NetSDK 编程指导手册

（智能分析服务器分册）



V2.0.0

前言

目的

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。

本文档描述了智能视频分析产品的通用业务涉及的 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》，基础核心业务流程（例如初始化、登录、普通报警和智能报警等）详见《NetSDK 编程指导手册》。

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、产品经理和项目经理。

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
V2.0.0	基础核心内容参考《NetSDK 编程指导手册》	2025.02
V1.0.4	全文优化语言。	2023.02
V1.0.3	● 删除 avnetsdk 依赖库信息。 ● 新增 StreamConvertor 依赖库。 ● 删除 fisheye 鱼眼矫正库。	2021.05
V1.0.2	修改登录设备和搜索设备接口函数	2020.02
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。

名词	说明
主码流	视频码流类型的一种，一般指码流大、压缩比小、图像质量高的码流，在网络资源不受限的前提下能得到更好的体验。
辅码流	较主码流分辨率、清晰度都低一些，但占用的网络资源少；用户可以根据不同的适用场景选择不同的码流类型。
视频通道	SDK 与设备通信，视频流传输的抽象概念。如存储设备（NVR 等），挂载若干前端设备（SD，IPC 等），存储设备（NVR 等）将前端设备（SD，IPC 等）抽象为视频通道进行管理。若 SDK 与单个前端设备直连，则视频通道号一般为 0。
登录句柄	在 SDK 内部维护，与设备通信的唯一标示符，为非 0 整数。
预览句柄	在 SDK 内部维护，与设备的某个视频通道进行数据传输的唯一标示符，为非 0 整数。
订阅句柄	在 SDK 内部维护，订阅设备实时信息上报（智能事件等）的唯一标示符，为非 0 整数。
能力集	设备支持某项或某套功能的能力集合。不同的设备型号能力集可能会有差异。
穿越围栏检测	自动检测目标翻越围栏的行为。
绊线入侵检测	自动检测穿越警戒线的行为。
区域入侵检测	自动检测目标入侵警戒区的行为，包括“穿越区域”和“在区域内”。
物品遗留检测	自动检测防区内新出现并且静止的物品。
物品保全检测	被选中的物体被长时间遮挡或者移动后报警。
物品搬移检测	防区内物品被搬离原覆盖区域后报警。
徘徊检测	自动检测防区内逗留时间超过设定时间的入侵行为。
视频异常检测	自动检测视频场景变化、视频丢失、视频遮挡、过亮、过暗、清晰度检测、条纹检测、噪声检测、偏色检测等异常行为。
声音异常检测	自动检测通道内声音异常的行为。
攀高检测	自动检测防区内人员攀高行为。
斗殴检测	自动检测防区内人员打架行为。
离岗检测	自动检测防区内人员离岗行为。
起身检测	自动检测目标起床的行为。
警戒线穿越检测	自动检测穿越警戒线的行为。
警戒区入侵检测	自动检测目标入侵警戒区的行为，包括“进入警戒区”、“离开警戒区”和“在警戒区内”。
正常人脸检测	人脸监控视频中正常人脸检测。
异常人脸检测	人脸监控视频中的异常人脸检测，异常人脸包括嘴部遮挡和眼部遮挡。
相邻人脸检测	人脸监控视频中的相邻人脸检测，相邻人脸主要指监控视频中出现两个及以上人脸的异常情况检测，即尾随现象的检测。
非法黏贴物贴条检测	出钞口监控视频中的非法黏贴物贴条检测。
操作区进入检测	出钞口监控视频中的操作区进入检测。
操作区离开检测	出钞口监控视频中的操作区离开检测。

名词	说明
操作区滞留检测	出钞口监控视频中的操作区滞留检测。

目录

前言 ..... **II**
名词解释 ..... **III**
第 1 章 内容简介 **1**
1.1 概述 1
1.2 适用性 2
1.3 应用场景 2
第 2 章 主要功能 **3**
2.1 SDK 初始化 3
2.1.1 简介 3
2.1.2 接口总览 3
2.1.3 流程说明 3
2.1.4 示例代码 4
2.2 设备初始化 4
2.2.1 简介 4
2.2.2 接口总览 5
2.2.3 流程说明 5
2.2.4 示例代码 8
2.3 设备登录 10
2.3.1 简介 10
2.3.2 接口总览 10
2.3.3 流程说明 10
2.3.4 示例代码 12
2.4 实时预览 13
2.4.1 简介 13
2.4.2 接口总览 13
2.4.3 流程说明 13
2.4.4 示例代码 17
2.5 监所专用 18
2.5.1 简介 18
2.5.2 接口总览 19
2.5.3 流程说明 19
2.5.4 示例代码 20
2.6 智能 ATM 22
2.6.1 简介 22
2.6.2 接口总览 23
2.6.3 流程说明 23
2.6.4 示例代码 24
2.7 客流量统计 27
2.7.1 简介 27
2.7.2 接口总览 28
2.7.3 流程说明 28
2.7.4 示例代码 31
第 3 章 接口说明 **36**

3.1 SDK 初始化 36
3.1.1 SDK 初始化 CLIENT_Init 36
3.1.2 SDK 清理 CLIENT_Cleanup 36
3.1.3 设置断线重连回调函数 CLIENT_SetAutoReconnect 36
3.1.4 设置网络参数 CLIENT_SetNetworkParam 37
3.2 设备初始化 37
3.2.1 搜索设备 CLIENT_StartSearchDevicesEx 37
3.2.2 设备初始化 CLIENT_InitDevAccount 37
3.2.3 获取密码重置信息 CLIENT_GetDescriptionForResetPwd 38
3.2.4 检验安全码是否有效 CLIENT_CheckAuthCode 38
3.2.5 重置密码 CLIENT_ResetPwd 39
3.2.6 获取密码规则 CLIENT_GetPwdSpecification 39
3.2.7 停止搜索设备 CLIENT_StopSearchDevices 40
3.3 设备登录 40
3.3.1 高安全级别登录 CLIENT_LoginWithHighLevelSecurity 40
3.3.2 用户登出设备 CLIENT_Logout 41
3.4 实时预览 42
3.4.1 打开预览 CLIENT_RealPlayEx 42
3.4.2 关闭预览 CLIENT_StopRealPlayEx 43
3.4.3 保存预览数据 CLIENT_SaveRealData 43
3.4.4 停止保存预览数据 CLIENT_StopSaveRealData 43
3.4.5 设置预览数据回调 CLIENT_SetRealDataCallBackEx2 44
3.5 智能事件订阅 44
3.5.1 开始智能事件订阅 CLIENT_RealLoadPictureEx 44
3.5.2 停止智能事件订阅 CLIENT_StopLoadPic 46
3.6 客流量统计 46
3.6.1 开始订阅客流量统计 CLIENT_AttachVideoStatSummary 46
3.6.2 停止订阅客流量统计 CLIENT_DetachVideoStatSummary 46
3.6.3 开始查询客流量统计信息 CLIENT_StartFindNumberStat 47
3.6.4 继续查询客流量统计信息 CLIENT_DoFindNumberStat 47
3.6.5 停止查询客流量统计信息 CLIENT_StopFindNumberStat 48
第 4 章 回调函数 49
4.1 搜索设备回调函数 fSearchDevicesCB 49
4.2 异步搜索设备回调函数 fSearchDevicesCBEx 49
4.3 断线回调函数 fDisConnect 49
4.4 断线重连回调函数 fHaveReConnect 50
4.5 实时预览数据回调函数 fRealDataCallBackEx2 50
4.6 智能事件回调 fAnalyzerDataCallBack 51
4.7 客流量统计信息回调 fVideoStatSumCallBack 52
附录 1 法律声明 53
附录 2 网络安全建议 54

