {sizeof(stOut)};
char szTemp[360];
```

stOut.pQrCode = szTemp;

CLIENT_GetDescriptionForResetPwd(&stIn, &stOut, 3000, NULL);//在单网卡的情况下最后一个参数可以不填；在多网卡的情况下，最后一个参数填主机 IP。接口执行成功后，stOut 会输出一个二维码，二维码信息地址为 stOut.pQrCode，扫描此二维码，获取重置密码的安全码，此安全码会发送到预留手机号或者邮箱里

//(可选)校验安全码

NET_IN_CHECK_AUTHCODE stIn1 = {sizeof(stIn1)};

strncpy(stIn1.szMac, szMac, sizeof(stIn1.szMac) - 1); //设置 mac

strncpy(stIn1.szSecurity, szSecu, sizeof(stIn1.szSecurity) - 1); // szSecu 为上一步骤中发送到预留手机号或者邮箱里的安全码

NET_OUT_CHECK_AUTHCODE stOut1 = {sizeof(stOut1)};

bRet = CLIENT_CheckAuthCode(&stIn1, &stOut1, 3000, NULL); //在单网卡的情况下最后一个参数可以不填；在多网卡的情况下，最后一个参数填主机 IP

//获取密码规则

NET_IN_PWD_SPECI stIn2 = {sizeof(stIn2)};

strncpy(stIn2.szMac, szMac, sizeof(stIn2.szMac) - 1); //设置 mac

NET_OUT_PWD_SPECI stOut2 = {sizeof(stOut2)};

CLIENT_GetPwdSpecification(&stIn2, &stOut2, 3000, NULL);//在单网卡的情况下最后一个参数可以不填；在多网卡的情况下，最后一个参数填主机 IP。获取成功的情况下，可根据获取出的设备密码规则设置符合规则的密码，此步骤主要是防止客户设置一些设备不支持的密码格式

//重置密码

NET_IN_RESET_PWD stIn3 = {sizeof(stIn3)};

strncpy(stIn3.szMac, szMac, sizeof(stIn3.szMac) - 1); //设置 mac 值

strncpy(stIn3.szUserName, szUserName, sizeof(stIn3.szUserName) - 1); //设置用户名

strncpy(stIn3.szPwd, szPassWd, sizeof(stIn3.szPwd) - 1); //szPassWd 为符合密码规则的重置密码

strncpy(stIn3.szSecurity, szSecu, sizeof(stIn1.szSecurity) - 1); // szSecu 为扫描二维码后发送到预留手机号或者邮箱里的安全码

stIn3.byInitStaus = bStstus; //bStstus 为搜索设备接口(CLIENT_SearchDevices、CLIENT_StartSearchDevices、CLIENT_StartSearchDevicesEx 的回调函数和 CLIENT_SearchDevicesByIPs)返回字段 byInitStatus 的值

stIn3.byPwdResetWay = bPwdResetWay; //bPwdResetWay 为搜索设备接口(CLIENT_SearchDevices、CLIENT_StartSearchDevices、CLIENT_StartSearchDevicesEx 的回调函数和 CLIENT_SearchDevicesByIPs)返回字段 byPwdResetWay 的值

NET_OUT_RESET_PWD stOut3 = {sizeof(stOut3)};

CLIENT_ResetPwd(&stIn3, &stOut3, 3000, NULL);// 在单网卡的情况下最后一个参数可以不填；在多网卡的情况下，最后一个参数填主机 IP

2.1.3 设备登录

2.1.3.1 简介

设备登录，即用户鉴权，是进行其他业务的前提。

用户登录设备产生唯一的登录 ID，其他功能的 SDK 接口需要传入登录 ID 才可执行。登出设备后，登录 ID 失效。

2.1.3.2 接口总览

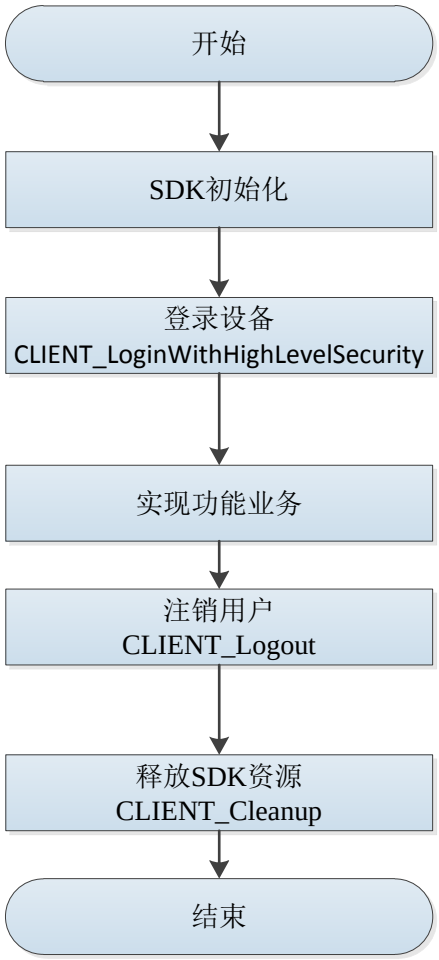
表2-3 设备登录接口信息

接口	说明
CLIENT_LoginWithHighLevelSecurity	高安全级别登录接口。 **说明** CLIENT_LoginEx2 仍然可以使用，但存在安全风险。所以强烈推荐使用最新接口 CLIENT_LoginWithHighLevelSecurity 登录设备。
CLIENT_Logout	登出接口

2.1.3.3 流程说明

登录业务流程如图 2-4 所示。

图2-4 登录业务流程



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　登录成功后，用户可以实现需要的业务功能。
步骤4　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- 登录句柄：登录成功时接口返回值非 0（即句柄可能小于 0，也属于登录成功）；同一设备登录多次，每次的登录句柄不一样。如果无特殊业务，建议只登录一次，登录的句柄可以重复用于其他各种业务。
- 登出：接口内部会释放登录会话中已打开的业务，但建议用户不要依赖登出接口的清理功能。例如打开监视后，在不需要使用监视时，用户应该调用结束监视的接口。
- 登录与登出配对使用，登录会消耗一定的内存和 socket 信息，在登出后释放资源。
- 登录失败：建议通过登录接口的 error 参数（登录错误码）初步排查。常见错误码如表 2-4 所示。

表2-4 常见错误码

error 的错误码	对应的含义
1	密码不正确
2	用户名不存在
3	登录超时
4	账号已登录
5	账号已被锁定
6	账号被列为黑名单
7	资源不足，设备系统忙
8	子连接失败
9	主连接失败
10	超过最大用户连接数
11	缺少 avnetsdk 或 avnetsdk 的依赖库
12	设备未插入 U 盘或 U 盘信息错误
13	客户端 IP 地址没有登录权限

更多错误码信息请参见《网络 SDK 开发手册》中的“CLIENT_LoginWithHighLevelSecurity 接口”描述。其中错误码 3 规避示例代码如下：

```
NET_PARAM stuNetParam = {0};
stuNetParam.nWaittime = 8000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

2.1.3.4 示例代码

```
NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY stInparam;
memset(&stInparam, 0, sizeof(stInparam));
stInparam.dwSize = sizeof(stInparam);
strncpy(stInparam.szIP, “192.168.1.108”, sizeof(stInparam.szIP) - 1);
strncpy(stInparam.szPassword, “123456”, sizeof(stInparam.szPassword) - 1);
strncpy(stInparam.szUserName, “admin”, sizeof(stInparam.szUserName) - 1);
stInparam.nPort = 37777;
stInparam.emSpecCap = EM_LOGIN_SPEC_CAP_TCP;

NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY stOutparam;
memset(&stOutparam, 0, sizeof(stOutparam));
stOutparam.dwSize = sizeof(stOutparam);
LLONG lLoginHandle = CLIENT_LoginWithHighLevelSecurity(&stInparam, &stOutparam);
// 登出设备
if (0 != lLoginHandle)
{
    CLIENT_Logout(lLoginHandle);
}
```

2.1.4 实时监视

2.1.4.1 简介

实时监视，即向存储设备或前端设备获取实时码流的功能，是监控系统的重要组成部分。

SDK 登录设备后，可向设备获取主码流和辅码流。

- 支持用户传入窗口句柄，SDK 直接进行码流解析及播放（此功能仅限 Windows 版本）。
- 支持回调实时码流数据给用户，让用户自己处理。
- 支持保存实时录像到指定文件，用户可通过自行保存回调码流实现，也可以通过调用 SDK 接口实现。

2.1.4.2 接口总览

表2-5 实时监视接口信息

接口	说明
CLIENT_RealPlayEx	开始实时监视扩展接口
CLIENT_StopRealPlayEx	停止实时监视扩展接口
CLIENT_SaveRealData	开始本地保存实时监视数据
CLIENT_StopSaveRealData	停止本地保存实时监视数据
CLIENT_SetRealDataCallBackEx2	设置实时监视数据回调函数扩展接口

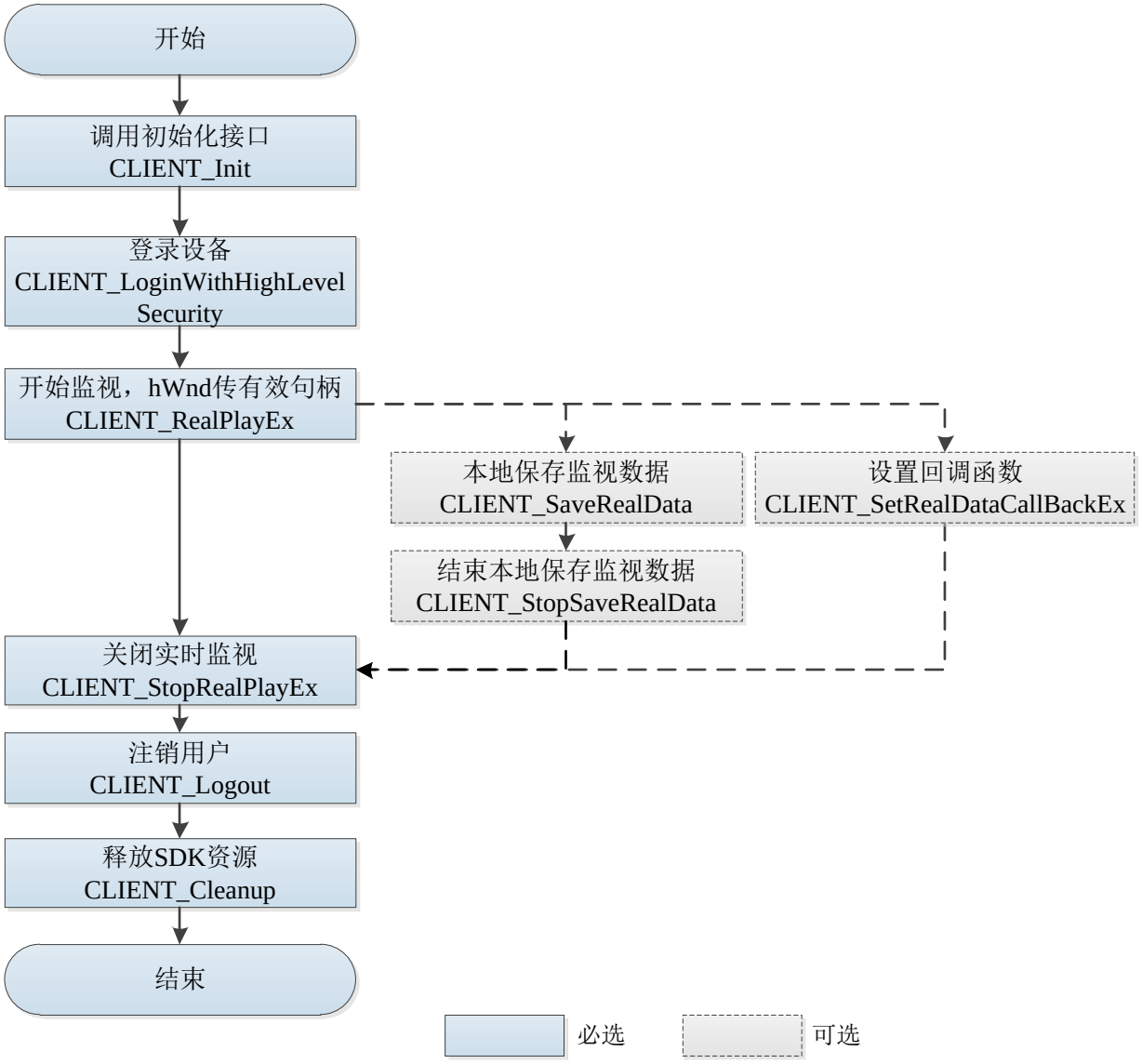
2.1.4.3 流程说明

实时监控的实现方式有两种，分别为 SDK 集成播放库进行播放及用户自己调用播放库播放码流方式进行播放。

2.1.4.3.1 SDK 集成播放库播放

SDK 内部调用辅助库里的 PlaySDK 库实现实时播放。SDK 集成播放库解码播放流程如图 2-5 所示。

图2-5 SDK 解码播放流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　调用 CLIENT_RealPlayEx 启动实时监视，参数 **hWnd** 为有效窗口句柄。
步骤4　（可选）调用 CLIENT_SaveRealData 开始保存监视数据。
步骤5　（可选）调用 CLIENT_StopSaveRealData 结束保存，生成本地视频文件。
步骤6　（可选）若调用 CLIENT_SetRealDataCallBackEx2，用户可将视频数据选择保存或转发。若保存成文件，与步骤 4、5 效果相同。
步骤7　实时监视使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时监视。
步骤8　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤9　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- SDK 解码播放只支持 Windows 系统，非 Windows 系统需要用户获取码流后自己调用解码

显示。

- 多线程调用：同一个登录会话内的业务，不支持多线程调用；但可以多个线程处理不同的登录会话中的业务，但不建议这样调用。
- 超时：接口内申请监视资源需和设备做一些约定，然后才请求监视数据，过程中有一些超时时间的设定（请参见NET_PARAM结构体），其中与监视相关的字段为**nGetConnInfoTime**。如果实际使用中（如网络状况不良）有超时现象，可将 **nGetConnInfoTime** 的值修改大一些。示例代码如下，在 CLIENT_Init 函数后调用，调用一次即可：

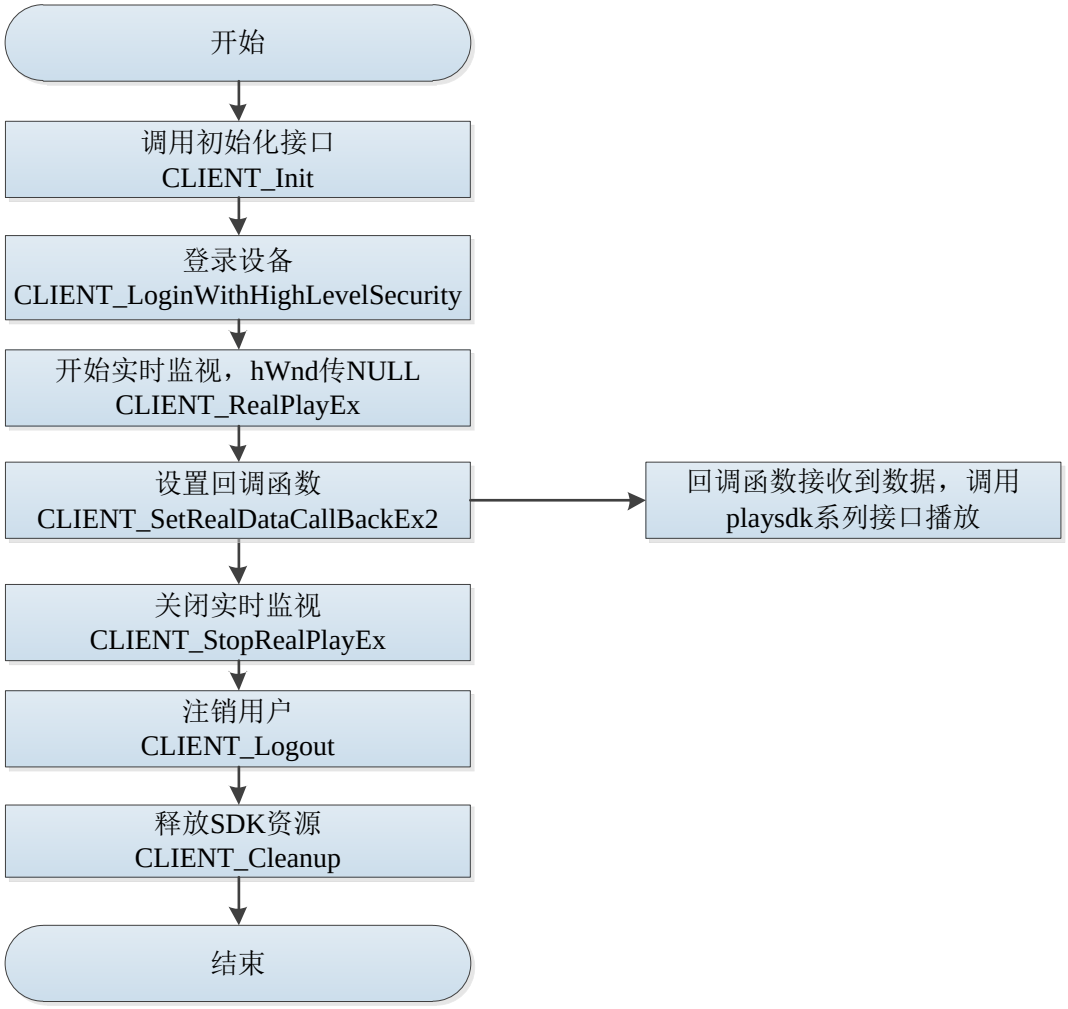
```
NET_PARAM stuNetParam = {0};
stuNetParam. nGetConnInfoTime = 5000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- 重复打开失败：部分设备不支持同一次登录下同一个通道多次打开，当重复打开同一通道的监视，可能会出现第一次打开成功，后续打开失败的现象。建议：
  - ◇ 将已打开的通道先关闭。例如已经开启通道一的主码流视频，希望再打开通道一的辅码流视频时，可先关闭通道一的主码流视频，再开启通道一的辅码流视频。
  - ◇ 登录两次设备获取两个登录句柄，分别处理主码流和辅码流业务。
- 接口成功无画面：SDK 内部解码需要使用到 dhplay.dll，建议查看运行目录下是否缺少 dhplay.dll 及其依赖的辅助库，具体请参见表 1-1。
- 系统资源不足的情况下，设备可能返回错误而不恢复码流，可以在报警回调函数（即 CLIENT_SetDVRMessCallBack 中设置的回调函数）收到事件 DH_REALPLAY_FAILD_EVENT，该事件包含了详细的错误码，请参见《网络 SDK 开发手册》中的“DEV_PLAY_RESULT 结构体”。
- 32 路限制：解码显示比较消耗资源，特别是高分辨率视频，考虑到客户端硬件资源有限，一般同时解码显示的通道数有限，所以该方式暂时限定为最多 32 路，如超过 32 路，建议使用“2.1.4.3.2 调用第三方解码播放库”。

2.1.4.3.2 调用第三方解码播放库

SDK 回调实时监视码流给用户，用户调用 PlaySDK 进行解码播放。用户调用第三方解码播放程如图 2-6 所示。

图2-6 第三方解码播放流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　登录成功后，调用 CLIENT_RealPlayEx 启动实时监视，参数 **hWnd** 为 **NULL**。
步骤4　调用 CLIENT_SetRealDataCallBackEx2 设置实时数据回调函数。
步骤5　在回调函数中将数据传给 PlaySDK 完成解码。
步骤6　实时监视使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时监视。
步骤7　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤8　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- 码流格式：推荐使用 PlaySDK 解码。
- 画面卡顿：
  - ◇ 使用 PlaySDK 解码时，解码通道缓存大小有默认值（PlaySDK 中的 PLAY_OpenStream 接口）。如果码流的分辨率很大，建议修改参数值，例如改为 3M。
  - ◇ SDK 回调函数需用户返回后才能回调下一段，建议用户在回调中不要做耗时操作，否则会严重影响性能。

2.1.4.4 示例代码

2.1.4.4.1 SDK 解码播放