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、实时预览、智能事件上报与抓图、客流量统计。

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	播放库
dhnetsdk 辅助库	lvsDrawer.dll	图像显示库
	StreamConvertor.dll	转码库

- 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
libdhnetsdk.so 辅助库	libStreamConvertor.so	转码库

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。
- 辅助库用于预览、回放、对讲等功能的音视频码流解码以及本地音频采集。

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。

1.3 应用场景

SDK 在智能分析设备的应用场景如下：

- 一般通用的登录、预览、对讲、报警等功能。
- 作为客户端向智能分析设备订阅智能事件，处理智能事件上报业务；向智能分析设备订阅客流量统计，完成客流量统计的智能上报等。

第 2 章 主要功能

2.1 SDK 初始化

2.1.1 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。

2.1.2 接口总览

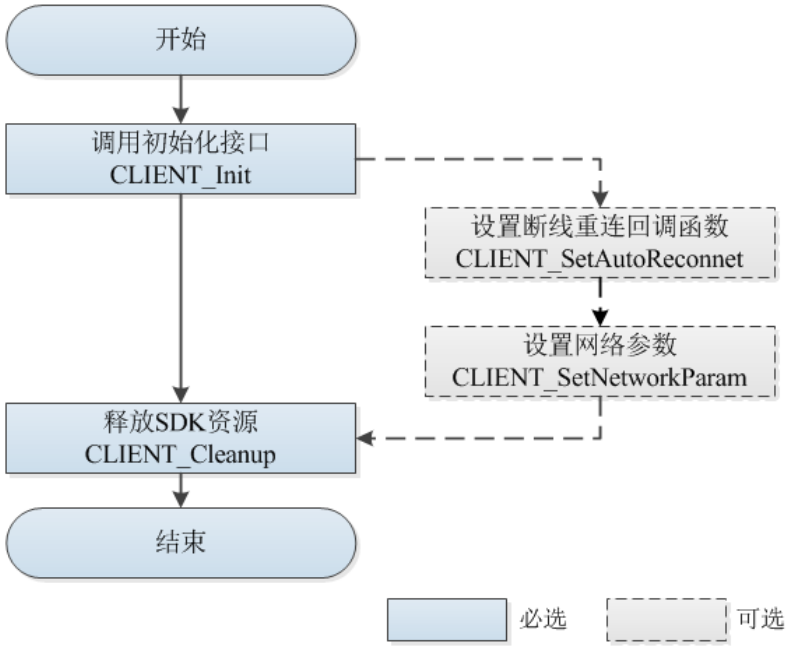
表2-1 SDK 初始化接口信息

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

2.1.3 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、实时预览和报警订阅等。
- 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预览、报警和智能图片订阅业务，其他业务无法恢复。

2.1.4 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();
```

2.2 设备初始化

2.2.1 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。

2.2.2 接口总览

表2-2 设备初始化接口信息

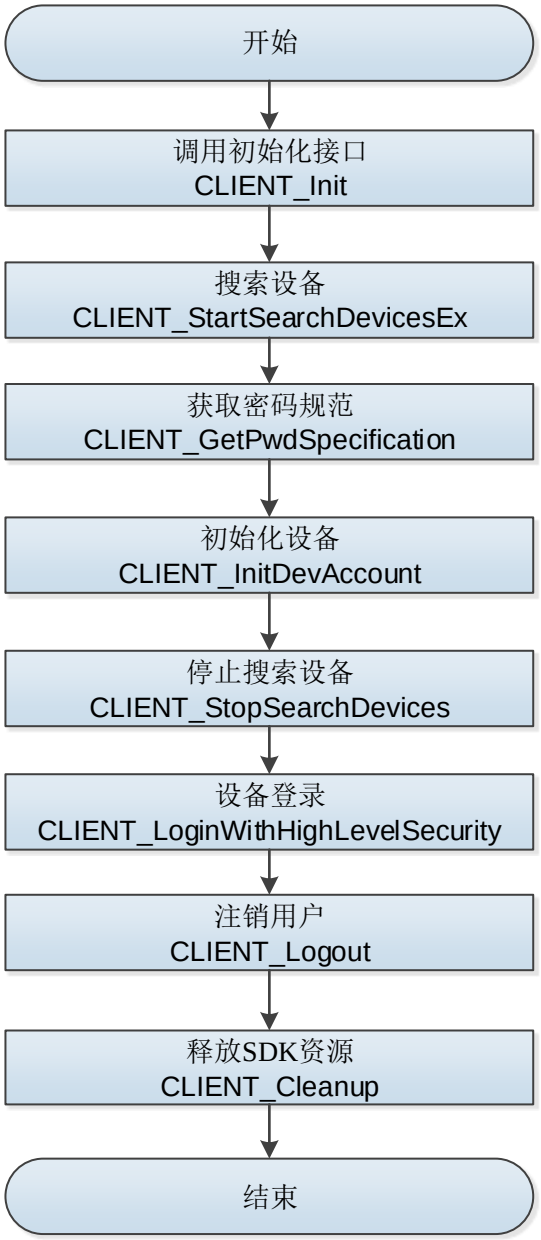
接口	说明
CLIENT_StartSearchDevicesEx	搜索局域网内的设备，找到未初始化设备
CLIENT_InitDevAccount	设备初始化接口
CLIENT_GetDescriptionForResetPwd	获取密码重置信息：手机号、邮箱和二维码信息
CLIENT_CheckAuthCode	校验安全码是否有效
CLIENT_ResetPwd	重置密码
CLIENT_GetPwdSpecification	获取密码规则
CLIENT_StopSearchDevices	停止搜索设备

2.2.3 流程说明

2.2.3.1 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。

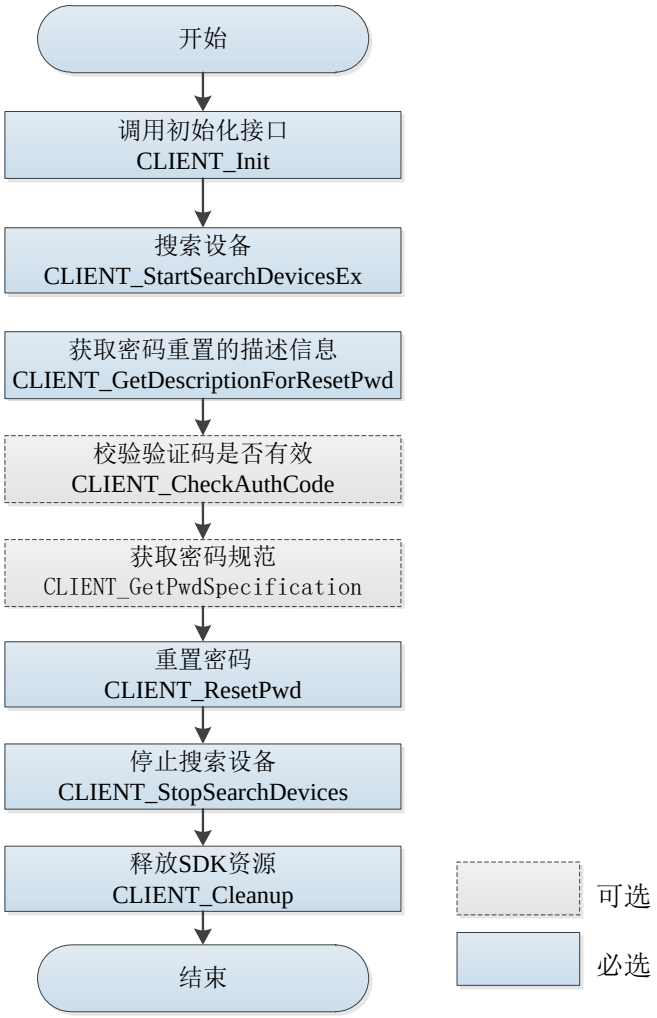
注意事项

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。

2.2.3.2 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。

2.2.4 示例代码

2.2.4.1 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);
```