```
//以开启第一路的主码流监视为例，hWnd 为界面窗口句柄
LLONG lRealHandle = CLIENT_RealPlayEx(lLoginHandle, 0, hWnd, DH_RType_Realplay);
if (NULL == lRealHandle)
{
    printf("CLIENT_RealPlayEx: failed! Error code: %x.\n", CLIENT_GetLastError());
}
printf("input any key to quit!\n");
getchar();
// 关闭预览
if (NULL != lRealHandle)
{
    CLIENT_StopRealPlayEx(lRealHandle);
}
```

2.1.4.4.2 调用播放库

```
void CALLBACK RealDataCallBackEx(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer,
DWORD dwBufSize, LLONG param, LDWORD dwUser);
//以开启第一路的主码流监视为例
LLONG lRealHandle = CLIENT_RealPlayEx(lLoginHandle, 0, NULL, DH_RType_Realplay);
if (NULL == lRealHandle)
{
    printf("CLIENT_RealPlayEx: failed! Error code: %x.\n", CLIENT_GetLastError());
}
else
{
    DWORD dwFlag = REALDATA_FLAG_RAW_DATA; //原始数据标志
    CLIENT_SetRealDataCallBackEx2(lRealHandle, &RealDataCallBackEx, NULL, dwFlag);
}

printf("input any key to quit!\n");
getchar();
// 关闭预览
if (0 != lRealHandle)
{
    CLIENT_StopRealPlayEx(lRealHandle);
}
```

```
void CALLBACK RealDataCallBackEx(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer,
DWORD dwBufSize, LLONG param, LDWORD dwUser)
{
    // 从设备获取的码流数据，需调用 PlaySDK 的接口，详见 SDK 监视 demo 源码
    printf("receive real data, param: lRealHandle[%p], dwDataType[%d], pBuffer[%p], dwBufSize[%d]\n",
lRealHandle, dwDataType, pBuffer, dwBufSize);
}
```

2.2 卡口

2.2.1 下载智能图片

2.2.1.1 简介

下载智能图片，即用户通过SDK获取ITSE上存有的已智能分析过的图片，并保存到本地的过程。用户可将本地的智能图片做更进一步的应用。

下载智能图片实现方式为 SDK 主动连接设备，先按智能图片的查询条件发送查询命令，查询到结果后，再发命令下载智能图片，设备收到命令后把智能图片和分析的数据发送给用户。

2.2.1.2 接口总览

表2-6 下载智能图片的接口信息

必选接口	说明
CLIENT_FindFileEx	按智能图片条件查询
CLIENT_GetTotalFileCount	获取查询到的数量
CLIENT_FindNextFileEx	查询智能图片信息
CLIENT_FindCloseEx	关闭查询
CLIENT_DownloadMediaFile	下载智能图片信息
CLIENT_StopDownloadMediaFile	关闭下载

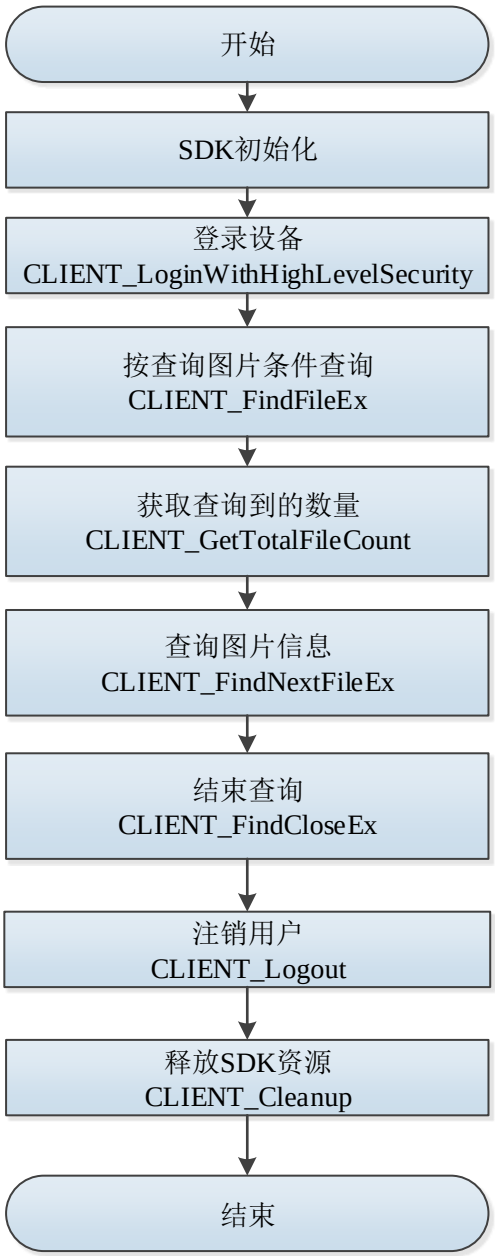
2.2.1.3 流程图

流程分为查询和下载智能图片信息两部分。

2.2.1.3.1 查询

查询智能图片信息流程如图 2-7 所示。

图2-7 查询智能图片业务流程



流程说明

步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。
步骤3　调用 CLIENT_FindFileEx 函数按查询图片条件进行查询。
步骤4　查询完成后，调用 CLIENT_GetTotalFileCount 函数获取查询到的总数。
步骤5　通过得到的总数调用 CLIENT_FindNextFileEx 函数遍历每个图片信息。
步骤6　查询结束后，调用 CLIENT_FindCloseEx 函数关闭查询。
步骤7　业务使用完后，调用 CLIENT_Logout 函数登出设备。
步骤8　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

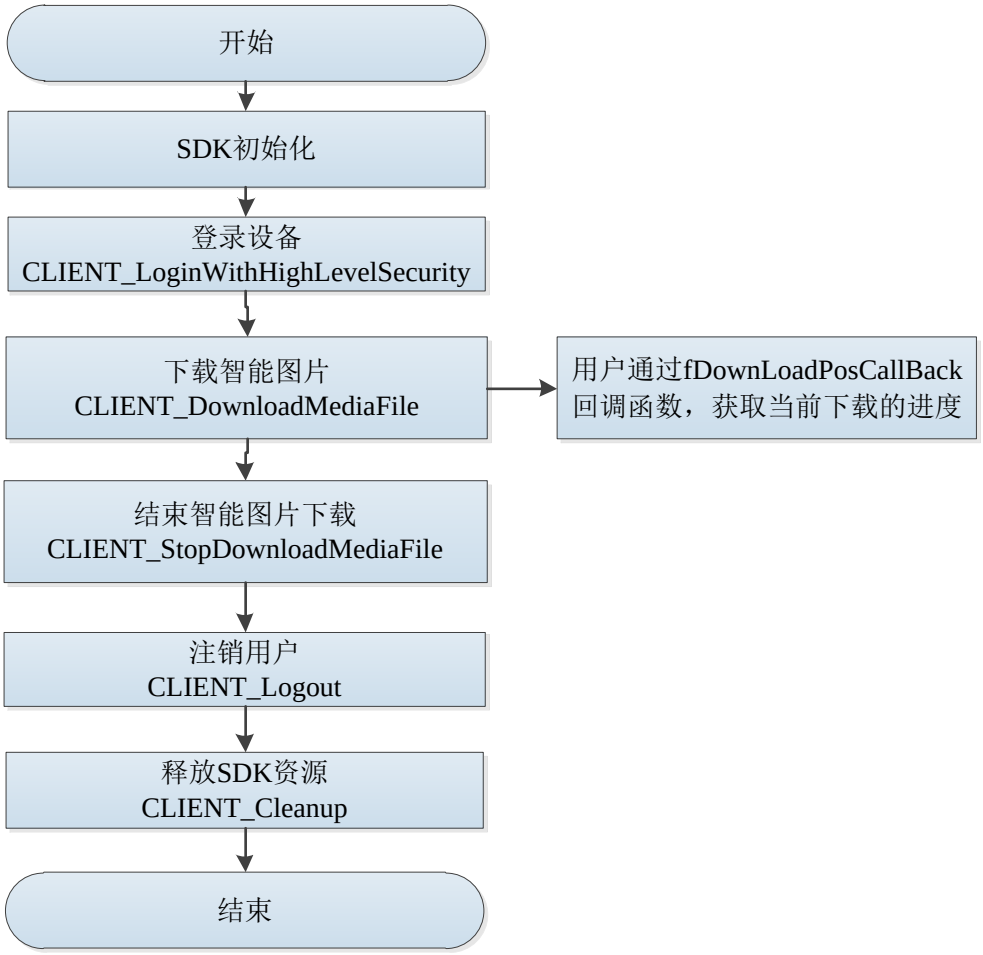
- 适用设备：ITSE 设备。一般情况 ITC 是没有存储数据的功能，只做抓拍和识别的功能。

- 参数： CLIENT_FindFileEx 中参数 emType 用 DH_FILE_QUERY_TRAFFICCAR_EX，对应的结构体使用 MEDIA_QUERY_TRAFFICCAR_PARAM_EX；CLIENT_FindNextFileEx 接口中对应的结构体使用 MEDIAFILE_TRAFFICCAR_INFO_EX。

2.2.1.3.2 下载

下载智能图片信息流程如图 2-8 所示。

图2-8 下载智能图片业务流程



流程说明

步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。
步骤3　调用 CLIENT_DownloadMediaFile 函数下载智能图片。
步骤4　下载完成后，调用 CLIENT_StopDownloadMediaFile 函数关闭下载。
步骤5　业务使用完后，调用 CLIENT_Logout 函数登出设备。
步骤6　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

- 适用设备： ITSE 设备。一般情况 ITC 是没有存储数据的功能，只做抓拍和识别的功能。
- 参数： CLIENT_DownloadMediaFile 中参数 emType 只能用 DH_FILE_QUERY_TRAFFICCAR，不支持 DH_FILE_QUERY_TRAFFICCAR_EX；参数 lpMediaFileInfo 可通过查询智能图片获取。

2.2.1.4 示例代码

2.2.1.4.1 查询

查询智能图片主要示例代码如下所示：

```
int main()
{
    ……
    //查询智能图片的查询条件
    MEDIA_QUERY_TRAFFICCAR_PARAM_EX stuCondition = {0};
    stuCondition.dwSize = sizeof(MEDIA_QUERY_TRAFFICCAR_PARAM_EX);
    stuCondition.stuParam.nMediaType = 1;
    ……
    //查询智能图片
    LLONG lFindHandle = CLIENT_FindFileEx(lLoginHandle, DH_FILE_QUERY_TRAFFICCAR_EX, (void*)&stuCondition, NULL);
    if(NULL == lFindHandle)
    {
        printf("CLIENT_FindFileEx: failed! Error code: %x.\n", CLIENT_GetLastError());
        return -1;
    }
    int nCount = 0;
    //获取查询到的智能图片个数
    BOOL bRet = CLIENT_GetTotalFileCount(lFindHandle,&nCount,NULL);
    if(FLASE == bRet)
    {
        printf("CLIENT_GetTotalFileCount: failed! Error code: %x.\n", CLIENT_GetLastError());
        return -2;
    }
    //一次一个遍历查询到智能图片信息。
    int nMaxConut = 1;
    do
    {
        MEDIAFILE_TRAFFICCAR_INFO_EX mediaFileInfo = {0};
        mediaFileInfo.dwSize = sizeof(MEDIAFILE_TRAFFICCAR_INFO_EX);
        //查询单个智能图片信息
        bRet = CLIENT_FindNextFileEx(lFindHandle, nMaxConut, (void*)&mediaFileInfo, sizeof(MEDIAFILE_TRAFFICCAR_INFO_EX), NULL);
        if(FALSE == bRet)
        {
```

```
            printf("CLIENT_FindNextFileEx: failed! Error code: %x.\n", CLIENT_GetLastError());
        }
    }
    While ((nCount -= nMaxConut) > 0);
    //关闭查询
    bRet = CLIENT_FindCloseEx(lFindHandle);
  if(FALSE == bRet)
  {
      printf("CLIENT_FindCloseEx: failed! Error code: %x.\n", CLIENT_GetLastError());
      return -3;
  }
}
```

2.2.1.4.2 下载

下载智能图片主要示例代码如下所示：

```
int main()
{
    ……
    //查询得到的智能图片信息
    MEDIAFILE_TRAFFICCAR_INFO info = mediaFileInfo.stuInfo;
    //下载智能图片
    LLONG lDownloadHandle = CLIENT_DownloadMediaFile(lLoginHandle, DH_FILE_QUERY_TRAFFICCAR, (void*)&info, szFileName, DownLoadPosCallBack, NULL, NULL);
    if(NULL == lDownloadHandle)
    {
        printf("CLIENT_DownloadMediaFile: failed! Error code: %x.\n", CLIENT_GetLastError());
    }
    Sleep(5000);
    //关闭下载
    BOOL bRet = CLIENT_StopDownloadMediaFile(lDownloadHandle);
    if(FALSE == bRet)
    {
        printf("CLIENT_StopDownloadMediaFile: failed! Error code: %x.\n", CLIENT_GetLastError());
    }
}
//下载进度回调
void CALLBACK DownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, LDWORD dwUser)
```

```
{
    if (dwDownLoadSize == -1) //表示下载结束
    {
        printf("lPlayHandle: %p Download end!\n", lPlayHandle);
     }
}
```

2.2.2 智能交通手动抓图

2.2.2.1 简介

智能交通手动抓图，即用户通过 SDK 下发命令给 ITC 或 ITSE 设备，通知设备抓拍图片，设备在抓拍到图片后自动分析图片并上报给用户的过程。

主要应用于用户需要分析车辆相关的信息，检测车辆是否有违章的行为或保存车辆信息等场景。

2.2.2.2 接口总览

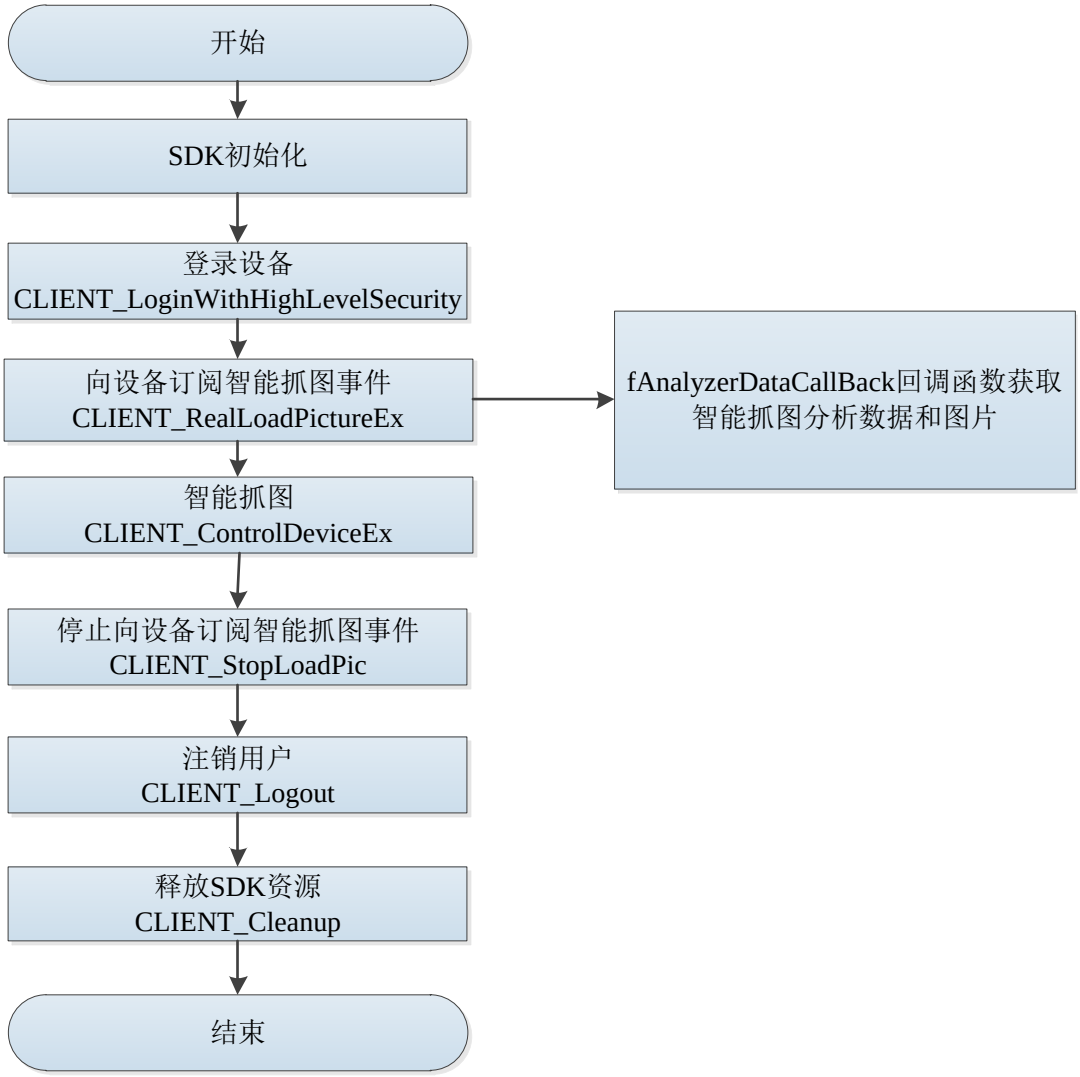
表2-7 智能交通手动抓图的接口信息

接口	说明
CLIENT_RealLoadPictureEx	订阅智能交通事件
CLIENT_ControlDeviceEx	智能交通手动抓图
CLIENT_StopLoadPic	取消订阅智能交通事件

2.2.2.3 流程图

智能交通手动抓图流程如图 2-9 所示。

图2-9 智能交通手动抓图业务流程



流程说明

步骤1 调用 CLIENT_Init 函数完成 SDK 初始化流程。

步骤2 初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。

步骤3 调用 CLIENT_ RealLoadPictureEx 函数向设备订阅智能交通事件。

步骤4 调用 CLIENT_ControlDeviceEx 函数触发智能抓图，参数 emType 值设置为 DH_MANUAL_SNAP。

步骤5 智能交通手动抓图事件通过 fAnalyzerDataCallBack 函数设置的回调函数通知用户。

步骤6 智能交通手动抓图功能使用完毕后，调用 CLIENT_StopLoadPic 函数停止订阅智能交通事件。

步骤7 业务使用完后，调用 CLIENT_Logout 函数登出设备。

步骤8 SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

2.2.2.4 示例代码