2.2.4.2 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_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

2.3 设备登录

2.3.1 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。

2.3.2 接口总览

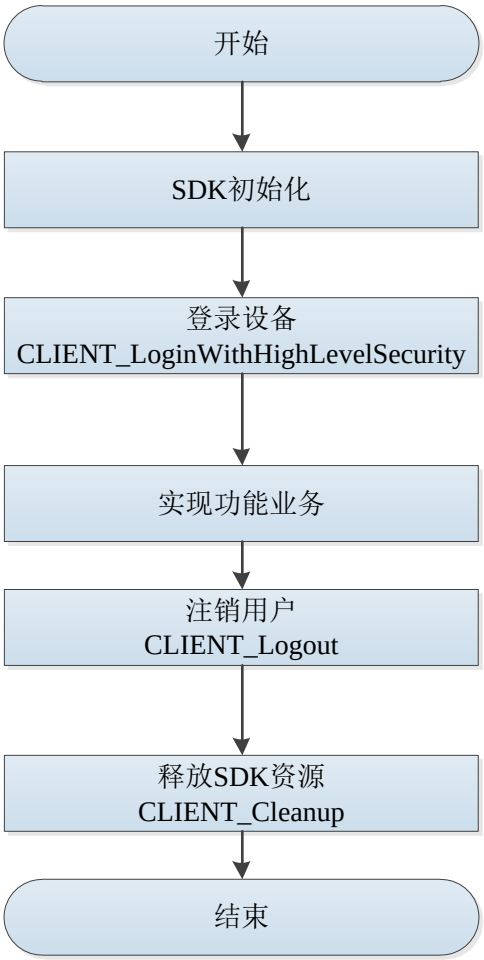
表2-3 设备登录接口信息

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。

2.3.3 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。例如打开预览后，在不需要使用预览时，用户应该调用结束预览的接口。
- 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);
```

2.3.4 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);
}
```

2.4 实时预览

2.4.1 简介

实时预览，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。

2.4.2 接口总览

表2-5 实时预览接口信息

接口	说明
CLIENT_RealPlayEx	开始实时预览扩展接口
CLIENT_StopRealPlayEx	停止实时预览扩展接口
CLIENT_SaveRealData	开始本地保存实时预览数据
CLIENT_StopSaveRealData	停止本地保存实时预览数据
CLIENT_SetRealDataCallBackEx2	设置实时预览数据回调函数扩展接口

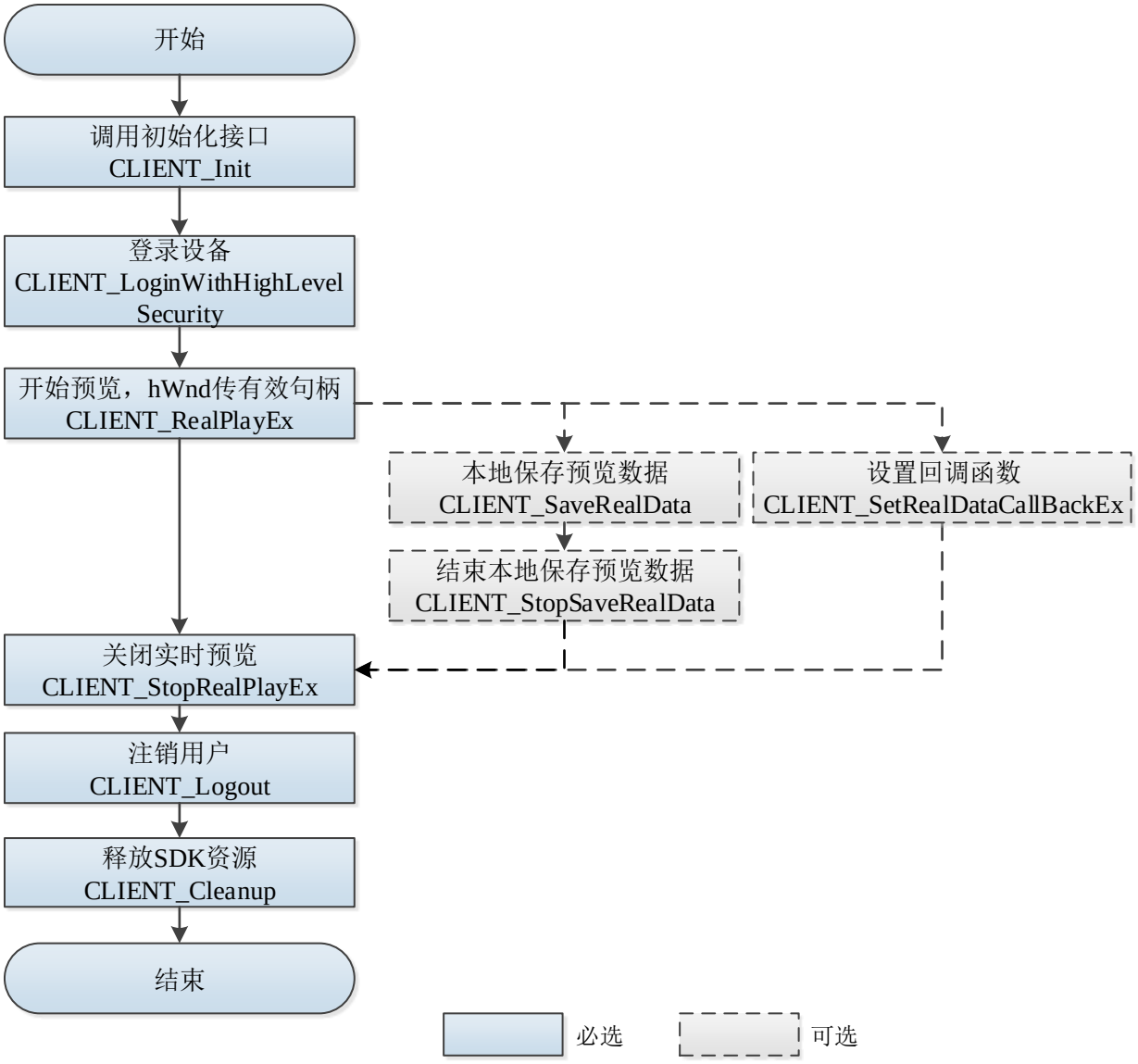
2.4.3 流程说明

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。

2.4.3.1 SDK 解码播放

SDK 内部调用辅助库里的 PlaySDK 库实现实时播放。SDK 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 启动实时预览，参数 **hWnd** 为有效窗口句柄。