```
int main()
{
    ……
```

```
    //订阅智能抓图事件
    LLONG lAnalyerHandle = CLIENT_RealLoadPictureEx(lLoginHandle, 0,
(DWORD)EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, NULL, NULL);
    if(NULL == lAnalyerHandle)
    {
        printf("CLIENT_RealLoadPictureEx: failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    MANUAL_SNAP_PARAMETER stuManualSnap = {0};
    stuManualSnap.nChannel = 0;
    sprintf((char*)stuManualSnap.bySequence,"abc");
    //智能抓图
    BOOL bRet = CLIENT_ControlDeviceEx(lLoginHandle,DH_MANUAL_SNAP,&stuManualSnap);
    if(FALSE == bRet)
    {
        printf("CLIENT_ControlDeviceEx: failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    Sleep(5000);
    //取消订阅智能抓图事件
    BOOL bRet = CLIENT_StopLoadPic(lAnalyerHandle);
    if(FALSE == bRet)
    {
        printf("CLIENT_StopLoadPic: failed! Error code %x.\n", CLIENT_GetLastError());
        return -3;
    }
    return 0;
}
//智能抓图回调
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void*
pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    switch(dwAlarmType)
    {
        case EVENT_IVS_TRAFFIC_MANUALSNAP:
            {
                DEV_EVENT_TRAFFIC_MANUALSNAP_INFO* pInfo =
                (DEV_EVENT_TRAFFIC_MANUALSNAP_INFO*)pAlarmInfo;
                printf("ManualSnapNo: %s", (char*)pInfo-> szManualSnapNo);
                    ……
```

```
                    break;
                }
            default:
                break;
        }
        return 0;
}
```

2.2.3 智能交通事件上报

2.2.3.1 简介

智能交通事件上报，即智能交通设备对实时码流进行分析，当检测到预先设定好的交通事件时，将交通事件发送给用户。智能交通事件有交通违章、停车场有无车位等事件。

智能交通事件上报实现方式为 SDK 主动连接设备，并向设备订阅智能事件功能，设备检测到智能事件立即发送给 SDK。

2.2.3.2 接口总览

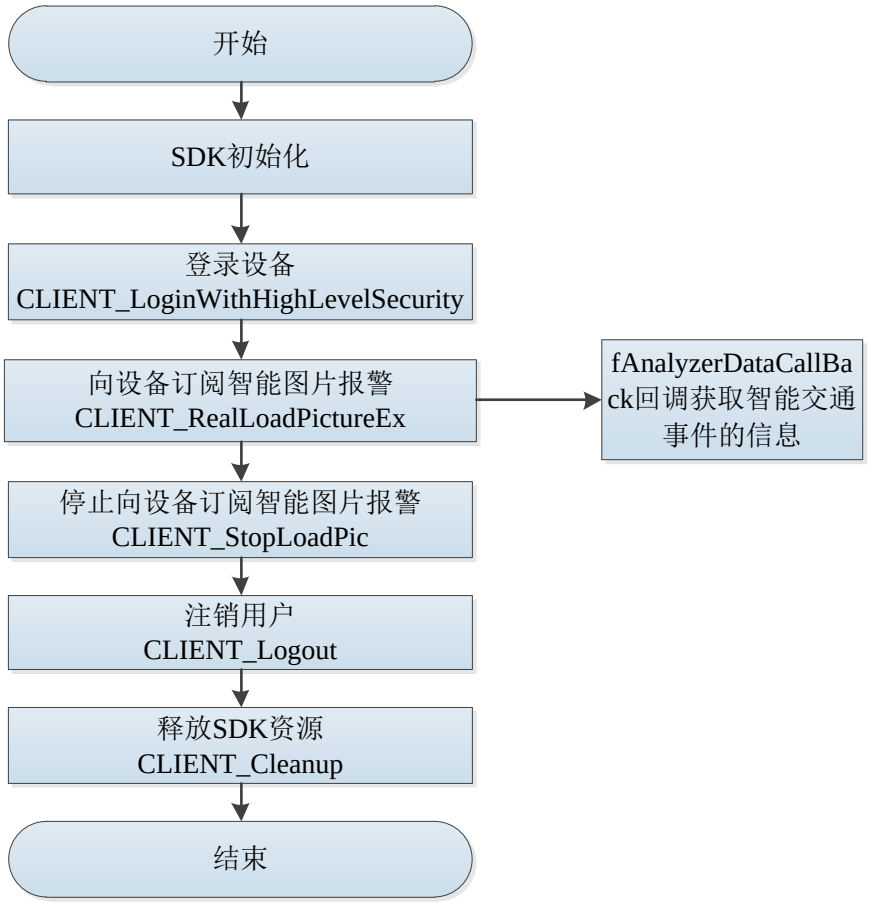
表2-8 智能交通时间上报的接口信息

接口	说明
CLIENT_RealLoadPictureEx	订阅智能交通事件
CLIENT_StopLoadPic	取消订阅智能交通事件

2.2.3.3 流程图

智能交通事件上报流程如图 2-10 所示。

图2-10　智能交通事件上报业务流程



流程说明

步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。

步骤3　调用 CLIENT_ RealLoadPictureEx 函数向设备订阅智能交通事件。

步骤4　订阅成功后设备上报的智能交通事件通过 fAnalyzerDataCallBack 回调函数获取智能交通事件并通知用户。

步骤5　智能交通事件上报功能使用完毕后，调用 CLIENT_StopLoadPic 函数停止订阅智能交通事件。

步骤6　业务使用完后，调用 CLIENT_Logout 函数登出设备。

步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

- 订阅事件类型：如果需要同时上报不同智能事件时，支持订阅所有智能事件（EVENT_IVS_ALL）；也支持订阅单个智能事件。
- 设置是否接收图片：由于某些设备所在网络环境是 3G 或 4G 网络，当 SDK 连接设备时，如不需要接收图片可以把 CLIENT_ RealLoadPictureEx 接口中 bNeedPicFile 参数设置为 False，只接收智能交通事件信息，不带图片。

2.2.3.4 示例代码

```
int main()
{
    ……
    //订阅智能交通事件上报
    LLONG lAnalyerHandle = CLIENT_RealLoadPictureEx(lLoginHandle, 0, (DWORD)EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, NULL, NULL);
    if(NULL == lAnalyerHandle)
    {
        printf("CLIENT_RealLoadPictureEx: failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    Sleep(5000);
    //取消订阅智能交通事件上报
    BOOL bRet = CLIENT_StopLoadPic(lAnalyerHandle);
    if(FALSE == bRet)
    {
        printf("CLIENT_StopLoadPic: failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    return 0;
}
//智能交通事件上报回调。
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    switch(dwAlarmType)
    {
        ……
        case EVENT_IVS_TRAFFIC_RUNREDLIGHT: //闯红灯事件
            {
                DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO* pInfo = (DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO*)pAlarmInfo;
                ……
                break;
            }
        ……
        default:
            break;
```

```
    }
    return 0;
}
```

2.2.4 车流量统计

2.2.4.1 简介

车流量统计即在道路上 ITC 设备统计所有经过的车辆，分析出道路的拥堵情况。

ITC 会把车流量结果发送给用户或发送给 ITSE 再发送给用户。

2.2.4.2 接口总览

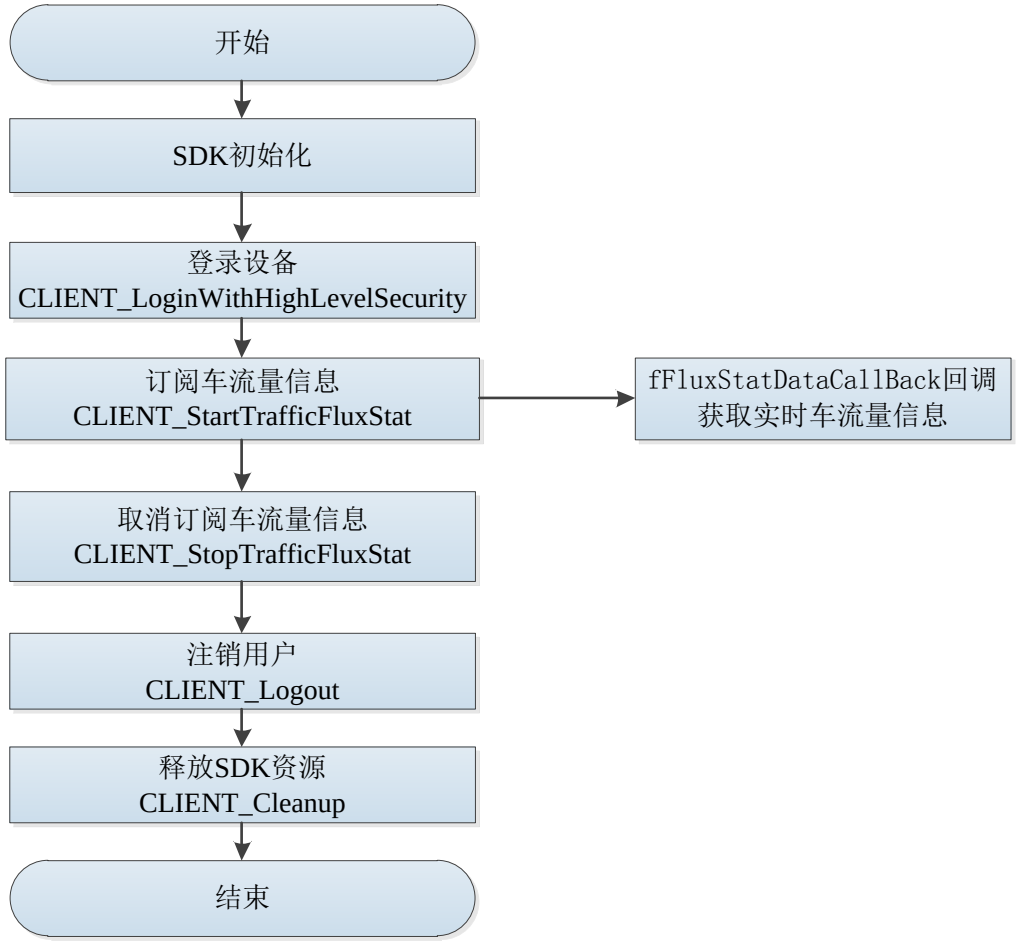
表2-9 车流量统计的接口信息

接口	说明
CLIENT_StartTrafficFluxStat	订阅车流量信息
CLIENT_StopTrafficFluxStat	取消订阅车流量信息

2.2.4.3 流程图

车流量统计流程如图 2-11 所示。

图2-11　车流量统计业务流程



流程说明

步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。
步骤3　调用 CLIENT_StartTrafficFluxStat 函数向设备订阅车流量信息。
步骤4　订阅成功后，ITC 或 ITSE 上报的车流量信息通过 fFluxStatDataCallBack 回调函数获取实时车流量信息并通知用户。
步骤5　车流量信息使用完毕后，调用 CLIENT_StopTrafficFluxStat 函数取消订阅车流量信息。
步骤6　业务使用完后，调用 CLIENT_Logout 函数登出设备。
步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

回调数据类型：参数 pEventInfo 对应的结构体为 DEV_EVENT_TRAFFIC_FLOWSTAT_INFO。

2.2.4.4 示例代码

```
int main()
{
    ……
    NET_IN_TRAFFICFLUXSTAT stuIn = {0};
```

```
    stuIn.dwSize = sizeof(NET_IN_TRAFFICFLUXSTAT);
    stuIn.cbData = FluxStatDataCallBack;
    NET_OUT_TRAFFICFLUXSTAT stuOut = {0};
    stuOut.dwSize = sizeof(NET_OUT_TRAFFICFLUXSTAT);
    //订阅车流量统计信息
    LLONG lFluxStatHandle = CLIENT_StartTrafficFluxStat(lLoginHandle, &stuIn, &stuOut);
    if(NULL == lFluxStatHandle)
    {
            printf("CLIENT_StartTrafficFluxStat: failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    Sleep(5000);
    //取消订阅车流量统计信息
    BOOL bRet = CLIENT_StopTrafficFluxStat(lFluxStatHandle);
    if(FALSE == bRet)
    {
        printf("CLIENT_StopTrafficFluxStat: failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    return 0;
}
//车流量统计信息回调
int CALLBACK FluxStatDataCallBack (LLONG lFluxStatHandle, DWORD dwEventType, void* pEventInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
        DEV_EVENT_TRAFFIC_FLOWSTAT_INFO* pInfo =
    (DEV_EVENT_TRAFFIC_FLOWSTAT_INFO*)pEventInfo;
    ……
    return 0;
}
```

2.3 停车场

2.3.1 道闸控制

2.3.1.1 简介

道闸控制即 IPMECK 设备（一般指的是抓拍相机）控制道路的闸门，可开启道闸和关闭道闸。用户通过 SDK 下发命令给 IPMECK 设备去手动控制道闸，例如：

- 可通过 SDK 下发配置命令给 IPMECK，设置道闸常开常闭，以及对时间段控制。
- 可根据车辆位置事件（主流做法）或交通路口事件来联动开闸。

道闸控制一般应用于大型商用停车场、收费站、小区或园区出入口等场景。

适用设备：IPMECK 设备。

2.3.1.2 接口总览

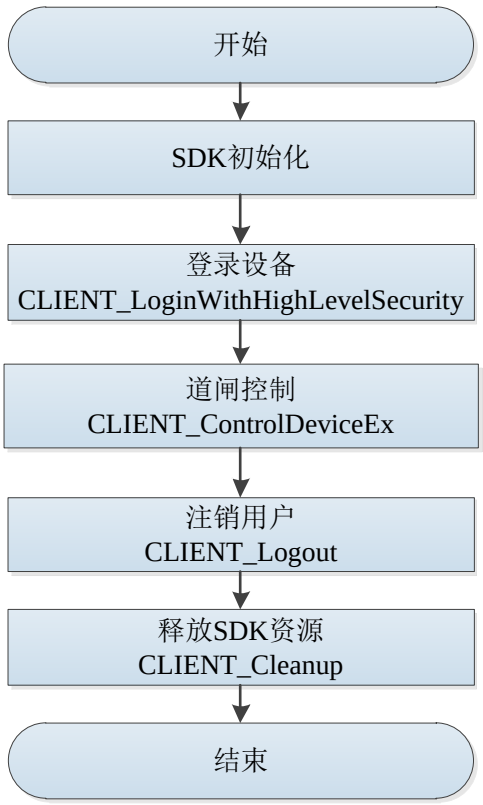
表2-10 道闸控制的接口信息

接口	说明
CLIENT_ControlDeviceEx	控制道闸
CLIENT_GetConfig	获取道闸配置
CLIENT_SetConfig	下发道闸配置
CLIENT_SetDVRMessCallBack	设置报警回调函数
CLIENT_StartListenEx	订阅车辆位置事件
CLIENT_StopListen	取消订阅车辆位置事件
CLIENT_RealLoadPictureEx	订阅交通路口事件
CLIENT_StopLoadPic	取消订阅交通路口事件

2.3.1.3 流程图

2.3.1.3.1 手动控制道闸流程

图2-12 手动控制道闸业务流程



流程说明

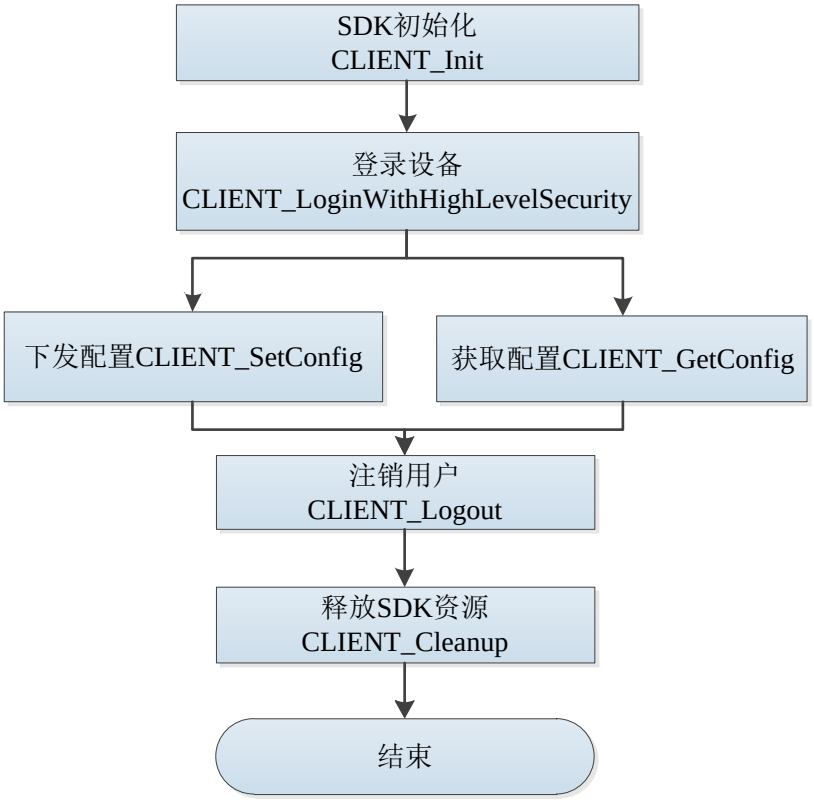
步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。
步骤3　调用 CLIENT_ControlDeviceEx 函数控制道闸开启或关闭。
步骤4　业务结束，调用 CLIENT_Logout 函数登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

无。

2.3.1.3.2 道闸配置流程

图2-13 道闸配置业务流程



流程说明

设置：

步骤1　完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　调用 CLIENT_SetConfig 设置道闸常开使能、常开模式执行时间段。
步骤4　调用 CLIENT_Logout，登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

获取：

步骤1　完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

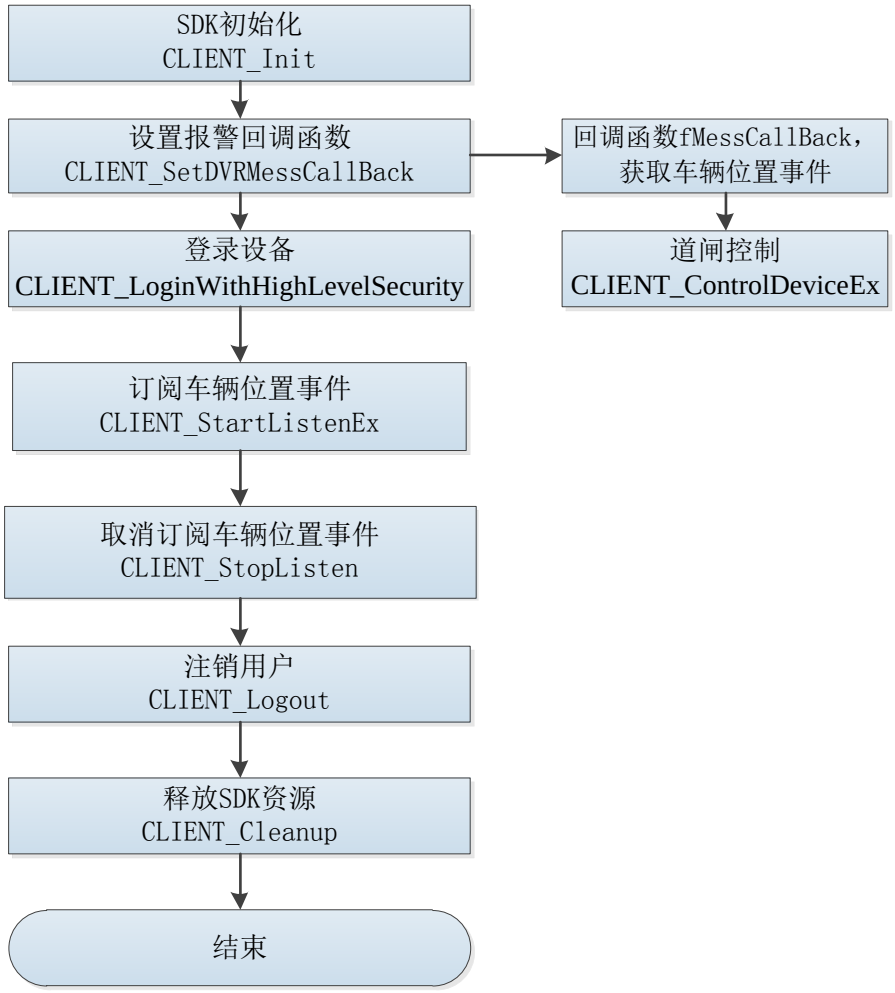
步骤3　调用 CLIENT_GetConfig 获取道闸常开使能、常开模式执行时间段配置。
步骤4　调用 CLIENT_Logout，登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

无。

2.3.1.3.3 车辆位置事件联动道闸控制流程

图2-14 车辆位置事件联动道闸控制业务流程

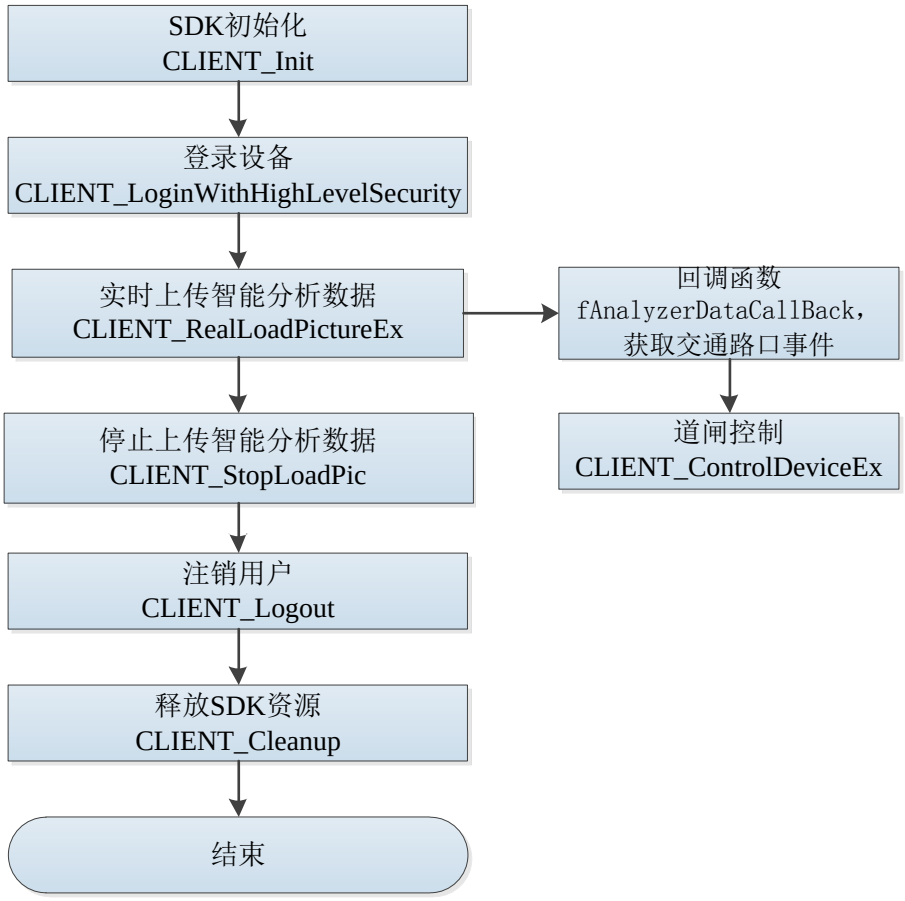


流程说明

步骤1　完成 SDK 初始化流程。
步骤2　调用 CLIENT_SetDVRMessCallBack 设置报警回调函数，当有车辆位置事件过来时调用函数 CLIENT_ControlDeviceEx 开闸。
步骤3　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤4　调用 CLIENT_StartListenEx 订阅车辆位置事件。
步骤5　调用 CLIENT_StopListen 取消订阅车辆位置事件。
步骤6　调用 CLIENT_Logout，登出设备。
步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

2.3.1.3.4 交通路口事件联动道闸控制流程

图2-15 交通路口事件联动道闸控制业务流程



流程说明

步骤1　完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　调用 CLIENT_RealLoadPictureEx 订阅交通路口事件，当有事件触发时 fAnalyzerDataCallBack 调用函数 CLIENT_ControlDeviceEx 开闸。
步骤4　调用 CLIENT_StopLoadPic 取消订阅交通路口事件。
步骤5　调用 CLIENT_Logout，登出设备。
步骤6　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

2.3.1.4 示例代码

2.3.1.4.1 手动控制道闸

```
int main()
{
    ……
    NET_CTRL_OPEN_STROBE stuOpenStrobe = {0};
    stuOpenStrobe.dwSize = sizeof(NET_CTRL_OPEN_STROBE);
    stuOpenStrobe.nChannelId = 0;
```

```
    sprintf(stuOpenStrobe.szPlateNumber,"浙 A54321");
    //开启道闸
    BOOL bRet = CLIENT_ControlDeviceEx(lLoginHandle,DH_CTRL_OPEN_STROBE,&stuOpenStrobe);
    if(FALSE == bRet)
    {
        printf("CLIENT_ControlDeviceEx: Open strobe failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    NET_CTRL_CLOSE_STROBE stuCloseStrobe = {0};
    stuCloseStrobe.dwSize = sizeof(NET_CTRL_CLOSE_STROBE);
    stuCloseStrobe.nChannelId = 0;
    //关闭道闸
    bRet = CLIENT_ControlDeviceEx(lLoginHandle,DH_CTRL_CLOSE_STROBE,&stuCloseStrobe);
    if(FALSE == bRet)
    {
        printf("CLIENT_ControlDeviceEx: Close strobe failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    return 0;
}
```

2.3.1.4.2 道闸配置

```
//获取道闸配置
NET_CFG_TRAFFICSTROBE_INFO m_stuTrafficstrobeInfo = {sizeof(m_ stuTrafficstrobeInfo)};
BOOL bRet = CLIENT_GetConfig(m_LoginID, NET_EM_CFG_TRAFFICSTROBE,m_nChannel,&m_ stuTrafficstrobeInfo,sizeof(m_ stuTrafficstrobeInfo), 5000);
if (！ bRet)
{
    //失败
}
//设置道闸配置
NET_CFG_TRAFFICSTROBE_INFO m_ stuTrafficstrobeInfo = {sizeof(m_ stuTrafficstrobeInfo)};
…….
BOOL bRet = CLIENT_SetConfig(m_LoginID, NET_EM_CFG_TRAFFICSTROBE,m_nChannel,&m_ stuTrafficstrobeInfo,sizeof(m_ stuTrafficstrobeInfo), 5000);
if (！ bRet)
{
```

```
    //失败
}
```

2.3.1.4.3 车辆位置事件联动开闸

```
// 事件回调
int CALLBACK    afMessCallBack(LONG lCommand, LLONG lLinID, char *pBuf, DWORD dwBufLen,
char *pchDVRIP, LONG nDVRPort, LDWORD dwUser)
{
    if(lCommand == DH_ALARM_TRAFFIC_VEHICLE_POSITION) // 车辆位置事件
    {
        ALARM_TRAFFIC_VEHICLE_POSITION* pstAccessInfo =
                                        (ALARM_TRAFFIC_VEHICLE_POSITION*)pBuf;
         //接下来可以通过 pstAccessInfo 的信息 对闸进行控制。
    }
}
// 设置事件回调
CLIENT_SetDVRMessCallBack(afMessCallBack,0);
// 订阅车辆位置事件
CLIENT_StartListenEx(lLoginHandle);
// 停止车辆位置事件订阅
CLIENT_StopListen(lLoginHandle);
```

2.3.1.4.4 交通路口事件联动开闸

```
int main()
{
    ……
    //订阅交通路口事件
    LLONG lAnalyerHandle = CLIENT_RealLoadPictureEx(lLoginHandle, 0,
(DWORD)EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, NULL, NULL);
    if(NULL == lAnalyerHandle)
    {
        printf("CLIENT_RealLoadPictureEx: failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    //取消订阅交通路口事件
    BOOL bRet = CLIENT_StopLoadPic(lAnalyerHandle);
    if(FALSE == bRet)
    {
        printf("CLIENT_StopLoadPic: failed! Error code %x.\n", CLIENT_GetLastError());
```

```
            return -3;
        }
        return 0;
}
//交通路口回调
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
        switch(dwAlarmType)
        {
            case EVENT_IVS_TRAFFICJUNCTION:
                {
                    DEV_EVENT_TRAFFICJUNCTION_INFO*  pInfo  =  (DEV_EVENT_TRAFFICJUNCTION_INFO*)pAlarmInfo;
                    //根据 pInfo 信息对道闸进行控制。
                   break;
                }
            default:
                break;
        }
        return 0;
}
```

2.3.2 黑白名单导入导出

2.3.2.1 简介

黑白名单导入导出一般应用于摄像机快速配置，导入的名单需要配合摄像机内设置才可以使用。

适用设备：IPMECK 设备。

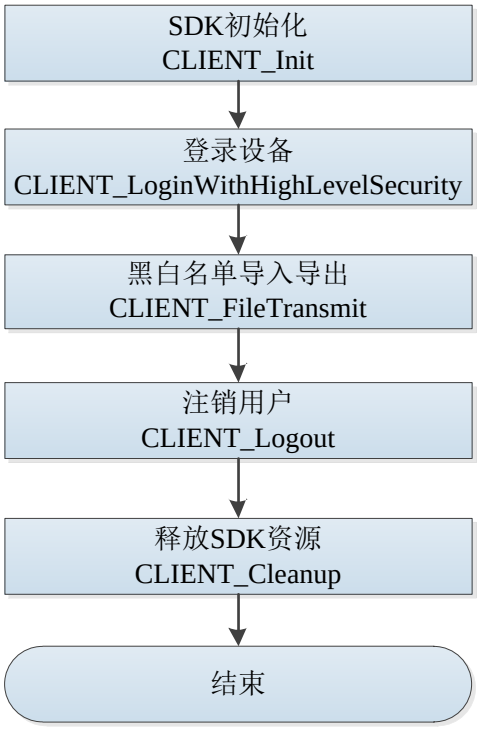
2.3.2.2 接口总览

表2-11 道闸控制的接口信息

接口	说明
CLIENT_FileTransmit	黑白名单导入导出

2.3.2.3 流程图

图2-16 黑白名单导入导出业务流程



流程说明

步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。
步骤3　调用 CLIENT_FileTransmit 函数控制黑白名单导入导出。
步骤4　业务使用完后，调用 CLIENT_Logout 函数登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

注意导入的表头与相机内置模板保持一致，如不一致，则无法在设备中查询成功。

2.3.2.4 示例代码

```
//文件传输进度回调函数
void CALLBACK   bfTransFileCallBack(LLONG lHandle, int nTransType, int nState, int nSendSize, int nTotalSize, LDWORD dwUser)
{
    if (nTransType == DH_DEV_BLACKWHITE_LOAD)
    {
        if (nState == 0)
        {
            //调用  stopLoadFileTransmit  结束导出黑白名单
```

```
        }
    }
    else if(nTransType == DH_DEV_BLACKWHITETRANS_SEND)
    {
        if (nState == 0)
        {
            //调用 stopSendFileTransmit 结束发送黑白名单
        }
    }
    //显示文件传输进度
}
//停止导出黑白名单
Void stopLoadFileTransmit(LLONG lHandle )
{
LLONG nRet =
CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITE_LOAD_STOP,(char*)&lHandle,sizeof(LLONG),NULL,NULL,5000);
}
//停止发送黑白名单
void CBWListDlg::stopSendFileTransmit(LLONG lHandle )
{
    LLONG nRet = CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITETRANS_STOP,(char*)&lHandle,sizeof(LLONG),NULL,NULL,5000);
}

int main()
{
    //黑白名单导出
    DHDEV_LOAD_BLACKWHITE_LIST_INFO stulistinfo;
    CString strPath = "C:\\1\\3.CSV";
    strncpy(stulistinfo.szFile, strPath.GetBuffer(), sizeof(stulistinfo.szFile)-1);
    stulistinfo.byFileType = 1;
    LLONG nRet =
CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITE_LOAD,(char*)&stulistinfo,sizeof(DHDEV_LOAD_BLACKWHITE_LIST_INFO),bfTransFileCallBack,(LDWORD)this,5000);
    if (nRet <= 0)
    {
        //失败
    }
```

```
    //黑白名单发送
    DHDEV_BLACKWHITE_LIST_INFO stulistinfo;
    CString strPath = "C:\\1\\3.CSV";
    strncpy(stulistinfo.szFile, strPath.GetBuffer(), sizeof(stulistinfo.szFile)-1);
    stulistinfo.byFileType = 1;
    stulistinfo.byAction = 0;
    LLONG nHandle = CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITETRANS_START,(char*)&stulistinfo,sizeof(DHDEV_BLACKWHITE_LIST_INFO),bfTransFileCallBack,(LDWORD)this,5000);
    if (nHandle > 0)
    {
        LLONG nRet = CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITETRANS_SEND,(char*)&nHandle,sizeof(LLONG),bfTransFileCallBack,(LDWORD)this,5000);
        if (nRet <= 0)
        {
            //失败
        }
    }
    else
    {
        //失败
    }
    return 0;
}
```

2.3.3 语音对讲

2.3.3.1 简介

语音对讲主要用于实现本地平台与前端设备所处环境间的语音交互，解决本地平台需要与现场环境语音交流的需求。例如：无人值守方案中，向中心平台请求沟通处理开闸异常。

本章主要介绍用户如何使用 SDK 实现与设备的语音对讲。

2.3.3.2 接口总览

表2-12 语音对讲接口信息

接口	说明
CLIENT_StartTalkEx	打开语音对讲扩展接口
CLIENT_StopTalkEx	停止语音对讲扩展接口

接口	说明
CLIENT_RecordStartEx	开始客户端录音扩展接口（只在 Windows 平台下有效）
CLIENT_RecordStopEx	结束客户端录音扩展接口（只在 Windows 平台下有效）
CLIENT_TalkSendData	发送语音数据到设备
CLIENT_AudioDecEx	解码音频数据扩展接口（只在 Windows 平台下有效）
CLIENT_SetDeviceMode	设置设备语音对讲工作模式
CLIENT_SetDVRMessCallBack	设置设备请求对方发起对讲事件回调函数
CLIENT_StartListenEx	订阅设备请求对方发起对讲事件
CLIENT_StopListen	取消订阅设备请求对方发起对讲事件

2.3.3.3 流程说明

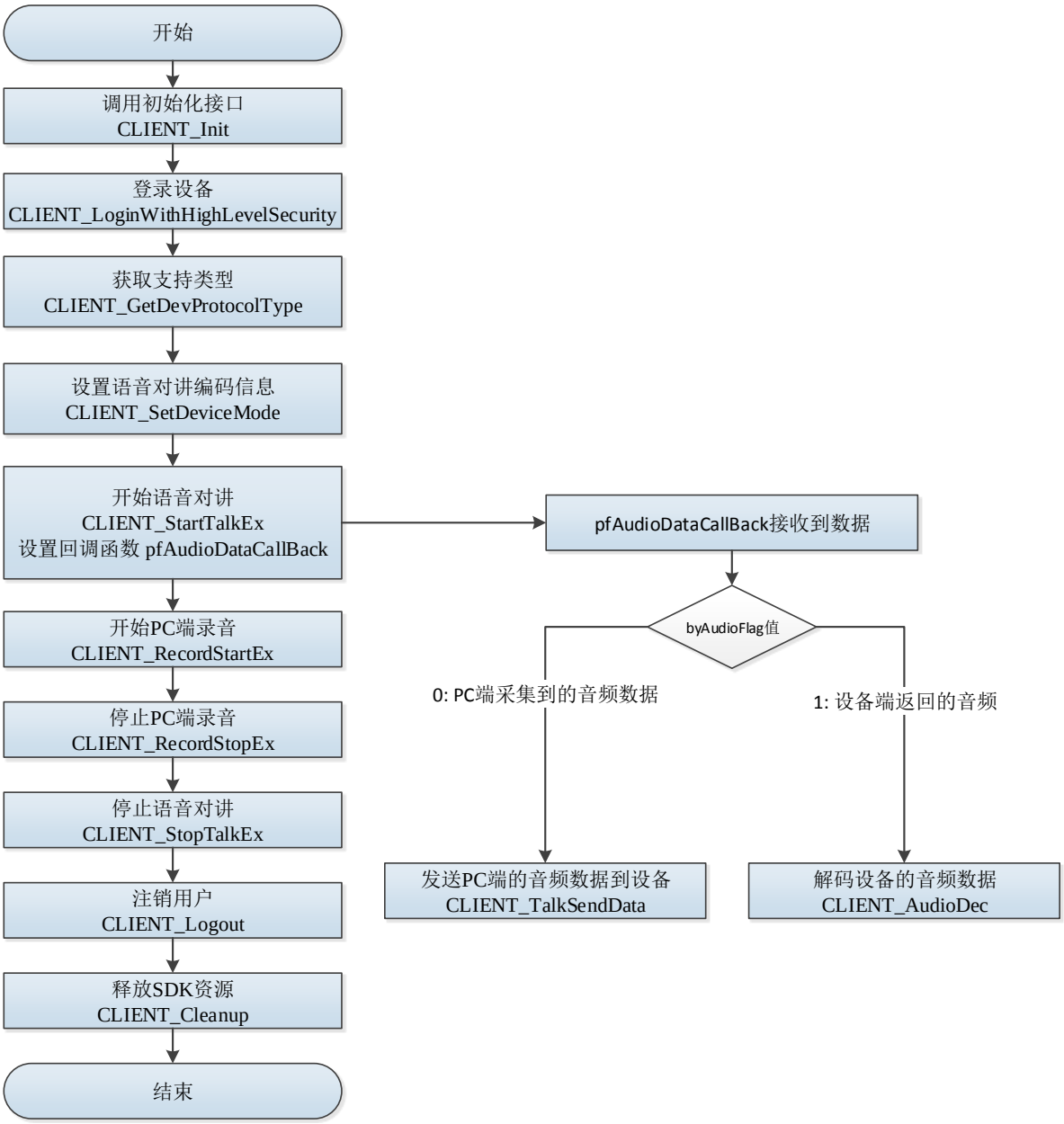
2.3.3.3.1 对讲流程

当 SDK 从本地声卡采集到音频数据或 SDK 接收到前端发送过来的音频数据时，会调用音频数据回调函数。用户可在回调函数中调用 SDK 接口将采集到的本地音频数据发送到前端设备，也可以调用 SDK 接口将接收到的前端设备的音频数据进行解码播放。语音对讲流程如图 2-17 所示。

说明

- 该模式只在 Windows 平台下有效。
- 目前语音对讲分为二代和三代语音对讲，可以使用接口 CLIENT_GetDevProtocolType 获取设备支持的语音对讲方式，二代和三代对讲流程相同，区别在于 CLIENT_SetDeviceMode 设置的对讲参数不同。

图2-17 二代语音对讲流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_GetDevProtocolType 获取支持二代对讲或者三代对讲。

步骤4　调用 CLIENT_SetDeviceMode 设置语音对讲参数。

- 如果支持二代对讲：设置编码模式、客户端模式和喊话模式，参数 emType 设置为 DH_TALK_ENCODE_TYPE、DH_TALK_CLIENT_MODE 和 DH_TALK_SPEAK_PARAM。
- 如果支持三代对讲：设置编码模式、客户端模式和三代语音对讲参数，参数 emType 设置为 DH_TALK_ENCODE_TYPE、DH_TALK_CLIENT_MODE 和 DH_TALK_MODE3。

步骤5　调用 CLIENT_StartTalkEx 设置回调函数并开始语音对讲。在回调函数中，调用 CLIENT_AudioDec，解码设备发送过来的音频数据；调用 CLIENT_TalkSendData，发

送 PC 端的音频数据到设备。

步骤6　调用 CLIENT_RecordStartEx 开始 PC 端录音，该接口调用后，CLIENT_StartTalkEx 设置的语音对讲回调函数中才会收到本地音频数据。

步骤7　对讲功能使用完毕后，调用 CLIENT_RecordStopEx 停止 PC 端录音。

步骤8　调用 CLIENT_StopTalkEx 停止语音对讲。

步骤9　调用 CLIENT_Logout 登出设备。

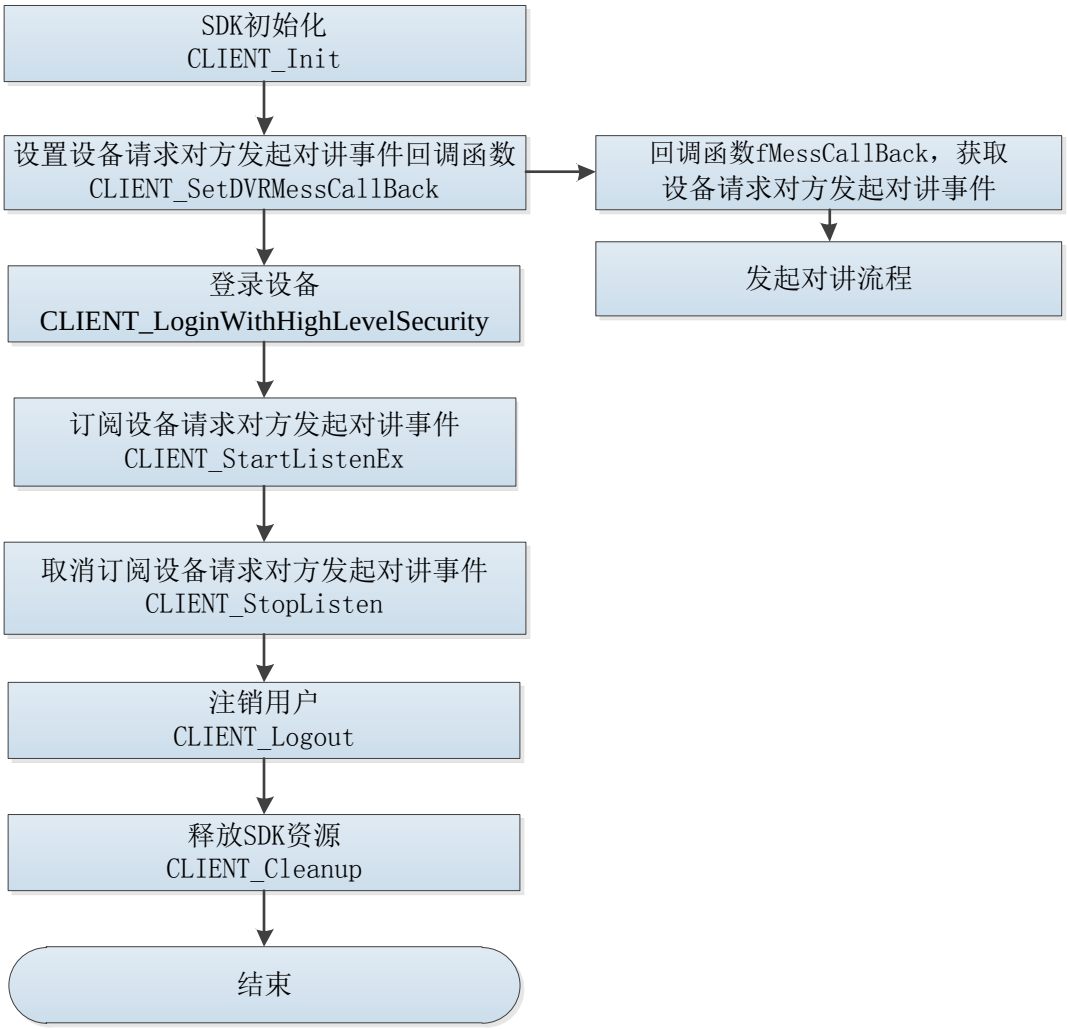
步骤10　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- 语音编码格式：示例采用了常用的 PCM 格式，SDK 支持获取设备支持的语音编码格式，示例源码详见官网发布包。如果默认 PCM 能满足需求，建议不用获取设备支持的语音编码格式。
- 设备端无声音：需要从麦克风等设备采集音频数据，建议检查是否插上麦克风等音频采集设备，同时检查 CLIENT_RecordStartEx 接口是否返回成功。

2.3.3.3.2 设备请求对方发起对讲事件的流程

图2-18 设备请求对方发起对讲事件的流程



流程说明

步骤1　完成 SDK 初始化流程。
步骤2　调用 CLIENT_SetDVRMessCallBack 设置报警回调函数，当有请求对讲事件时调用对讲流程
步骤3　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤4　调用 CLIENT_StartListenEx 订阅请求对讲事件，
步骤5　调用 CLIENT_StopListen 取消订阅请求对讲事件
步骤6　调用 CLIENT_Logout，登出设备
步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

无。

2.3.3.4 示例代码

2.3.3.4.1 对讲示例