步骤4　（可选）调用 CLIENT_SaveRealData 开始保存预览数据。

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　实时预览使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时预览。

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：接口内申请预览资源需和设备做一些约定，然后才请求预览数据，过程中有一些超时时间的设定（请参见 NET_PARAM 结构体），其中与预览相关的字段为 **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：

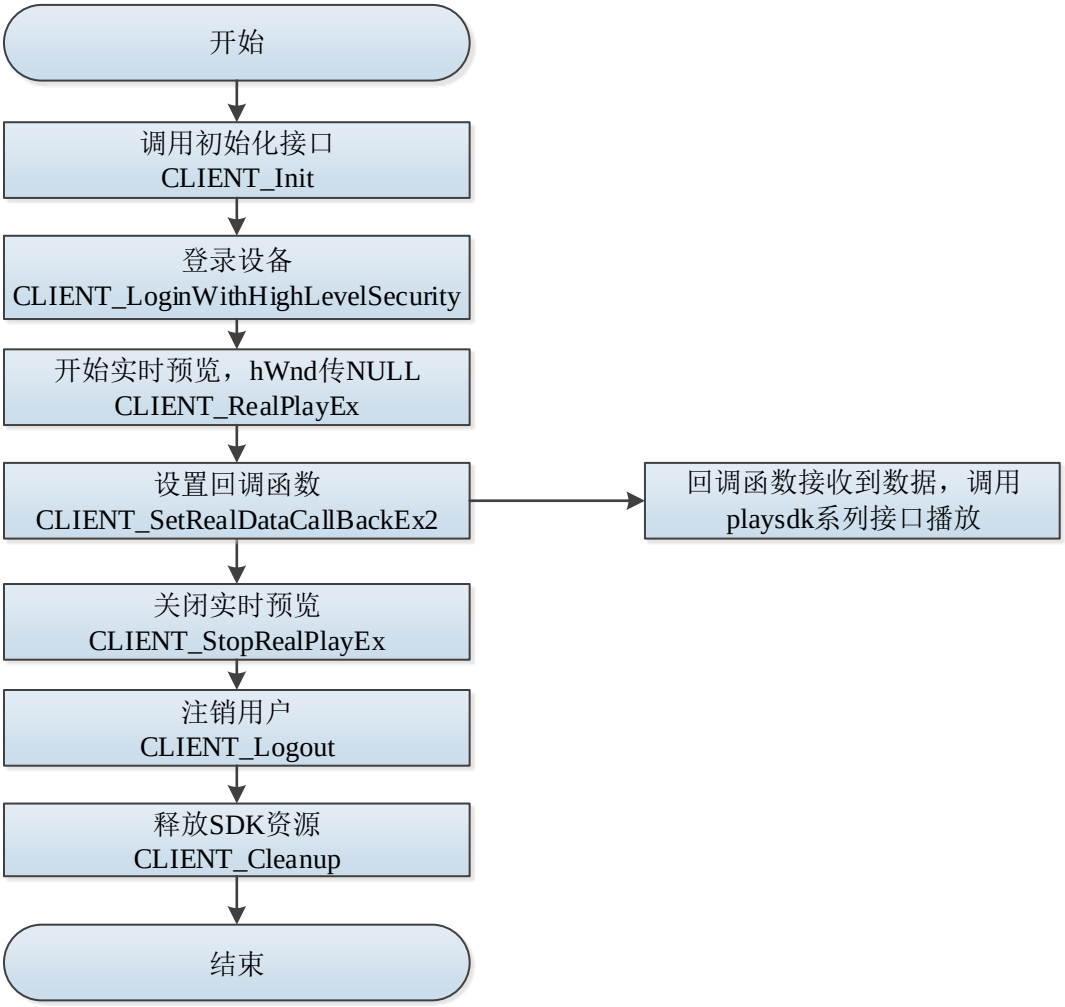
```
NET_PARAM stuNetParam = {0};
stuNetParam. nGetConnInfoTime = 5000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- 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预览，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“2.4.3.2 调用第三方解码播放库”。

2.4.3.2 调用第三方解码播放库

SDK 回调实时预览码流给用户，用户调用 PlaySDK 进行解码播放。用户调用第三方解码播放流程如图 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 启动实时预览，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　实时预览使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时预览。
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。

2.4.4 示例代码

2.4.4.1 SDK 解码播放

```
//以开启第一路的主码流预览为例，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);
}
```

2.4.4.2 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预览为例
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 预览 demo 源码
    printf("receive real data, param: lRealHandle[%p], dwDataType[%d], pBuffer[%p], dwBufSize[%d]\n", lRealHandle, dwDataType, pBuffer, dwBufSize);
}
```

2.5 监所专用

2.5.1 简介

产品介绍

监所专用功能集音视频数据采集、传输（有线）、存储、智能检测报警功能于一体，除具有普通室外智能检测功能外，还具有打架斗殴、起身检测、离岗检测、攀高检测和视频异常等室内检测功能。

产品型号

监所专用功能主要应用于 DH-IVS-IP7200 设备。

SDK 接入功能

用户接入 SDK，向 DH-IVS-IP7200 设备订阅行为分析智能事件，并获取已订阅的智能事件及数据信息。

说明

用户在订阅行为检测事件前，需要在设备 Web 端配置行为检测触发规则。

DH-IVS-IP7200 系列产品支持的行为检测请参见表 2-6。

表2-6 支持的行为检测

场景	支持的行为检测
普通场景	穿越围栏、绊线入侵、区域入侵、物品遗留、物品保全、物品搬移、徘徊检测、视频异常等行为分析检测。
监室场景	声音异常、区域入侵、攀高检测、斗殴检测、离岗检测、起身检测、视频异常、徘徊检测等行为分析检测。