```
// 获取设备支持二代还是三代语音对讲。
EM_DEV_PROTOCOL_TYPE emTpye = EM_DEV_PROTOCOL_UNKNOWN;
CLIENT_GetDevProtocolType(g_lLoginHandle, &emTpye);

DHDEV_TALKDECODE_INFO curTalkMode = {0};
curTalkMode.encodeType = DH_TALK_PCM;
curTalkMode.nAudioBit = 16;
curTalkMode.dwSampleRate = 8000;
curTalkMode.nPacketPeriod = 25;
CLIENT_SetDeviceMode(lLoginHandle, DH_TALK_ENCODE_TYPE, &curTalkMode); //设置对讲编码格式
CLIENT_SetDeviceMode(lLoginHandle, DH_TALK_CLIENT_MODE, NULL);// 设置客户端方式语音对讲

// 根据获取出的对讲类型，设置对讲参数
if (emTpye == EM_DEV_PROTOCOL_V3) //三代对讲需要设备此类参数，而二代设备不需要设置此类参数
{
    NET_TALK_EX stuTalk = {sizeof(stuTalk)};
        stuTalk.nAudioPort    = RECEIVER_AUDIO_PORT;  // 用户自定义接收端口
        stuTalk.nChannel = 0;
        stuTalk.nWaitTime     = 5000;
        CLIENT_SetDeviceMode(m_lLoginHandle, DH_TALK_MODE3, &stuTalk)
}
```

```
// 开始语音对讲
lTalkHandle = CLIENT_StartTalkEx(lLoginHandle, AudioDataCallBack, (LDWORD)NULL);
// 开始本地录音
CLIENT_RecordStartEx(lLoginHandle);
// 停止本地录音
CLIENT_RecordStopEx(lLoginHandle)
// 停止语音对讲
CLIENT_StopTalkEx(lTalkHandle);
// 语音对讲回调数据处理
void CALLBACK AudioDataCallBack(LLONG lTalkHandle, char *pDataBuf, DWORD dwBufSize, BYTE byAudioFlag, LDWORD dwUser)
{
if(0 == byAudioFlag)
    {
        // 将收到的本地 PC 端检测到的声卡数据发送给设备端
        CLIENT_TalkSendData(lTalkHandle, pDataBuf, dwBufSize);
    }
else if(1 == byAudioFlag)
    {
        // 将收到的设备端发送过来的语音数据传给 SDK 解码播放
        CLIENT_AudioDec(pDataBuf, dwBufSize);
    }
}
```

2.3.3.4.2 设备请求对方发起对讲事件示例

```
// 设备请求对方发起对讲事件回调
int CALLBACK    afMessCallBack(LONG lCommand, LLONG lLinID, char *pBuf, DWORD dwBufLen, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser)
{
    if(lCommand == DH_ALARM_TALKING_INVITE) // 设备请求对方发起对讲事件
    {
        //调用对讲流程 2.3.3.4.1
    }
}
// 设置设备请求对方发起对讲事件回调
CLIENT_SetDVRMessCallBack(afMessCallBack,0);
// 订阅设备请求对方发起对讲事件
CLIENT_StartListenEx(lLoginHandle);
// 停止设备请求对方发起对讲事件订阅
```

```
CLIENT_StopListen(lLoginHandle);
```

2.3.4 点阵屏字符控制

2.3.4.1 简介

点阵屏控制分 2 个状态，一个是过车状态，一个是常态（无过车的）。

- 过车状态：车辆出入会使得相机抓拍，触发事件，进入过车状态，持续一定时间（时间设备可设置），过车时显示相关车牌、月卡剩余天数及自定义相关数据，并自动播报屏上数据。
- 常态状态：当过车状态结束后即进入常态状态，此时显示车场余位等信息。

可在设备上分别设置过车状态、常态的显示内容。

说明

- 当处于过车状态时，下发常态下显示信息，无法立即在屏显示，会在设备进入常态时自动刷新显示最新内容（如余位等）。
- 同样当处于常态时，下发过车信息无法立即显示，需车辆出入触发事件，进入过车状态，才能显示过车信息。

2.3.4.2 接口总览

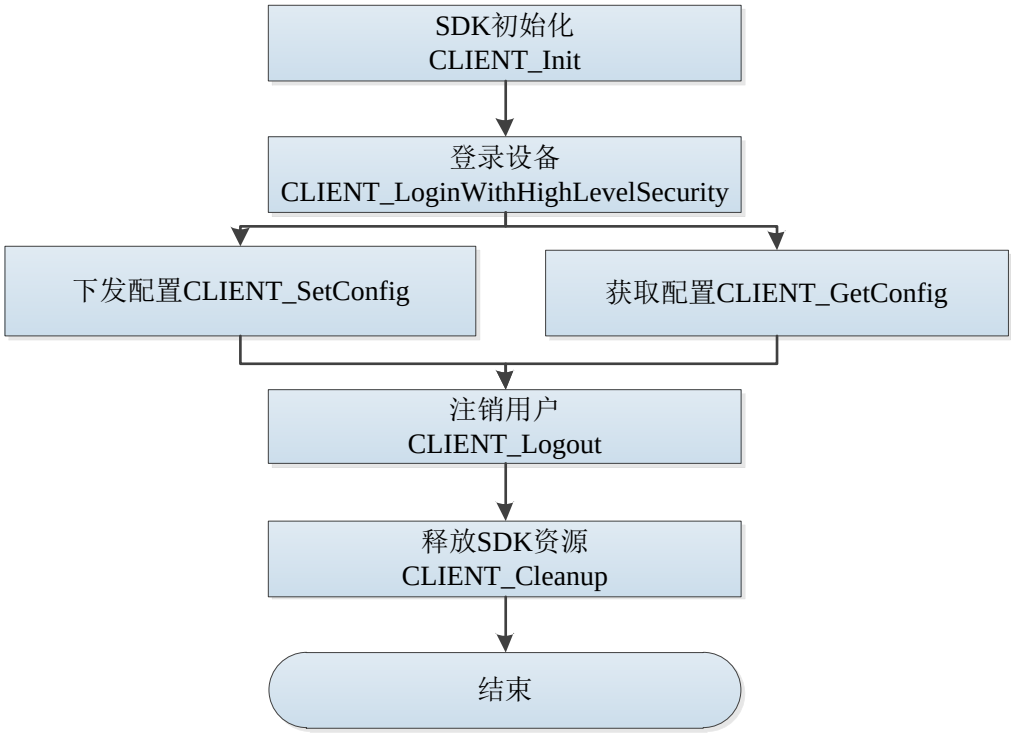
表2-13 点阵屏控制的接口信息

接口	说明
CLIENT_GetConfig	获取点阵屏显示信息配置
CLIENT_SetConfig	设置点阵屏显示信息配置

2.3.4.3 流程图

点阵屏字符控制流程如图 2-19 所示。

图2-19 点阵屏字符控制业务流程



流程说明

设置：

步骤1　完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_SetConfig 设置点阵屏显示信息配置。

步骤4　调用 CLIENT_Logout，登出设备。

步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

获取：

步骤1　完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_GetConfig 获取点阵屏显示信息配置。

步骤4　调用 CLIENT_Logout，登出设备。

步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

无。

2.3.4.4 示例代码

```
//获取点阵屏配置
NET_CFG_TRAFFIC_LATTICE_SCREEN_INFO m_stuTrafficscreenInfo= {sizeof(m_
stuTrafficscreenInfo)};
BOOL bRet = CLIENT_GetConfig(m_LoginID,
NET_EM_CFG_TRAFFIC_LATTICE_SCREEN,m_nChannel,&m_ stuTrafficscreenInfo,sizeof(m_
```

```
stuTrafficscreenInfo), 5000);
if (！ bRet)
{
    //失败
}

//设置点阵屏配置
NET_CFG_TRAFFIC_LATTICE_SCREEN_INFO m_ stuTrafficscreenInfo= {sizeof(m_
stuTrafficscreenInfo)};
……
BOOL bRet = CLIENT_SetConfig(m_LoginID,
NET_EM_CFG_TRAFFIC_LATTICE_SCREEN,m_nChannel,&m_ stuTrafficscreenInfo,sizeof(m_
stuTrafficscreenInfo), 5000);
if (！ bRet)
{
    //失败
}
```

2.3.5 车位指示灯本机配置

2.3.5.1 简介

设置获取车位的灯组监管状态。

2.3.5.2 接口总览

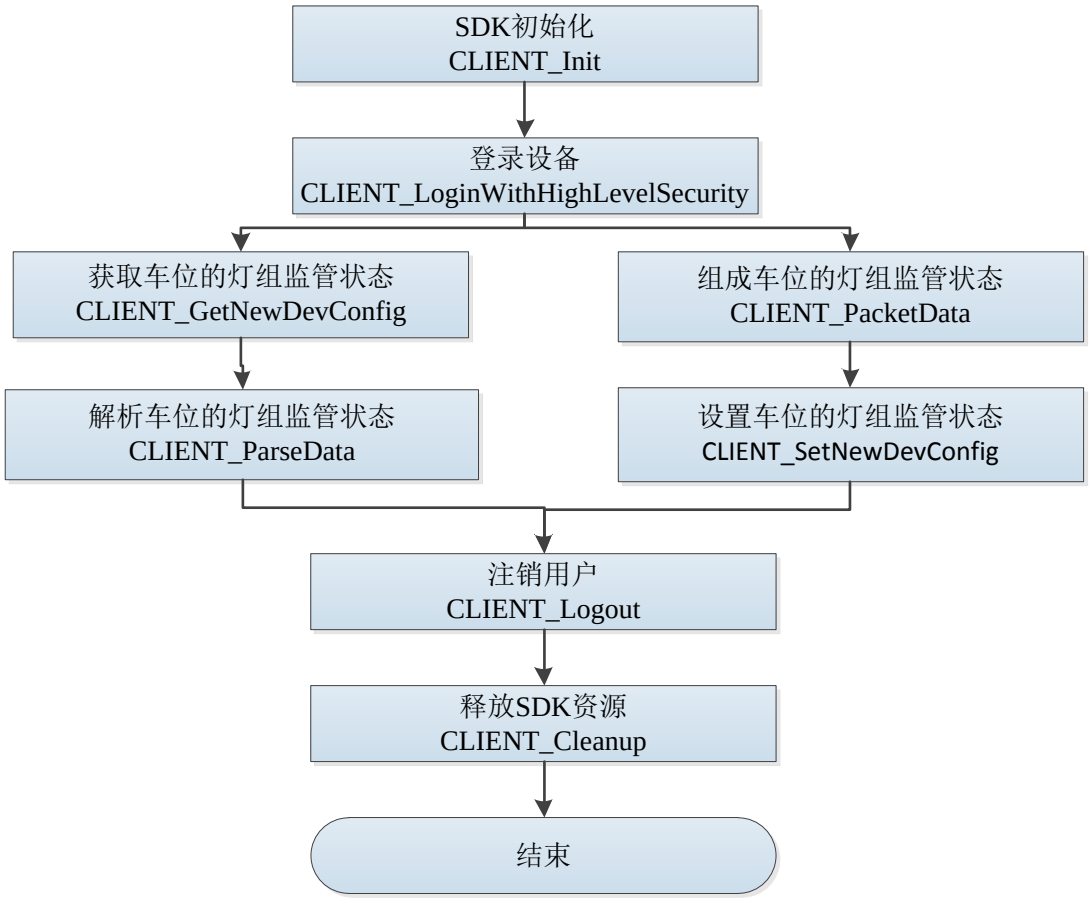
表2-14 车位指示灯本机配置的接口信息

接口	说明
CLIENT_SetNewDevConfig	设置车位的灯组监管状态
CLIENT_GetNewDevConfig	获取车位的灯组监管状态
CLIENT_ParseData	解析车位的灯组监管状态
CLIENT_PacketData	组成车位的灯组监管状态

2.3.5.3 流程图

设置获取车位指示灯本机配置流程如图 2-20 所示。

图2-20 车位指示灯本机配置流程



流程说明

获取：

步骤1　完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_GetNewDevConfig 获取车位的灯组监管状态。

步骤4　调用 CLIENT_ParseData 解析车位的灯组监管状态，得到对应结构体。

步骤5　调用 CLIENT_Logout，登出设备。

步骤6　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

设置

步骤1　完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_PacketData 组成车位的灯组监管状态。

步骤4　调用 CLIENT_SetNewDevConfig 设置车位的灯组监管状态。

步骤5　调用 CLIENT_Logout，登出设备。

步骤6　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

无。

2.3.5.4 示例代码

```
//设置车位指示灯本机配置
CFG_PARKING_SPACE_LIGHT_GROUP_INFO_ALL stuInfo = {0};
stuInfo. nCfgNum= m_ nCfgNum;
for (int i = 0;i<m_ nCfgNum;i++)
{
    stuInfo.stuLightGroupInfo.bEnable = TRUE;
    .....
}
BOOL bRet = CLIENT_PacketData(CFG_CMD_PARKING_SPACE_LIGHT_GROUP,(LPVOID)&stuInfo, sizeof(stuInfo), szJsonBuf, sizeof(szJsonBuf));
if (bRet)
{
    int nerror = 0;
    int nrestart = 0;
    int nChannelID = -1;
    bRet = CLIENT_SetNewDevConfig(m_iLoginID, CFG_CMD_PARKING_SPACE_LIGHT_GROUP, nChannelID, szJsonBuf, 512*40, &nerror, &nrestart, 3000);
}
//获取车位指示灯本机配置
char szJsonBuf[1024 * 40] = {0};
int nerror = 0;
int nChannel = -1;
BOOL ret = CLIENT_GetNewDevConfig(m_iLoginID, CFG_CMD_PARKING_SPACE_LIGHT_GROUP,nChannel,szJsonBuf,1024*40,&nerror,3000);
if (0 != ret)
{
    CFG_PARKING_SPACE_LIGHT_GROUP_INFO_ALL stuInfo = {0};
    DWORD dwRetLen = 0;
    ret = CLIENT_ParseData(CFG_CMD_PARKING_SPACE_LIGHT_GROUP,szJsonBuf,(char*)&stuInfo,sizeof(stuInfo),&dwRetLen);
    if (!ret)
    {
        //获取配置失败
        return ;
    }
}
else
```

```
{
    //获取配置失败
    return ;
}
```

2.3.6 车位状态对应的车位指示灯配置

2.3.6.1 简介

配置车位状态对应的车位指示灯。

- 设置获取车位空闲状态灯色。
- 设置获取车位满状态灯色。
- 设置获取单网口异常灯色。
- 设置获取双网口异常灯色。

2.3.6.2 接口总览

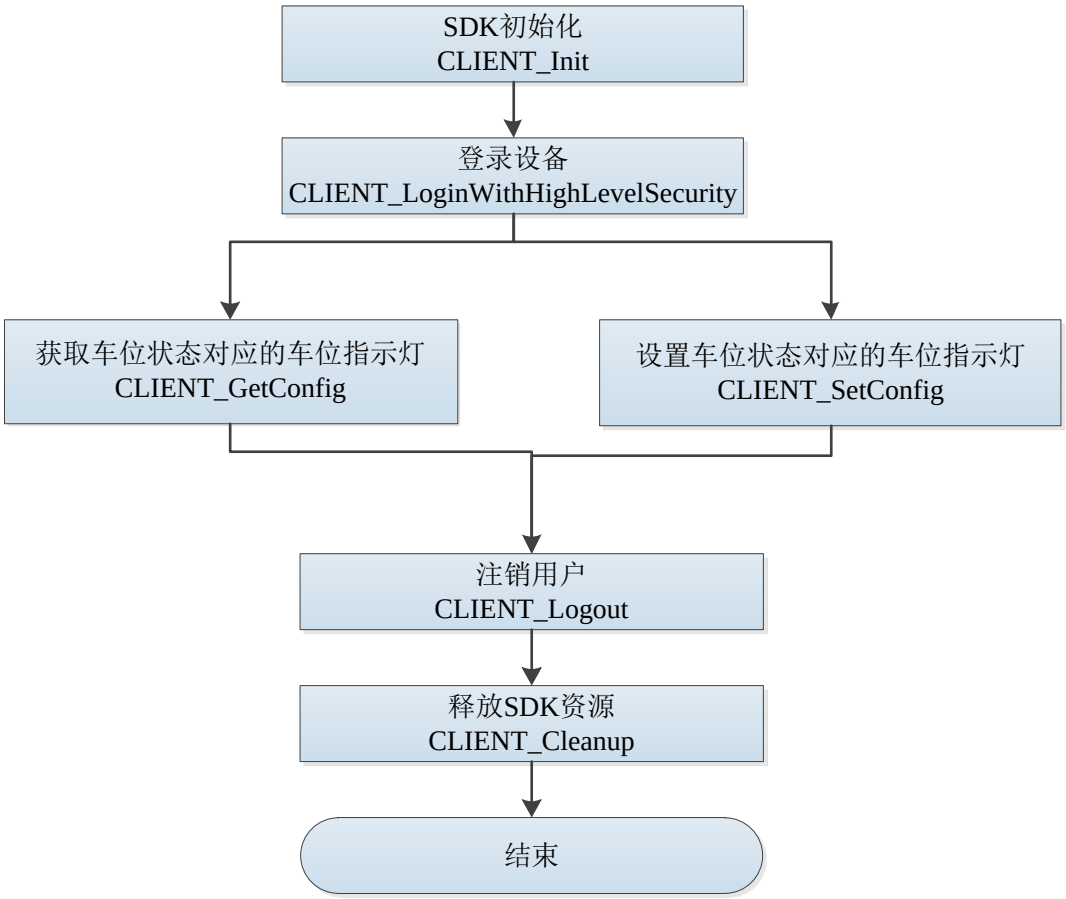
表2-15 车位状态对应的车位指示灯配置的接口信息

接口	说明
CLIENT_SetConfig	设置车位状态对应的车位指示灯
CLIENT_GetConfig	获取车位状态对应的车位指示灯

2.3.6.3 流程图

设置获取车位状态对应的车位指示灯流程如图 2-21 所示。

图2-21 车位状态对应的车位指示灯配置流程



流程说明

获取：

步骤1　完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_GetConfig 获取车位状态对应的车位指示灯。

步骤4　调用 CLIENT_Logout，登出设备。

步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

设置

步骤1　完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。

步骤3　调用 CLIENT_SetConfig 设置车位状态对应的车位指示灯。

步骤4　调用 CLIENT_Logout，登出设备。

步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

无。

2.3.6.4 示例代码