2.5.2 接口总览

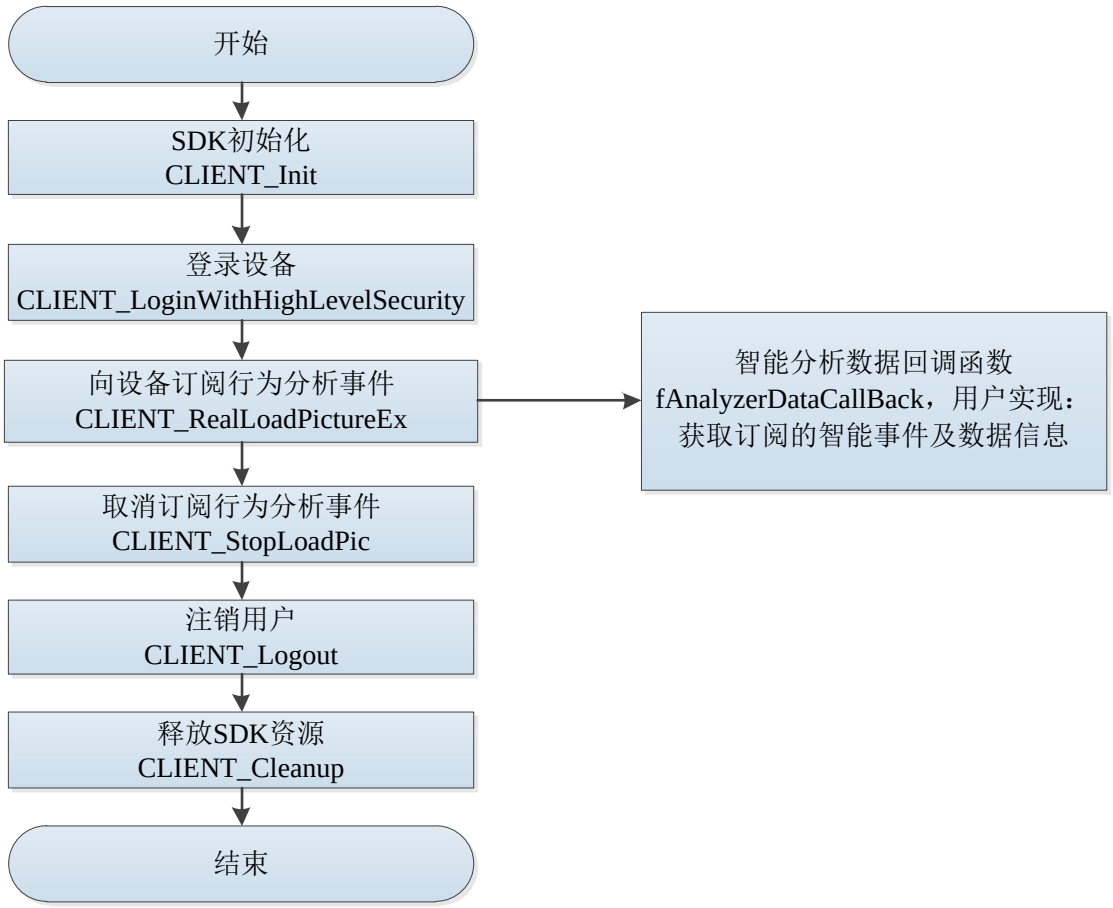
表2-7 监所专用的接口信息

接口	说明
CLIENT_RealLoadPictureEx	开始智能事件分析数据订阅扩展接口
CLIENT_StopLoadPic	停止智能事件分析数据订阅接口
fAnalyzerDataCallBack	智能事件分析数据回调接口

2.5.3 流程说明

监所专用 SDK 行为分析智能事件订阅如图 2-7 所示。

图2-7 监所专用行为分析智能事件



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 开始智能事件订阅。

步骤4　调用 fAnalyzerDataCallBack 智能分析回调函数处理已订阅的智能事件和数据。

步骤5　调用 CLIENT_StopLoadPic 函数取消智能事件订阅。

步骤6　业务使用完后，调用 CLIENT_Logout 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

回调函数数据处理：不建议在 fAnalyzerDataCallBack 函数中进行耗时高的 I/O 或延时等操作，如本地保存图片，数据库插入等。Windows 下可使用 postMessage 将数据抛出去，开启线程处理。

2.5.4 示例代码

```
// 智能分析数据回调
// 不建议在该回调函数中调用 SDK 接口
// 通过 CLIENT_RealLoadPictureEx/CLIENT_RealLoadPicture 设置该回调函数，当设备端有智能图片事件上报时，SDK 会调用该函数
// nSequence 表示上传的相同图片情况，为 0 时表示是第一次出现，为 2 表示最后一次出现或仅出现一次，为 1 表示此次之后还有
// int nState = *(int*) reserved 表示当前回调数据的状态，为 0 表示当前数据为实时数据，为 1 表示当前回调数据是离线数据，为 2 时表示离线数据传送结束
// 返回值已废除，无特殊意义
int  CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    switch(dwAlarmType)
    {
        // 翻越围栏事件(对应 DEV_EVENT_CROSSFENCEDETECTION_INFO)
        case EVENT_IVS_CROSSFENCEDETECTION:
        {
            DEV_EVENT_CROSSFENCEDETECTION_INFO* pCrossFence = NULL;
            pCrossFence = (DEV_EVENT_CROSSFENCEDETECTION_INFO*)pAlarmInfo;
            ivsInfoEx.stuEventInfo[0].alarmAction = pCrossFence->bEventAction;
            ivsInfoEx.stuEventInfo[0].alarmType = dwAlarmType;
            strncpy(ivsInfoEx.stuEventInfo[0].szRuleName, pCrossFence->szName,
            (sizeof(pCrossFence->szName) >= sizeof(ivsInfoEx.stuEventInfo[0].szRuleName)
            ? sizeof(ivsInfoEx.stuEventInfo[0].szRuleName) : sizeof(pCrossFence->szName)));
            ivsInfoEx.stuEventInfo[0].nObjectNum = 1;
            memcpy(&ivsInfoEx.stuEventInfo[0].stuObject[0],&pCrossFence->stuObject,
             sizeof(pCrossFence->stuObject));
         }
            break;
```

```
        // 绊线/警戒线入侵（对应 DEV_EVENT_CROSSREGION_INFO）
        case EVENT_IVS_CROSSLINEDETECTION:
            {
                //.....................................................
            }
            break;
        .......
        default:
            printf("other event type[%d]\n", dwAlarmType);
            break;
}
// 下载图片
if (dwBufSize > 0 && NULL != pBuffer)
{
    // 预防同一时间收到多张图片，只通过接收时间来保存图片可能会覆盖，于是通过 i 来标记
    static int i;
    char szPicturePath[256] = "";
    time_t stuTime;
    time(&stuTime);
    char szTmpTime[128] = "";
    strftime(szTmpTime, sizeof(szTmpTime) - 1, "%y%m%d_%H%M%S", gmtime(&stuTime));
    _snprintf(szPicturePath, sizeof(szPicturePath)-1, "%d_%s.jpg", ++i, szTmpTime);
    FILE* pFile = fopen(szPicturePath, "wb");
    if (NULL == pFile)
    {
        return 0;
    }
    int nWrite = 0;
    while(nWrite != dwBufSize)
    {
        nWrite += fwrite(pBuffer + nWrite, 1, dwBufSize - nWrite, pFile);
    }

    fclose(pFile);
```

```
    }
    return 1;
}
//智能事件订阅代码
{
    // 资源初始化
    ……
    // 登录设备
    ……
    // 订阅智能图片报警
    LDWORD dwUser = 0;
    int nChannel = 0;
    // 每次设置对应一个通道，并且对应一种类型的事件
    // 如果要设置该通道上传所有类型的事件，可以将参数 dwAlarmType 设置为 EVENT_IVS_ALL
    // 如果需要设置一个通道上传两种事件，那么请调用两次 CLIENT_RealLoadPictureEx，并且传入不同的事件类型
    lRealLoadHandle = CLIENT_RealLoadPictureEx(lLoginHandle, nChannel, EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, dwUser, NULL);
    //……

    // 停止订阅图片报警
    CLIENT_StopLoadPic(lRealLoadHandle))
    // 退出设备
    CLIENT_Logout(lLoginHandle))
    CLIENT_Cleanup();
}
```