```
//设置车位状态对应的车位指示灯
```

```
NET_PARKINGSPACELIGHT_STATE_INFO stuInfo;
memset(&stuInfo, 0, sizeof(stuInfo));
stuInfo.dwSize = sizeof(stuInfo);
stuInfo.stuSpaceFreeInfo.nRed = 1;          //设置车位空闲状态灯为红色常亮
BOOL bRet = CLIENT_SetConfig(m_lLoginID, NET_EM_CFG_PARKINGSPACELIGHT_STATE, -1, &stuInfo, sizeof(stuInfo));
if (bRet == FALSE)
{
    //设置失败
    return;
}
//获取车位状态对应的车位指示灯配置
NET_PARKINGSPACELIGHT_STATE_INFO stuInfo;
memset(&stuInfo, 0, sizeof(stuInfo));
stuInfo.dwSize = sizeof(stuInfo);
BOOL bRet = CLIENT_GetConfig(m_lLoginID, NET_EM_CFG_PARKINGSPACELIGHT_STATE, -1, &stuInfo, sizeof(stuInfo));
if (bRet == FALSE)
{
    //获取失败
    return;
}
```

第 3 章 接口函数

3.1 通用接口

3.1.1 SDK 初始化

3.1.1.1 SDK 初始化 CLIENT_Init

表3-1 SDK 初始化 CLIENT_Init

选项	说明	
描述	对整个 SDK 进行初始化	
函数	BOOL CLIENT_Init(fDisConnect cbDisConnect, LDWORD dwUser);	
参数	[in]cbDisConnect	断线回调函数
	[in]dwUser	断线回调函数的用户参数
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	● 调用网络 SDK 其他函数的前提 ● 回调函数设置成 NULL 时，设备断线后不会回调给用户	

3.1.1.2 SDK 清理 CLIENT_Cleanup

表3-2 SDK 清理 CLIENT_Cleanup

选项	说明
描述	清理 SDK
函数	void CLIENT_Cleanup()
参数	无
返回值	无
说明	SDK 清理接口，在结束前最后调用

3.1.1.3 设置断线重连回调函数 CLIENT_SetAutoReconnect

表3-3 设置断线重连回调函数 CLIENT_SetAutoReconnect

选项	说明	
描述	设置自动重连回调函数	
函数	void CLIENT_SetAutoReconnect(fHaveReConnect cbAutoConnect, LDWORD dwUser);	
参数	[in]cbAutoConnect	断线重连回调函数

选项	说明	
	[in]dwUser	断线重连回调函数的用户参数
返回值	无	
说明	设置断线重连回调接口。如果回调函数设置为 NULL，则不自动重连	

3.1.1.4 设置网络参数 CLIENT_SetNetworkParam

表3-4 设置网络参数 CLIENT_SetNetworkParam

选项	说明	
描述	设置网络环境相关参数	
函数	void CLIENT_SetNetworkParam(NET_PARAM *pNetParam);	
参数	[in]pNetParam	网络延迟、重连次数、缓存大小等参数
返回值	无	
说明	可根据实际网络环境，调整参数	

3.1.2 设备初始化

3.1.2.1 搜索设备 CLIENT_StartSearchDevicesEx

表3-5 搜索设备 CLIENT_StartSearchDevicesEx

选项	说明	
描述	搜索设备信息	
函数	LLONG CLIENT_StartSearchDevicesEx (NET_IN_STARTSERACH_DEVICE* pInBuf, NET_OUT_STARTSERACH_DEVICE* pOutBuf);	
参数	[in] pInBuf	异步搜索设备入参，具体参考 NET_IN_STARTSERACH_DEVICE 结构体定义
	[out] pOutBuf	异步搜索设备出参，具体参考 NET_OUT_STARTSERACH_DEVICE 结构体定义
返回值	搜索句柄	
说明	不支持多线程调用	

3.1.2.2 设备初始化 CLIENT_InitDevAccount

表3-6 设备初始化 CLIENT_InitDevAccount

选项	说明
描述	初始化设备

选项	说明	
函数	BOOL CLIENT_InitDevAccount(const NET_IN_INIT_DEVICE_ACCOUNT *pInitAccountIn, NET_OUT_INIT_DEVICE_ACCOUNT *pInitAccountOut, DWORD dwWaitTime, char *szLocalIp);	
参数	[in]pInitAccountIn	输入参数，对应 NET_IN_INIT_DEVICE_ACCOUNT 结构体
	[out]pInitAccountOut	输出参数，对应 NET_OUT_INIT_DEVICE_ACCOUNT 结构体
	[in]dwWaitTime	超时时间
	[in]szLocalIp	● 在单网卡的情况下，最后一个参数可不填 ● 在多网卡的情况下，最后一个参数填主机 IP
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.2.3 获取密码重置信息 CLIENT_GetDescriptionForResetPwd

表3-7 获取密码重置信息 CLIENT_GetDescriptionForResetPwd

选项	说明	
描述	获取密码重置信息	
函数	BOOL CLIENT_GetDescriptionForResetPwd(const NET_IN_DESCRIPTION_FOR_RESET_PWD *pDescriptionIn, NET_OUT_DESCRIPTION_FOR_RESET_PWD *pDescriptionOut, DWORD dwWaitTime, char *szLocalIp);	
参数	[in]pDescriptionIn	输入参数，对应 NET_IN_DESCRIPTION_FOR_RESET_PWD 结构体
	[out]pDescriptionOut	输出参数，对应 NET_OUT_DESCRIPTION_FOR_RESET_PWD 结构体
	[in]dwWaitTime	超时时间
	[in]szLocalIp	● 在单网卡的情况下，最后一个参数可不填 ● 在多网卡的情况下，最后一个参数填主机 IP
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.2.4 检验安全码是否有效 CLIENT_CheckAuthCode

表3-8 检验安全码是否有效 CLIENT_CheckAuthCode

选项	说明	
描述	检验安全码否有效	
函数	BOOL CLIENT_CheckAuthCode(const NET_IN_CHECK_AUTHCODE *pCheckAuthCodeIn, NET_OUT_CHECK_AUTHCODE *pCheckAuthCodeOut, DWORD dwWaitTime, char *szLocalIp);	
参数	[in]pCheckAuthCodeIn	输入参数，对应 NET_IN_CHECK_AUTHCODE 结构体
	[out]pCheckAuthCodeOut	输出参数，对应 NET_OUT_CHECK_AUTHCODE 结构体
	[in]dwWaitTime	超时时间
	[in]szLocalIp	● 在单网卡的情况下，最后一个参数可不填 ● 在多网卡的情况下，最后一个参数填主机 IP
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.2.5 重置密码 CLIENT_ResetPwd

表3-9 重置密码 CLIENT_ResetPwd

选项	说明	
描述	重置密码	
函数	BOOL CLIENT_ResetPwd(const NET_IN_RESET_PWD *pResetPwdIn, NET_OUT_RESET_PWD *pResetPwdOut, DWORD dwWaitTime, char *szLocalIp);	
参数	[in]pResetPwdIn	输入参数，对应 NET_IN_RESET_PWD 结构体
	[out]pResetPwdOut	输出参数，对应 NET_OUT_RESET_PWD 结构体
	[in]dwWaitTime	超时时间
	[in]szLocalIp	● 在单网卡的情况下，最后一个参数可不填 ● 在多网卡的情况下，最后一个参数填主机 IP
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.2.6 获取密码规则 CLIENT_GetPwdSpecification

表3-10 获取密码规则 CLIENT_GetPwdSpecification

选项	说明	
描述	获取密码规则	
函数	BOOL CLIENT_GetPwdSpecification(const NET_IN_PWD_SPECI *pPwdSpeciIn, NET_OUT_PWD_SPECI *pPwdSpeciOut, DWORD dwWaitTime, char *szLocalIp);	
参数	[in]pPwdSpeciIn	输入参数，对应 NET_IN_PWD_SPECI 结构体
	[out]pPwdSpeciOut	输出参数，对应 NET_OUT_PWD_SPECI 结构体
	[in]dwWaitTime	超时时间
	[in]szLocalIp	● 在单网卡的情况下，最后一个参数可以不填 ● 在多网卡的情况下，最后一个参数填主机 IP
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.2.7 停止搜索设备 CLIENT_StopSearchDevices

表3-11 停止搜索设备 CLIENT_StopSearchDevices

选项	说明	
描述	停止搜索设备信息	
函数	BOOL CLIENT_StopSearchDevices (LLONG lSearchHandle);	
参数	[in] lSearchHandle	输入参数，搜索句柄
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	不支持多线程调用	

3.1.3 设备登录

3.1.3.1 高安全级别登录 CLIENT_LoginWithHighLevelSecurity

表3-12 高安全级别登录 CLIENT_LoginWithHighLevelSecurity

选项	说明		
描述	用户登录设备		
函数	LLONG CLIENT_LoginWithHighLevelSecurity (NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY* pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY* pstOutParam);		
参数	[in] pstInParam	[in] dwSize	结构体大小

选项	说明		
		[in] szIP	设备 IP
		[in] nPort	设备端口
		[in] szUserName	用户名
		[in] szPassword	密码
		[in] emSpecCap	登录类别
		[in] pCapParam	登录类别参数
	[out] pstOutParam	[in]dwSize	结构体大小
		[out] stuDeviceInfo	设备信息
		[out] nError	失败的错误码
返回值	成功返回设备 ID，失败返回 0。 登录成功之后对设备的操作都可以通过此值（设备 ID）配合 SDK 接口实现。		
说明	高安全级别登录接口。 **说明** CLIENT_LoginEx2 仍然可以使用，但存在安全风险。所以强烈推荐使用最新接口 CLIENT_LoginWithHighLevelSecurity 登录设备。		

参数 error 的错误码及含义说明，请参见表 3-13。

表3-13 参数 error 的错误码及含义

error 的错误码	对应的含义
1	密码不正确
2	用户名不存在
3	登录超时
4	账号已登录
5	账号已被锁定
6	账号被列为黑名单
7	资源不足，设备系统忙
8	子连接失败
9	主连接失败
10	超过最大用户连接数
11	缺少 avnetsdk 或 avnetsdk 的依赖库
12	设备未插入 U 盘或 U 盘信息错误
13	客户端 IP 地址没有登录权限

3.1.3.2 用户登出设备 CLIENT_Logout

表3-14 用户登出设备 CLIENT_Logout

选项	说明	
描述	用户登出设备	
函数	BOOL CLIENT_Logout(LLONG lLoginID);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.4 实时监视

3.1.4.1 打开监视 CLIENT_RealPlayEx

表3-15 打开监视 CLIENT_RealPlayEx

选项	说明	
描述	打开实时监视	
函数	LLONG CLIENT_RealPlayEx(LLONG lLoginID, int nChannelID, HWND hWnd, DH_RealPlayType rType);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]nChannelID	视频通道号，从 0 开始递增的整数
	[in]hWnd	窗口句柄，仅在 Windows 系统下有效
	[in]rType	预览类型
返回值	● 成功返回非 0 ● 失败返回 0	
说明	在 Windows 环境下： ● hWnd 为有效值时，在对应窗口显示画面 ● hWnd 为 NULL 时，表示取流方式，通过设置回调函数来获取视频数据，交由用户处理	

预览类型及含义请参见表 3-16。

表3-16 预览类型说明

预览类型	含义
DH_RType_Realplay	实时预览
DH_RType_Multiplay	多画面预览
DH_RType_Realplay_0	实时监视-主码流，等同于 DH_RType_Realplay
DH_RType_Realplay_1	实时监视-从码流 1
DH_RType_Realplay_2	实时监视-从码流 2
DH_RType_Realplay_3	实时监视-从码流 3
DH_RType_Multiplay_1	多画面预览－1 画面
DH_RType_Multiplay_4	多画面预览－4 画面
DH_RType_Multiplay_8	多画面预览－8 画面
DH_RType_Multiplay_9	多画面预览－9 画面
DH_RType_Multiplay_16	多画面预览－16 画面
DH_RType_Multiplay_6	多画面预览－6 画面
DH_RType_Multiplay_12	多画面预览－12 画面
DH_RType_Multiplay_25	多画面预览－25 画面
DH_RType_Multiplay_36	多画面预览－36 画面

3.1.4.2 关闭监视 CLIENT_StopRealPlayEx

表3-17 关闭监视 CLIENT_StopRealPlayEx

选项	说明	
描述	关闭实时监视	
函数	BOOL CLIENT_StopRealPlayEx(LLONG lRealHandle);	
参数	[in]lRealHandle	CLIENT_RealPlayEx 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.4.3 保存监视数据 CLIENT_SaveRealData

表3-18 保存监视数据 CLIENT_SaveRealData

选项	说明	
描述	保存实时监视数据为文件	
函数	BOOL CLIENT_SaveRealData(LLONG lRealHandle, const char *pchFileName);	
参数	[in] lRealHandle	CLIENT_RealPlayEx 的返回值
	[in] pchFileName	需要保存的文件路径
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.4.4 停止保存监视数据 CLIENT_StopSaveRealData

表3-19 停止保存监视数据 CLIENT_StopSaveRealData

选项	说明	
描述	停止保存实时监视数据为文件	
函数	BOOL CLIENT_StopSaveRealData(LLONG lRealHandle);	
参数	[in] lRealHandle	CLIENT_RealPlayEx 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.1.4.5 设置监视数据回调 CLIENT_SetRealDataCallBackEx2

表3-20 设置监视数据回调 CLIENT_SetRealDataCallBackEx2

选项	说明
描述	设置实时监视数据回调

选项	说明	
函数	BOOL CLIENT_SetRealDataCallBackEx2(LLONG lRealHandle, fRealDataCallBackEx2 cbRealData, LDWORD dwUser, DWORD dwFlag);	
参数	[in] lRealHandle	CLIENT_RealPlayEx 的返回值
	[in] cbRealData	监视数据流回调函数
	[in] dwUser	监视数据流回调函数的参数
	[in] dwFlag	回调中监视数据的类型，EM_REALDATA_FLAG 类型，支持或运算
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

表3-21 dwFlag 类型及含义

dwFlag	**含义**
REALDATA_FLAG_RAW_DATA	原始数据标志
REALDATA_FLAG_DATA_WITH_FRAME_INFO	带有帧信息的数据标志
REALDATA_FLAG_YUV_DATA	YUV 数据标志
REALDATA_FLAG_PCM_AUDIO_DATA	PCM 音频数据标志

3.2 卡口接口

3.2.1 下载智能图片

3.2.1.1 按查询条件查询媒体文件 CLIENT_FindFileEx

表3-22 按查询条件查询媒体文件 CLIENT_FindFileEx

选项	说明	
描述	按查询条件查询媒体文件	
函数	LLONG CLIENT_FindFileEx(LLONG lLoginID, EM_FILE_QUERY_TYPE emType, void* pQueryCondition, void* reserved, int waittime);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] emType	查询媒体文件信息类型，请参见表 3-23
	[in] pQueryCondition	查询条件
	[in]reserved	保留参数，无效
	[in] waittime	超时时间

选项	说明
返回值	● 成功返回非 0 ● 失败返回 0
说明	智能图片信息查询时，emType 使用 DH_FILE_QUERY_TRAFFICCAR_EX，pQueryCondition 对应结构体 MEDIA_QUERY_TRAFFICCAR_PARAM_EX

表3-23 查询媒体文件信息类型

emType 枚举定义	含义	pQueryCondition 对应结构体
DH_FILE_QUERY_TRAFFICCAR	交通车辆信息	MEDIA_QUERY_TRAFFICCAR_PARAM
DH_FILE_QUERY_FACE	人脸信息	MEDIAFILE_FACERECOGNITION_PARAM
DH_FILE_QUERY_FILE	文件信息	NET_IN_MEDIA_QUERY_FILE
DH_FILE_QUERY_TRAFFICCAR_EX	交通车辆信息（扩展）	MEDIA_QUERY_TRAFFICCAR_PARAM_EX
DH_FILE_QUERY_FACE_DETECTION	人脸检测信息	MEDIAFILE_FACE_DETECTION_PARAM

3.2.1.2 获取查询到的文件总数 CLIENT_GetTotalFileCount

表3-24 获取查询到的文件总数 CLIENT_GetTotalFileCount

选项	说明	
描述	获取查询到的文件总数	
函数	BOOL CLIENT_GetTotalFileCount(LLONG lFindHandle, int* pTotalCount, void* reserved, int waittime);	
参数	[in] lFindHandle	CLIENT_FindFileEx 返回值
	[out] pTotalCount	查询到信息的总数
	[in]reserved	保留参数，无效
	[in] waittime	超时时间
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.2.1.3 查询媒体文件 CLIENT_FindNextFileEx

表3-25 查询媒体文件 CLIENT_FindNextFileEx

选项	说明
描述	查询媒体文件

选项	说明	
函数	int CLIENT_FindNextFileEx(LLONG lFindHandle, int nFilecount, void* pMediaFileInfo, int maxlen, void* reserved, int waittime);	
参数	[in] lFindHandle	CLIENT_FindFileEx 返回值
	[in] nFilecount	查询的条数
	[out] pMediaFileInfo	媒体文件信息的输出缓冲
	[in] maxlen	最大缓冲区的值
	[in]reserved	保留参数，无效
	[in] waittime	超时时间
返回值	● 返回查询到的媒体文件的条数 ● 当返回值小于查询条数时，查询完毕	
说明	无	

3.2.1.4 关闭查询媒体文件 CLIENT_FindCloseEx

表3-26 关闭查询媒体文件 CLIENT_FindCloseEx

选项	说明	
描述	关闭查询媒体文件	
函数	BOOL CLIENT_FindCloseEx(LLONG lFindHandle);	
参数	[in] lFindHandle	CLIENT_FindFileEx 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.2.1.5 下载媒体文件 CLIENT_DownloadMediaFile

表3-27 下载媒体文件 CLIENT_DownloadMediaFile

选项	说明
描述	下载媒体文件
函数	LLONG CLIENT_DownloadMediaFile(LLONG lLoginID, EM_FILE_QUERY_TYPE emType, void* lpMediaFileInfo, char* sSavedFileName, fDownLoadPosCallBack cbDownLoadPos, LDWORD dwUserData, void* reserved);

选项	说明	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] emType	下载媒体文件信息类型，请参见表 3-23
	[in] lpMediaFileInfo	媒体文件信息
	[in] sSavedFileName	保存文件的路径
	[in] cbDownLoadPos	下载媒体文件进度回调函数 fDownLoadPosCallBack
	[in] dwUserData	回调接口对应的用户数组
	[in]reserved	保留参数，无效
返回值	● 成功返回非 0 ● 失败返回 0	
说明	下载交通车辆图片时，emType 只支持 DH_FILE_QUERY_TRAFFICCAR 类型	

3.2.1.6 停止下载媒体文件 CLIENT_StopDownloadMediaFile

表3-28 停止下载媒体文件 CLIENT_StopDownloadMediaFile

选项	说明	
描述	停止下载媒体文件	
函数	BOOL CLIENT_StopDownloadMediaFile(LLONG lFileHandle);	
参数	[in] lFindHandle	CLIENT_DownloadMediaFile 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.2.2 智能交通手动抓图

3.2.2.1 订阅智能事件 CLIENT_RealLoadPictureEx

表3-29 订阅智能事件 CLIENT_RealLoadPictureEx

选项	说明	
描述	订阅智能事件	
函数	LLONG CLIENT_RealLoadPictureEx(LLONG lLoginID, int nChannelID, DWORD dwAlarmType, BOOL bNeedPicFile, fAnalyzerDataCallBack cbAnalyzerData, LDWORD dwUser, void* Reserved);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nChannelID	设备通道号
	[in] dwAlarmType	智能交通事件类型，请参见表 3-30 和表 3-34
	[in] bNeedPicFile	是否需要图片

选项	说明	
	[in] cbAnalyzerData	智能事件信息回调 fAnalyzerDataCallBack
	[in] dwUser	回调函数对应的用户数据
	[in] Reserved	保留参数，无效
返回值	● 成功返回非 0 ● 失败返回 0	
说明	智能交通手动抓图需要提前调用该接口，用来接收抓取的图片 智能交通事件上报需要提前调用该接口，用来接收智能交通事件信息及图片	

表3-30 智能交通手动抓图类型

dwAlarmType 宏定义	宏定义值	含义	回调 pAlarmInfo 对应结构体
EVENT_IVS_TRAFFIC_MANUALSNAP	0x00000118	智能抓图事件	DEV_EVENT_TRAFFIC_MANUALSNAP_INFO

3.2.2.2 智能交通手动抓图 CLIENT_ControlDeviceEx

表3-31 智能交通手动抓图 CLIENT_ControlDeviceEx

选项	说明	
描述	设备控制	
函数	BOOL CLIENT_ControlDeviceEx(LLONG lLoginID, CtrlType emType, void* pInBuf, void* pOutBuf, int nWaitTime);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] emType	控制类型，请参见表 3-30 和表 3-32
	[in] pInBuf	控制输入缓存，请参见表 3-30 和表 3-32
	[in] pOutBuf	控制输出缓存
	[in] nWaitTime	超时时间
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	手动触发设备抓图，通过订阅接口的回调函数来收取图片	

表3-32 控制类型

emType 枚举定义	含义	pInBuf 对应结构体
DH_MANUAL_SNAP	智能交通手动抓图	MANUAL_SNAP_PARAMETER

3.2.2.3 取消订阅智能事件 CLIENT_StopLoadPic

表3-33 取消订阅智能事件 CLIENT_StopLoadPic

选项	说明
描述	取消订阅智能事件
函数	BOOL CLIENT_StopLoadPic(LLONG lAnalyzerHandle);

选项	说明	
参数	[in] lAnalyzerHandle	CLIENT_RealLoadPictureEx 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	调用该接口后，继续触发手动抓图也不会收到图片	

3.2.3 智能交通事件上报

3.2.3.1 订阅智能交通事件 CLIENT_RealLoadPictureEx

接口函数请参见“3.2.2.1 订阅智能事件 CLIENT_RealLoadPictureEx”。

智能交通事件类型请参见表 3-34。

表3-34 智能交通事件类型

dwAlarmType 宏定义	宏定义值	含义	回调 pAlarmInfo 对应结构体
EVENT_IVS_ALL	0x00000001	所有事件	无
EVENT_IVS_TRAFFICCONTROL	0x00000015	交通管制事件	DEV_EVENT_TRAFFICCONTROL_INFO
EVENT_IVS_TRAFFICACCIDENT	0x00000016	交通事故事件	DEV_EVENT_TRAFFICACCIDENT_INFO
EVENT_IVS_TRAFFICJUNCTION	0x00000017	交通路口事件	DEV_EVENT_TRAFFICJUNCTION_INFO
EVENT_IVS_TRAFFICGATE	0x00000018	交通卡口事件	DEV_EVENT_TRAFFICGATE_INFO
EVENT_IVS_TRAFFIC_RUNREDLIGHT	0x00000100	闯红灯事件	DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO
EVENT_IVS_TRAFFIC_OVERLINE	0x00000101	压线事件	DEV_EVENT_TRAFFIC_OVERLINE_INFO
EVENT_IVS_TRAFFIC_RETROGRADE	0x00000102	逆行事件	DEV_EVENT_TRAFFIC_RETROGRADE_INFO
EVENT_IVS_TRAFFIC_TURNLEFT	0x00000103	左转违章事件	DEV_EVENT_TRAFFIC_TURNLEFT_INFO
EVENT_IVS_TRAFFIC_TURNRIGHT	0x00000104	右转违章事件	DEV_EVENT_TRAFFIC_TURNRIGHT_INFO
EVENT_IVS_TRAFFIC_UTURN	0x00000105	违章调头事件	DEV_EVENT_TRAFFIC_UTURN_INFO
EVENT_IVS_TRAFFIC_OVERSPEED	0x00000106	超速事件	DEV_EVENT_TRAFFIC_OVERSPEED_INFO
EVENT_IVS_TRAFFIC_UNDERSPEED	0x00000107	低速事件	DEV_EVENT_TRAFFIC_UNDERSPEED_INFO
EVENT_IVS_TRAFFIC_PARKING	0x00000108	违章停车事件	DEV_EVENT_TRAFFIC_PARKING_INFO
EVENT_IVS_TRAFFIC_WRONGROUTE	0x00000109	不按车道行驶事件	DEV_EVENT_TRAFFIC_WRONGROUTE_INFO
EVENT_IVS_TRAFFIC_CROSSLANE	0x0000010A	变道违章事件	DEV_EVENT_TRAFFIC_CROSSLANE_INFO

dwAlarmType 宏定义	宏定义值	含义	回调 pAlarmInfo 对应结构体
EVENT_IVS_TRAFFIC_OVERYELLOWLINE	0x0000010B	压黄线事件	DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO
EVENT_IVS_TRAFFIC_DRIVINGONSHOULDER	0x0000010C	路肩行驶事件	DEV_EVENT_TRAFFIC_DRIVINGONSHOULDER_INFO
EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE	0x0000010E	黄牌车占道事件	DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO
EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY	0x0000010F	斑马线行人优先事件	DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO
EVENT_IVS_TRAFFIC_PARKINGONYELLOWBOX	0x0000012A	黄网络线抓拍事件	DEV_EVENT_TRAFFIC_PARKINGONYELLOWBOX_INFO
EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING	0x0000012B	车位有车事件	DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO
EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING	0x0000012C	车位无车事件	DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO
EVENT_IVS_TRAFFIC_PEDESTRAIN	0x0000012D	行人事件	DEV_EVENT_TRAFFIC_PEDESTRAIN_INFO
EVENT_IVS_TRAFFIC_THROW	0x0000012E	抛物事件	DEV_EVENT_TRAFFIC_THROW_INFO
EVENT_IVS_TRAFFIC_IDLE	0x0000012F	空闲事件	DEV_EVENT_TRAFFIC_IDLE_INFO
EVENT_IVS_TRAFFIC_RESTRICTED_PLATE	0X00000136	受限车牌事件	DEV_EVENT_TRAFFIC_RESTRICTED_PLATE
EVENT_IVS_TRAFFIC_OVERSTOPLINE	0X00000137	压停止线事件	DEV_EVENT_TRAFFIC_OVERSTOPLINE
EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT	0x00000138	未系安全带事件	DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT
EVENT_IVS_TRAFFIC_DRIVER_SMOKING	0x00000139	驾驶员抽烟事件	DEV_EVENT_TRAFFIC_DRIVER_SMOKING
EVENT_IVS_TRAFFIC_DRIVER_CALLING	0x0000013A	驾驶员打电话事件	DEV_EVENT_TRAFFIC_DRIVER_CALLING
EVENT_IVS_TRAFFIC_PEDESTRAINRUNREDLIGHT	0x0000013B	行人闯红灯事件	DEV_EVENT_TRAFFIC_PEDESTRAINRUNREDLIGHT_INFO
EVENT_IVS_TRAFFIC_PASSNOTINORDER	0x0000013C	未按规定依次通行事件	DEV_EVENT_TRAFFIC_PASSNOTINORDER_INFO

3.2.3.2 取消订阅智能交通事件 CLIENT_StopLoadPic

接口函数请参见“3.2.2.3 取消订阅智能事件 CLIENT_StopLoadPic”。

3.2.4 车流量统计

3.2.4.1 订阅交通车流量统计 CLIENT_StartTrafficFluxStat

表3-35 订阅交通车流量统计 CLIENT_StartTrafficFluxStat

选项	说明	
描述	订阅交通车流量统计	
函数	LLONG CLIENT_StartTrafficFluxStat(LLONG lLoginID, NET_IN_TRAFFICFLUXSTAT* pstInParam, NET_OUT_TRAFFICFLUXSTAT* pstOutParam);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] pstInParam	输入参数，车辆流量统计回调 fFluxStatDataCallBack
	[out] pstOutParam	输出参数
返回值	● 成功返回非 0 ● 失败返回 0	
说明	无	

3.2.4.2 取消订阅交通车流量统计 CLIENT_StopTrafficFluxStat

表3-36 取消订阅交通车流量统计 CLIENT_StopTrafficFluxStat

选项	说明	
描述	取消订阅交通车流量统计	
函数	BOOL CLIENT_StopTrafficFluxStat(LLONG lFluxStatHandle);	
参数	[in] lFluxStatHandle	CLIENT_StartTrafficFluxStat 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.3 停车场接口

3.3.1 道闸控制

3.3.1.1 道闸控制 CLIENT_ControlDeviceEx

接口函数请参见“3.2.2.2 智能交通手动抓图 CLIENT_ControlDeviceEx”。

控制类型请参见表 3-37。

表3-37 控制类型

emType 枚举定义	含义	pInBuf 对应结构体
DH_CTRL_OPEN_STROBE	开启道闸	NET_CTRL_OPEN_STROBE

emType 枚举定义	含义	pInBuf 对应结构体
DH_CTRL_CLOSE_STROBE	关闭道闸	NET_CTRL_CLOSE_STROBE

3.3.1.2 设置道闸配置 CLIENT_SetConfig

表3-38 设置道闸配置 CLIENT_SetConfig

选项	说明	
描述	设置道闸配置	
函数	BOOL CLIENT_SetConfig (LLONG lLoginID NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szInBuffer DWORD dwInBufferSize int waittime=3000 int * restart=NULL void * reserve=NULL);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in] emCfgOpType	设置配置的类型 道闸配置：NET_EM_CFG_TRAFFICSTROBE
	[in] nChannelID	通道号
	[in] szInBuffer	配置的缓存地址
	[in] dwInBufferSize	缓存地址大小
	[in] waittime	超时时间
	[in] restart	是否需要重启
	[in] reserve	保留参数
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.1.3 获取道闸配置 CLIENT_GetConfig

图3-1 获取道闸配置 CLIENT_GetConfig

选项	说明	
描述	获取道闸配置	
函数	BOOL CLIENT_GetConfig (LLONG lLoginID NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szOutBuffer DWORD dwOutBufferSize int waittime=3000 void * reserve=NULL);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值

选项	说明	
	[in] emCfgOpType	设置配置的类型 道闸配置：NET_EM_CFG_TRAFFICSTROBE
	[in] nChannelID	通道号
	[in] szOutBuffer	获取配置的缓存地址
	[in] dwOutBufferSize	缓存地址大小
	[in] waittime	超时时间
	[in] reserve	实际获取到配置大小
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.1.4 设置车辆位置信息回调函数接口 CLIENT_SetDVRMessCallBack

表3-39 设置车辆位置信息回调函数接口 CLIENT_SetDVRMessCallBack

选项	说明	
描述	设置车辆位置信息回调函数接口	
函数	void CLIENT_SetDVRMessCallBack(fMessCallBack cbMessage, LDWORD dwUser);	
参数	[in] cbMessage	报警回调函数
	[in] dwUser	用户数据
返回值	无	
说明	CLIENT_SetDVRMessCallBack 接口需在报警订阅之前调用，设置的回调函数不能接收包含图片的事件	

3.3.1.5 订阅车辆位置信息接口 CLIENT_StartListenEx

表3-40 订阅车辆位置信息接口 CLIENT_StartListenEx

选项	说明	
描述	订阅车辆位置信息接口	
函数	BOOL CLIENT_StartListenEx(LLONG lLoginID);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	成功返回 TRUE，失败返回 FALSE	
说明	所有设备的报警事件通过 CLIENT_SetDVRMessCallBack 接口设置的回调函数反馈给用户	

3.3.1.6 停止订阅车辆位置信息接口 CLIENT_StopListen

表3-41 停止订阅车辆位置信息接口 CLIENT_StopListen

选项	说明
描述	停止订阅车辆位置信息接口
函数	BOOL CLIENT_StopListen(

选项	说明	
	LLONG lLoginID);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.1.7 订阅交通路口事件 CLIENT_RealLoadPictureEx

接口函数请参见“3.2.2.1 订阅智能事件 CLIENT_RealLoadPictureEx”。

3.3.1.8 取消订阅交通路口事件 CLIENT_StopLoadPic

接口函数请参见“3.2.2.3 取消订阅智能事件 CLIENT_StopLoadPic”。

3.3.2 黑白名单导入导出

黑白名单导入导出 CLIENT_FileTransmit。

选项	说明	
描述	文件传输接口	
函数	LLONG CLIENT_FileTransmit (LLONG lLoginID, Int nTransType char* szInBuf int nInBufLen fTransFileCallBack cbTransFile LDWORD dwUserData Int waittime);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nTransType	文件控制类型，参见表 3-42
	[in] szInBuf	输入数据，参见表 3-42
	[in] nInBufLen	nInBufLen 大于等于 szInBuf 结构体的大小
	[in] cbTransFile	回调函数 fTransFileCallBack
	[in] dwUserData	用户自定义数据
	[in] waittime	等待超时时间
返回值	● 开始发送黑白名单/下载黑白名单，返回名单文件句柄，大于 0 为有效句柄；小于等于 0 为无效句柄 ● 其他类型操作，成功返回大于 0；失败返回小于等于 0	
说明	无	

表3-42 文件控制类型

nTransType 枚举类型	值	含义	szInBuf
DH_DEV_BLACKWHITE TRANS_START	0x0003	开始发送黑白名单	DHDEV_BLACKWHITE_LIST _INFO

nTransType 枚举类型	值	含义	szInBuf
DH_DEV_BLACKWHITE TRANS_SEND	0x0004	发送黑白名单	LONG，具体为开始发送文件返回的句柄
DH_DEV_BLACKWHITE TRANS_STOP	0x0005	停止发送黑白名单	LONG，具体为开始发送文件返回的句柄
DH_DEV_BLACKWHITE _LOAD	0x0006	下载黑白名单	DHDEV_LOAD_BLACKWHITE_LIST_INFO
DH_DEV_BLACKWHITE _LOAD_STOP	0x0007	停止下载黑白名单	LONG，开始下载文件返回的句柄

3.3.3 语音对讲

3.3.3.1 获取设备支持对讲类型 CLIENT_GetDevProtocolType

表3-43 获取设备支持对讲类型 CLIENT_GetDevProtocolType

选项	说明	
描述	获取设备支持对讲类型	
函数	BOOL CLIENT_GetDevProtocolType(LLONG lLoginID, EM_DEV_PROTOCOL_TYPE *pemProtocolType);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[out]pemProtocolType	设备支持的协议类型，对应 EM_DEV_PROTOCOL_TYPE 结构体
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.3.3.2 设置设备语音对讲工作模式 CLIENT_SetDeviceMode

表3-44 设置设备语音对讲工作模式 CLIENT_SetDeviceMode

选项	说明	
描述	设置设备语音对讲工作模式	
函数	BOOL CLIENT_SetDeviceMode(LLONG lLoginID, EM_USEDEV_MODE emType, void *pValue);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]emType	枚举值
	[in]pValue	与枚举值对应的结构体数据指针，请参见表 3-45
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

emType 和 pValue 对照关系请参见表 3-45。

表3-45 emType 和 pValue 对照关系

emType	描述	pValue
DH_TALK_ENCODE_TYPE	指定某种格式进行对讲	DHDEV_TALKDECODE_INFO
DH_TALK_CLIENT_MODE	设置语音对讲客户端方式	无
DH_TALK_SPEAK_PARAM	设置语音对讲喊话参数	NET_SPEAK_PARAM
DH_TALK_MODE3	设置三代设备的语音对讲参数	NET_TALK_EX

3.3.3.3 开启对讲 CLIENT_StartTalkEx

表3-46 开启对讲 CLIENT_StartTalkEx

选项	说明	
描述	打开语音对讲	
函数	LLONG CLIENT_StartTalkEx(LLONG lLoginID, pfAudioDataCallBack pfcb, LDWORD dwUser);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]pfcb	音频数据回调函数
	[in]dwUser	音频数据回调函数的参数
返回值	● 成功返回非 0 ● 失败返回 0	
说明	无	

3.3.3.4 关闭对讲 CLIENT_StopTalkEx

表3-47 关闭对讲 CLIENT_StopTalkEx

选项	说明	
描述	关闭语音对讲	
函数	BOOL CLIENT_StopTalkEx(LLONG lTalkHandle);	
参数	[in]lTalkHandle	CLIENT_StartTalkEx 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.3.3.5 开启录音 CLIENT_RecordStartEx

表3-48 开启录音 CLIENT_RecordStartEx

选项	说明
描述	开启本地录音

选项	说明	
函数	BOOL CLIENT_RecordStartEx(LLONG lLoginID);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	此接口只在 Windows 下有效	

3.3.3.6 关闭录音 CLIENT_RecordStopEx

表3-49 关闭录音 CLIENT_RecordStopEx

选项	说明	
描述	关闭本地录音	
函数	BOOL CLIENT_RecordStopEx(LLONG lLoginID);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	此接口只在 Windows 下有效	

3.3.3.7 发送语音 CLIENT_TalkSendData

表3-50 发送语音 CLIENT_TalkSendData

选项	说明	
描述	发送音频数据给设备	
函数	LONG CLIENT_TalkSendData(LLONG lTalkHandle, char *pSendBuf, DWORD dwBufSize);	
参数	[in]lTalkHandle	CLIENT_StartTalkEx 的返回值
	[in]pSendBuf	需要发送的音频数据块的指针
	[in]dwBufSize	需要发送的音频数据块的长度，单位：字节
返回值	● 成功返回音频数据块的长度 ● 失败返回-1	
说明	无	

3.3.3.8 解码语音 CLIENT_AudioDecEx

表3-51 解码语音 CLIENT_AudioDecEx

选项	说明
描述	解码音频数据

选项	说明	
函数	BOOL CLIENT_AudioDecEx(LLONG lTalkHandle, char *pAudioDataBuf, DWORD dwBufSize);	
参数	[in]lTalkHandle	CLIENT_StartTalkEx 的返回值
	[in]pAudioDataBuf	需要解码的音频数据块的指针
	[in]dwBufSize	需要解码的音频数据块的长度，单位：字节
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.3.3.9 CLIENT_SetDVRMessCallBack

设置设备请求对方发起对讲事件回调函数接口 CLIENT_SetDVRMessCallBack，接口函数请参见“3.3.1.4 设置车辆位置信息回调函数接口 CLIENT_SetDVRMessCallBack”。

3.3.3.10 CLIENT_StartListenEx

订阅设备请求对方发起对讲事件信息接口 CLIENT_StartListenEx，接口函数请参见“3.3.1.5 订阅车辆位置信息接口 CLIENT_StartListenEx”。

3.3.3.11 CLIENT_StopListen

停止订阅设备请求对方发起对讲事件信息接口，接口函数请参见“3.3.1.6 停止订阅车辆位置信息接口 CLIENT_StopListen”。

3.3.4 点阵屏字符控制

3.3.4.1 设置点阵屏显示信息配置 CLIENT_SetConfig

接口函数请参见“3.3.1.2 设置道闸配置 CLIENT_SetConfig”。

其中 emCfgOpType 为 NET_EM_CFG_TRAFFIC_LATTICE_SCREEN。

3.3.4.2 获取点阵屏显示信息配置 CLIENT_GetConfig

接口函数请参见“3.3.1.3 获取道闸配置 CLIENT_GetConfig”。

其中 emCfgOpType 为 NET_EM_CFG_TRAFFIC_LATTICE_SCREEN。

3.3.5 车位指示灯本机配置

3.3.5.1 组成配置 CLIENT_PacketData

表3-52 组成配置 CLIENT_PacketData

选项	说明	
描述	组成配置	
函数	BOOL CLIENT_PacketData(char* szCommand, LPVOID lpInBuffer, DWORD dwInBufferSize, char* szOutBuffer, DWORD dwOutBufferSize);	
参数	[in] szCommand	命令参数 车位指示灯本机配置： CFG_CMD_PARKING_SPACE_LIGHT_GROUP
	[in] lpInBuffer	输入缓冲
	[in] dwInBufferSize	输入缓冲大小
	[out] szOutBuffer	输出缓冲
	[in] dwOutBufferSize	输出缓冲大小
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.5.2 设置配置 CLIENT_SetNewDevConfig

表3-53 设置配置 CLIENT_SetNewDevConfig

选项	说明	
描述	设置配置	
函数	BOOL CLIENT_SetNewDevConfig(LLONG lLoginID, char* szCommand, int nChannelID, char* szInBuffer, DWORD dwInBufferSize, int *error, int *restart, int waittime=500);	
[in] dwInBufferSize	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in] szCommand	命令参数 车位指示灯本机配置： CFG_CMD_PARKING_SPACE_LIGHT_GROUP

选项	说明	
	[in] nChannelID	通道号
	[in] szInBuffer	输入缓存，用户存储组成的用于设置配置的 json 串信息
	[in] dwInBufferSize	缓存地址大小
	[out] error	错误码地址
	[in] restart	重启标志地址
	[in] waittime	设置配置超时时间
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.5.3 获取配置 CLIENT_GetNewDevConfig

表3-54 获取配置 CLIENT_GetNewDevConfig

选项	说明	
描述	获取配置	
函数	BOOL CLIENT_GetNewDevConfig(LLONG lLoginID, char* szCommand, int nChannelID, char* szOutBuffer, DWORD dwOutBufferSize, int *error, int waittime=500);	
[in] dwInBufferSize	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in] szCommand	命令参数 车位指示灯本机配置： CFG_CMD_PARKING_SPACE_LIGHT_GROUP
	[in] nChannelID	通道号
	[out] szOutBuffer	输出缓存，用户存储从设备端获取的 json 字符串信息
	[in] dwOutBufferSize	缓存地址大小
	[out] error	错误码地址
	[in] waittime	获取配置超时时间
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.5.4 解析配置 CLIENT_ParseData

表3-55 解析配置 CLIENT_ParseData

选项	说明
描述	解析配置
函数	BOOL CLIENT_ParseData(

选项	说明	
	char* szCommand, char* szInBuffer, LPVOID lpOutBuffer, DWORD dwOutBufferSize, void* pReserved);	
参数	[in] szCommand	命令参数 车位指示灯本机配置： CFG_CMD_PARKING_SPACE_LIGHT_GROUP
	[in] szInBuffer	输入缓冲，字符配置缓冲
	[out]lpOutBuffer	输出缓冲
	[in]dwOutBufferSize	输出缓冲的大小
	[in] pReserved	保留参数
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

3.3.6 车位状态对应的车位指示灯配置

3.3.6.1 设置车位状态对应的车位指示灯配置 CLIENT_SetConfig

接口函数请参见“3.3.1.2 设置道闸配置 CLIENT_SetConfig”。

其中 emCfgOpType 为 NET_EM_CFG_PARKINGSPACELIGHT_STATE。

3.3.6.2 获取车位状态对应的车位指示灯配置 CLIENT_GetConfig

接口函数请参见“3.3.1.3 获取道闸配置 CLIENT_GetConfig”。

其中 emCfgOpType 为 NET_EM_CFG_PARKINGSPACELIGHT_STATE。

第 4 章 回调函数定义

4.1 搜索设备回调函数 fSearchDevicesCB

表4-1 搜索设备回调函数 fSearchDevicesCB

选项	说明	
描述	搜索设备回调函数	
函数	typedef void(CALLBACK *fSearchDevicesCB)(DEVICE_NET_INFO_EX * pDevNetInfo, void* pUserData);	
参数	[out]pDevNetInfo	搜索的设备信息
	[out]pUserData	用户数据
返回值	无	
说明	无	

4.2 异步搜索设备回调函数 fSearchDevicesCBEx

表4-2 异步搜索设备回调函数 fSearchDevicesCBEx

选项	说明	
描述	搜索设备回调函数	
函数	typedef void(CALLBACK * fSearchDevicesCBEx)(LLONG lSearchHandle, DEVICE_NET_INFO_EX2 *pDevNetInfo, void* pUserData);	
参数	[out]lSearchHandle	搜索句柄
	[out]pDevNetInfo	搜索的设备信息
	[out]pUserData	用户数据
返回值	无	
说明	无	

4.3 断线回调函数 fDisConnect

表4-3 断线回调函数 fDisConnect

选项	说明
描述	断线回调函数

选项	说明	
函数	typedef void (CALLBACK *fDisConnect)(LLONG lLoginID, char* pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
参数	[out] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[out] pchDVRIP	断线的设备 IP
	[out] nDVRPort	断线的设备端口
	[out] dwUser	回调函数的用户参数
返回值	无	
说明	无	

4.4 断线重连回调函数 fHaveReConnect

表4-4 断线重连回调函数 fHaveReConnect

选项	说明	
描述	断线重连回调函数	
函数	typedef void (CALLBACK *fHaveReConnect)(LLONG lLoginID, char* pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
参数	[out] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[out] pchDVRIP	断线后重连成功的设备 IP
	[out] nDVRPort	断线后重连成功的设备端口
	[out] dwUser	回调函数的用户参数
返回值	无	
说明	无	

4.5 实时监视数据回调函数 fRealDataCallBackEx2

表4-5 实时监视数据回调函数 fRealDataCallBackEx2

选项	说明
描述	实时监视数据回调函数

选项	说明	
函数	typedef void (CALLBACK *fRealDataCallBackEx2)(LLONG lRealHandle, DWORD dwDataType, BYTE* pBuffer, DWORD dwBufSize, LLONG param, LDWORD dwUser);	
参数	[out] lRealHandle	CLIENT_RealPlayEx 的返回值
	[out] dwDataType	数据类型 ● 0 表示原始数据 ● 1 表示带有帧信息的数据 ● 2 表示 YUV 数据 ● 3 表示 PCM 音频数据
	[out] pBuffer	监视数据块地址
	[out] dwBufSize	监视数据块的长度，字节为单位
	[out] param	回调数据参数结构体，dwDataType 值不同类型不同 ● dwDataType 为 0 时，param 为空指针 ● dwDataType 为 1 时，param 为 tagVideoFrameParam 结构体指针 ● dwDataType 为 2 时，param 为 tagCBYUVDataParam 结构体指针 ● dwDataType 为 3 时，param 为 tagCBPCMDataParam 结构体指针
	[out] dwUser	回调函数的用户参数
返回值	无	
说明	无	

4.6 下载媒体文件进度回调 fDownLoadPosCallBack

表4-6 下载媒体文件进度回调 fDownLoadPosCallBack

选项	说明	
描述	下载媒体文件进度回调	
函数	typedef void (CALLBACK *fDownLoadPosCallBack)(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, LDWORD dwUser);	
参数	[out]lPlayHandle	CLIENT_DownloadMediaFile 返回值
	[out]dwTotalSize	总大小

选项	说明	
	[out]dwDownLoadSize	已下载数据的大小 ● -1：表示下载结束 ● -2：下载时写数据出错
	[out]dwUser	用户数据
返回值	无	
说明	无	

4.7 智能事件信息回调 fAnalyzerDataCallBack

表4-7 智能事件信息回调 fAnalyzerDataCallBack

选项	说明	
描述	智能事件信息回调	
函数	typedef int (CALLBACK *fAnalyzerDataCallBack)(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE* pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void* reserved);	
参数	[out]lAnalyzerHandle	CLIENT_RealLoadPictureEx 返回值
	[out]dwAlarmType	智能交通事件类型，请参见表 3-30
	[out]pAlarmInfo	事件信息缓存，请参见表 3-30
	[out]pBuffer	图片缓存
	[out]dwBufSize	图片缓存大小
	[out]dwUser	用户数据
	[out]reserved	保留
返回值	无	
说明	无	

4.8 交通车流量统计回调 fFluxStatDataCallBack

表4-8 交通车流量统计回调 fFluxStatDataCallBack

选项	说明
描述	智能事件信息回调

选项	说明	
函数	typedef int (CALLBACK *fFluxStatDataCallBack)(LLONG lFluxStatHandle, DWORD dwEventType, void* pEventInfo, BYTE* pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void* reserved);	
参数	[out]lFluxStatHandle	CLIENT_StartTrafficFluxStat 返回值
	[out]dwEventType	事件类型
	[out]pEventInfo	车流量事件信息
	[out]pBuffer	数据缓存
	[out]dwBufSize	数据大小
	[out]dwUser	用户数据
	[out]nSequence	次序
	[out]reserved	保留
返回值	无	
说明	pEventInfo 对应结构体 DEV_EVENT_TRAFFIC_FLOWSTAT_INFO	

4.9 文件传输回调 fTransFileCallBack

表4-9 文件传输回调 fTransFileCallBack

选项	说明	
描述	文件传输回调	
函数	typedef int (CALLBACK *fFluxStatDataCallBack)(LLONG lHandle, int nTransType, int nState, int nSendSize, int nTotalSize, LDWORD dwUser);	
参数	[out] lHandle	文件传输句柄
	[out] nTransType	文件传输类型
	[out] nState	文件传输状态
	[out] nSendSize	发送的文件长度
	[out] nTotalSize	文件总的大小
	[out] dwUser	用户自定义数据
返回值	无	
说明	无	

4.10 音频数据回调函数 pfAudioDataCallBack

表4-10 音频数据回调函数 pfAudioDataCallBack

选项	说明	
描述	语音对讲的音频数据回调函数	
函数	typedef void (CALLBACK *pfAudioDataCallBack)(LLONG lTalkHandle, char *pDataBuf, DWORD dwBufSize, BYTE byAudioFlag, LDWORD dwUser);	
参数	[out]lTalkHandle	CLIENT_StartTalkEx 的返回值
	[out]pDataBuf	音频数据块地址
	[out]dwBufSize	音频数据块的长度，单位：字节
	[out]byAudioFlag	数据类型标志 ● 0 表示来自本地采集 ● 1 表示来自设备发送
	[out]dwUser	回调函数的用户参数
返回值	无	
说明	无	

附录1 法律声明

商标声明

- VGA 是 IBM 公司的商标。
- Windows 标识和 Windows 是微软公司的商标或注册商标。
- 在本文档中可能提及的其他商标或公司的名称，由其各自所有者拥有。

责任声明

- 在适用法律允许的范围内，在任何情况下，本公司都不对因本文档中相关内容及描述的产品而产生任何特殊的、附随的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉、文档丢失或预期节约的损失进行赔偿。
- 本文档中描述的产品均“按照现状”提供，除非适用法律要求，本公司对文档中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证。

隐私保护提醒

您安装了我们的产品，您可能会采集人脸、指纹、车牌、邮箱、电话、GPS 等个人信息。在使用产品过程中，您需要遵守所在地区或国家的隐私保护法律法规要求，保障他人的合法权益。如，提供清晰、可见的标牌，告知相关权利人视频监控区域的存在，并提供相应的联系方式。

关于本文档

- 产品请以实物为准，本文档仅供参考。
- 本公司保留随时维护本文档中任何信息的权利，维护的内容将会在本文档的新版本中加入，恕不另行通知。
- 本文档如有不准确或不详尽的地方，或印刷错误，请以公司最终解释为准。
- 本文档供多个型号产品做参考，每个产品的具体操作不逐一例举，请用户根据实际产品自行对照操作。
- 如不按照本文档中的指导进行操作，因此而造成的任何损失由使用方自行承担。
- 如获取到的 PDF 文档无法打开，请将阅读工具升级到最新版本或使用其他主流阅读工具。

附录2 网络安全建议

保障设备基本网络安全的必须措施：

1. 使用复杂密码

请参考如下建议进行密码设置：

- 长度不小于 8 个字符。
- 至少包含两种字符类型，字符类型包括大小写字母、数字和符号。
- 不包含帐户名称或帐户名称的倒序。
- 不要使用连续字符，如 123、abc 等。
- 不要使用重叠字符，如 111、aaa 等。

2. 及时更新固件和客户端软件

- 按科技行业的标准作业规范，设备的固件需要及时更新至最新版本，以保证设备具有最新的功能和安全性。设备接入公网情况下，建议开启在线升级自动检测功能，便于及时获知厂商发布的固件更新信息。
- 建议您下载和使用最新版本客户端软件。

增强设备网络安全的建议措施：

1. 物理防护

建议您对设备（尤其是存储类设备）进行物理防护，比如将设备放置在专用机房、机柜，并做好门禁权限和钥匙管理，防止未经授权的人员进行破坏硬件、外接设备（例如 U 盘、串口）等物理接触行为。

2. 定期修改密码

建议您定期修改密码，以降低被猜测或破解的风险。

3. 及时设置、更新密码重置信息

设备支持密码重置功能，为了降低该功能被攻击者利用的风险，请您及时设置密码重置相关信息，包含预留手机号/邮箱、密保问题，如有信息变更，请及时修改。设置密保问题时，建议不要使用容易猜测的答案。

4. 开启帐户锁定

出厂默认开启帐户锁定功能，建议您保持开启状态，以保护帐户安全。在攻击者多次密码尝试失败后，其对应帐户及源 IP 将会被锁定。

5. 更改 HTTP 及其他服务默认端口

建议您将 HTTP 及其他服务默认端口更改为 1024～65535 间的任意端口，以减小被攻击者猜测服务端口的风险。

6. 使能 HTTPS

建议您开启 HTTPS，通过安全的通道访问 Web 服务。

7. 启用白名单

建议您开启白名单功能，开启后仅允许白名单列表中的 IP 访问设备。因此，请务必将您的电脑 IP 地址，以及配套的设备 IP 地址加入白名单列表中。

8. MAC 地址绑定

建议您在设备端将其网关设备的 IP 与 MAC 地址进行绑定，以降低 ARP 欺骗风险。

9. 合理分配帐户及权限

根据业务和管理需要，合理新增用户，并合理为其分配最小权限集合。

10. 关闭非必需服务，使用安全的模式

如果没有需要，建议您关闭 SNMP、SMTP、UPnP 等功能，以降低设备面临的风险。

如果有需要，强烈建议您使用安全的模式，包括但不限于：

- SNMP：选择 SNMP v3，并设置复杂的加密密码和鉴权密码。
- SMTP：选择 TLS 方式接入邮箱服务器。

- FTP：选择 SFTP，并设置复杂密码。
- AP 热点：选择 WPA2-PSK 加密模式，并设置复杂密码。

11. 音视频加密传输

如果您的音视频数据包含重要或敏感内容，建议启用加密传输功能，以降低音视频数据传输过程中被窃取的风险。

12. 使用 PoE 方式连接设备

如果设备支持 PoE 功能，建议采用 PoE 方式连接设备，使摄像机与其他网络隔离。

13. 安全审计

- 查看在线用户：建议您不定期查看在线用户，识别是否有非法用户登录。
- 查看设备日志：通过查看日志，可以获知尝试登录设备的 IP 信息，以及已登录用户的关键操作信息。

14. 网络日志

由于设备存储容量限制，日志存储能力有限，如果您需要长期保存日志，建议您启用网络日志功能，确保关键日志同步至网络日志服务器，便于问题回溯。

15. 安全网络环境的搭建

为了更好地保障设备的安全性，降低网络安全风险，建议您：

- 关闭路由器端口映射功能，避免外部网络直接访问路由器内网设备的服务。
- 根据实际网络需要，对网络进行划区隔离：若两个子网间没有通信需求，建议使用 VLAN、网闸等方式对其进行网络分割，达到网络隔离效果。
- 建立 802.1x 接入认证体系，以降低非法终端接入专网的风险。
- 启用设备的防火墙或者黑白名单功能，降低设备可能遭受攻击的风险。