2.6 智能 ATM

2.6.1 简介

产品介绍

智能 ATM 功能指根据平台配置的分析视频、规则和功能，对前端视频设备进行检测分析，对出现的非法黏贴物贴条、异常人脸、徘徊、物品遗留等异常现象进行报警。主要应用于大华金融行业智能分析服务器 DH-IVS-IF70XX 系列产品。

产品型号

智能 ATM 功能主要应用于以下型号的产品：

- 16 路智能分析基础版：DH-IVS -IF7016-B
- 16 路智能分析高级版：DH-IVS –IF7016-A
- 16 路智能分析完整版：DH-IVS –IF7016-F
- 24 路智能分析基础版：DH-IVS –IF7024-B
- 24 路智能分析高级版：DH-IVS –IF7024-A
- 24 路智能分析完整版：DH-IVS –IF7024-F

SDK 接入功能

用户接入 SDK，向 DH-IVS-IF70XX 系列设备订阅行为分析智能事件，并获取已订阅的智能事件及数据信息。

说明

用户在订阅行为检测事件前，需要在 Web 端配置行为检测触发规则。

DH-IVS-IF70XX 系列产品支持的行为检测有：警戒线穿越检测，警戒区入侵检测，徘徊检测，物品遗留检测，物品搬移检测，正常人脸检测，异常人脸检测，相邻人脸检测，非法黏贴物贴条检测，操作区进入检测，操作区离开检测和操作区滞留检测。

2.6.2 接口总览

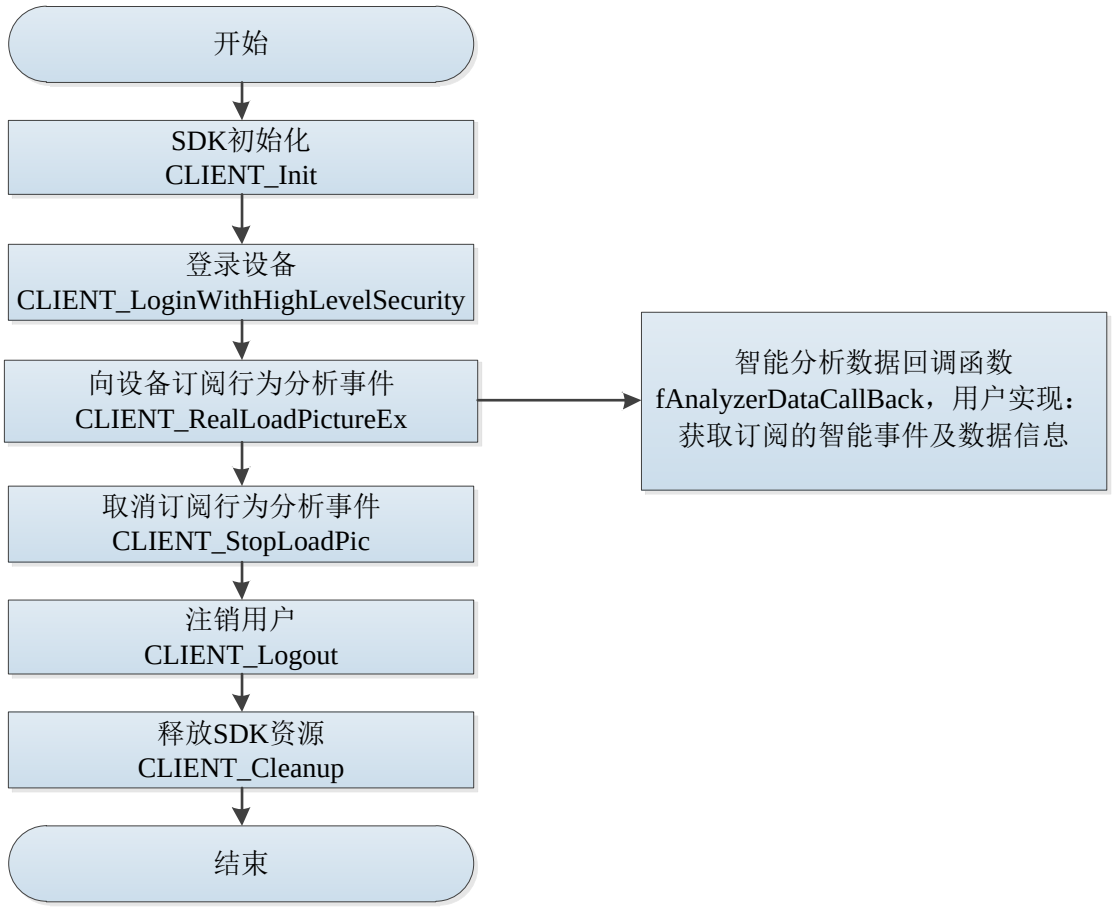
表2-8 智能 ATM 的接口信息

接口	说明
CLIENT_RealLoadPictureEx	开始智能事件分析数据订阅扩展接口
CLIENT_StopLoadPic	停止智能事件分析数据订阅接口
fAnalyzerDataCallBack	智能事件分析数据回调接口

2.6.3 流程说明

智能 ATM 中 SDK 行为分析智能事件订阅如图 2-8 所示。

图2-8 智能 ATM 智能事件订阅



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 开始智能事件订阅。
步骤4　调用 fAnalyzerDataCallBack 智能分析回调函数处理已订阅的智能事件和数据。
步骤5　调用 CLIENT_StopLoadPic 函数取消智能事件订阅。
步骤6　业务使用完后，调用 CLIENT_Logout 函数退出设备。
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

回调函数数据处理：不建议在 fAnalyzerDataCallBack 函数中进行耗时高的 I/O 或延时等操作，如本地保存图片，数据库插入等。Windows 下可使用 postMessage 将数据抛出去，开启线程处理；Linux 环境下同理。

2.6.4 示例代码

```
// 智能分析数据回调
// 不建议在该回调函数中调用 SDK 接口
// 通过 CLIENT_RealLoadPictureEx/CLIENT_RealLoadPicture 设置该回调函数，当设备端有智能图片事件
```

上报时，SDK 会调用该函数

// nSequence 表示上传的相同图片情况，为 0 时表示是第一次出现，为 2 表示最后一次出现或仅出现一次，为 1 表示此次之后还有

// int nState = *(int*) reserved 表示当前回调数据的状态，为 0 表示当前数据为实时数据，为 1 表示当前回调数据是离线数据，为 2 时表示离线数据传送结束

// 返回值已废除，无特殊意义

```
int  CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    int nAlarmChn = 0;
    IVS_CFG_ANALYSEVENTS_INFOEX ivsInfoEx = {0};
    ivsInfoEx.nEventsNum = 1;
    switch(dwAlarmType)
    {
        // 绊线/警戒线入侵（对应 DEV_EVENT_CROSSREGION_INFO）
        case EVENT_IVS_CROSSLINEDETECTION::
        {
            pCrossLine = (DEV_EVENT_CROSSLINE_INFO*)pAlarmInfo;
            ivsInfoEx.stuEventInfo[0].alarmAction = pCrossLine->bEventAction;
            ivsInfoEx.stuEventInfo[0].alarmType = dwAlarmType;
            strncpy(ivsInfoEx.stuEventInfo[0].szRuleName,
                pCrossLine->szName,
            (sizeof(pCrossLine->szName) >= sizeof(ivsInfoEx.stuEventInfo[0].szRuleName)
            ? sizeof(ivsInfoEx.stuEventInfo[0].szRuleName) : sizeof(pCrossLine->szName)));
            ivsInfoEx.stuEventInfo[0].nObjectNum = 1;
            memcpy(&ivsInfoEx.stuEventInfo[0].stuObject[0],
                &pCrossLine->stuObject,
                sizeof(pCrossLine->stuObject));
        }
        break;
        //警戒区事件（对应 DEV_EVENT_CROSSREGION_INFO）
        case EVENT_IVS_CROSSREGIONDETECTION
            {
                //.....
            }
```

```
                break;


            .....
            default:
                printf("other event type[%d]\n", dwAlarmType);
                break;
        }
        // 下载图片
        if (dwBufSize > 0 && NULL != pBuffer)
        {
            // 预防同一时间收到多张图片，只通过接收时间来保存图片可能会覆盖，于是通过 i 来标记
            static int i;
            char szPicturePath[256] = "";
            time_t stuTime;
            time(&stuTime);
            char szTmpTime[128] = "";
            strftime(szTmpTime, sizeof(szTmpTime) - 1, "%y%m%d_%H%M%S", gmtime(&stuTime));
            _snprintf(szPicturePath, sizeof(szPicturePath)-1, "%d_%s.jpg", ++i, szTmpTime);
            FILE* pFile = fopen(szPicturePath, "wb");
            if (NULL == pFile)
            {
                return 0;
            }
            int nWrite = 0;
            while(nWrite != dwBufSize)
            {
                nWrite += fwrite(pBuffer + nWrite, 1, dwBufSize - nWrite, pFile);
            }

            fclose(pFile);
        }
        return 1;
}
//智能事件订阅代码
```

```
{
    // 资源初始化
    ……
    // 登录设备
    ……
    // 订阅智能图片报警
    LDWORD dwUser = 0;
    int nChannel = 0;
    // 每次设置对应一个通道，并且对应一种类型的事件
    // 如果要设置该通道上传所有类型的事件，可以将参数 dwAlarmType 设置为 EVENT_IVS_ALL
    // 如果需要设置一个通道上传两种事件，那么请调用两次 CLIENT_RealLoadPictureEx，并且传入不同的事件类型
    lRealLoadHandle = CLIENT_RealLoadPictureEx(lLoginHandle, nChannel, EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, dwUser, NULL);
    //……
    // 停止订阅图片报警
    CLIENT_StopLoadPic(lRealLoadHandle))
    // 退出设备
    CLIENT_Logout(lLoginHandle))
    CLIENT_Cleanup();
}
```

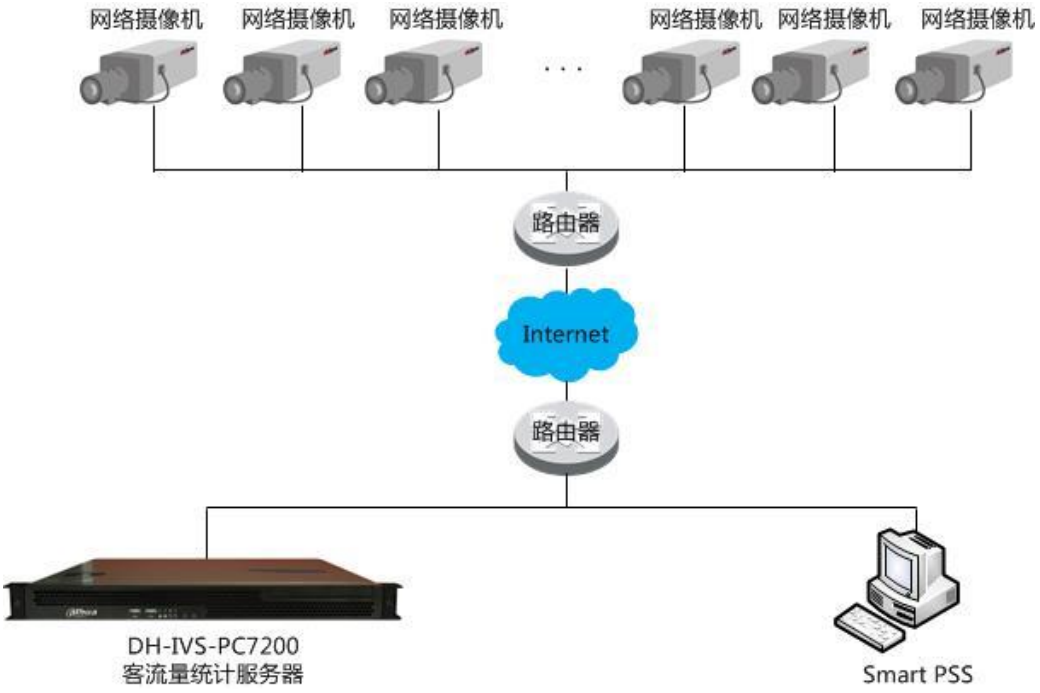
2.7 客流量统计

2.7.1 简介

产品介绍

客流量统计功能指在经营区域安装前端设备，智能分析服务器根据前端采集的视频数据精确统计出每个入口实时客流进出人数。该类产品被广泛应用于大型商业、旅游行业、公共安全、文博和连锁等行业。

图2-9 客流量统计组网图



产品型号

客流量统计功能主要应用于 DH-IVS-PC7200 设备。

SDK 接入功能

SDK 接入实现以下功能：

- 实时统计前端设备采集的客流量信息
- 查询智能分析设备统计的历史客流量信息。

说明

用户在订阅客流量统计服务前，需要在设备 Web 端配置客流量统计规则。

2.7.2 接口总览

表2-9 客流量统计的接口信息

接口	说明
CLIENT_AttachVideoStatSummary	开始订阅实时客流量统计
CLIENT_DetachVideoStatSummary	取消订阅实时客流量统计
fVideoStatSumCallBack	客流量信息回调函数
CLIENT_StartFindNumberStat	开始查询历史客流量信息
CLIENT_DoFindNumberStat	继续查询历史客流量信息
CLIENT_StopFindNumberStat	结束查询历史客流量信息

2.7.3 流程说明

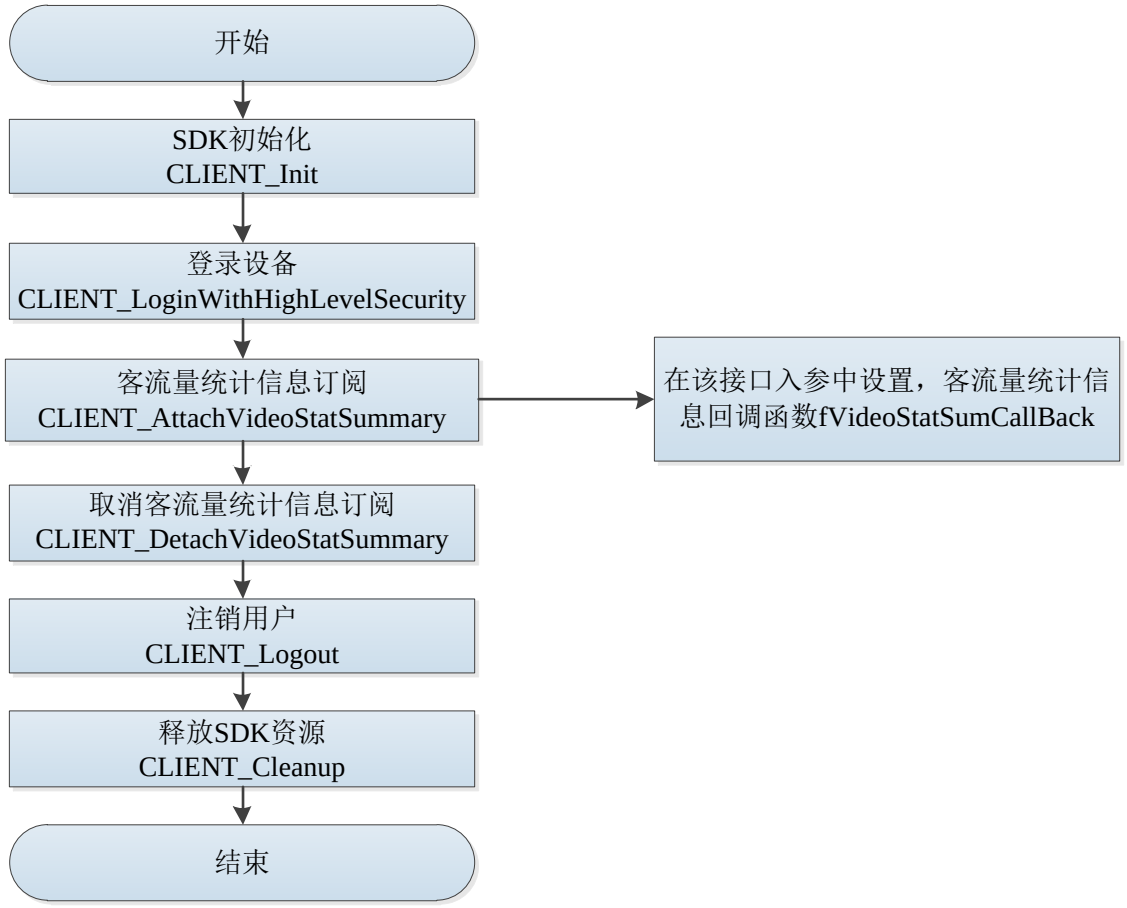
客流量统计主要应用于以下两种应用场景：

- 实时统计客流量信息
  即 SDK 向设备端订阅客流量统计服务后，设备将客流量信息实时上报到 SDK。
- 查询历史客流量信息
  即用户指定查询客流量信息的起始时间和结束时间，设备端返回该段时间内的客流量信息。

2.7.3.1 SDK 实时客流量统计

SDK 实时客流量统计流程如图 2-10 所示。

图2-10 实时客流量统计流程图



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　开始该业务调用 CLIENT_AttachVideoStatSummary 函数订阅客流量统计信息，在该接口入参中设置用户自己实现的客流量统计信息回调函数 fVideoStatSumCallBack，用户在该回调函数中获取客流量信息。

步骤4　停止该业务：调用 CLIENT_DetachVideoStatSummary 函数取消订阅客流量统计。

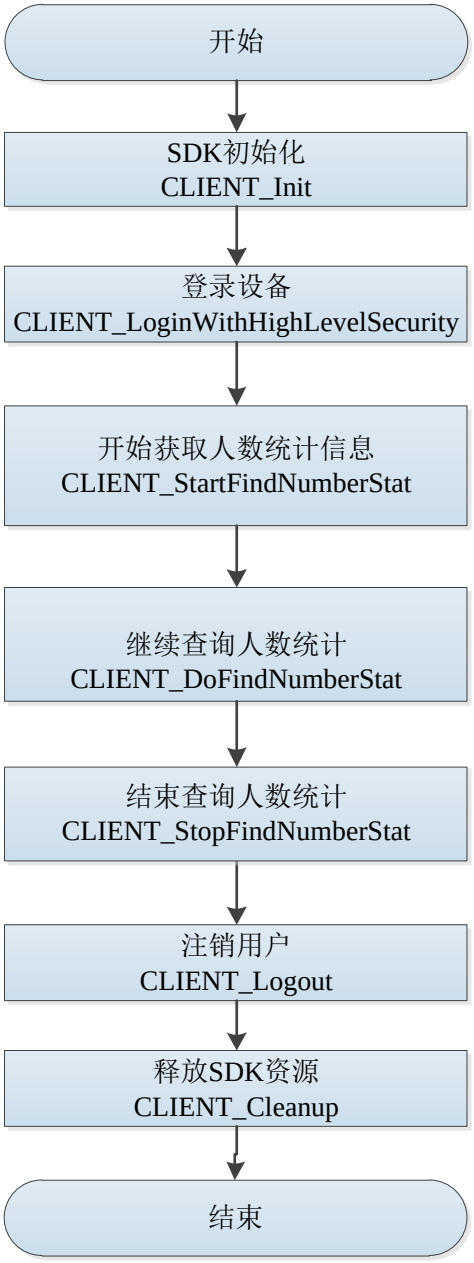
步骤5　业务使用完后，调用 CLIENT_Logout 函数退出设备。

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

2.7.3.2 查询历史客流量统计信息

查询历史客流量统计信息流程如图 2-11 所示。

图2-11 查询历史客流量统计信息



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 登录设备。
步骤3　调用 CLIENT_StartFindNumberStat 函数开始获取人数统计信息。
步骤4　调用 CLIENT_DoFindNumberStat 函数继续查询某段时间的客流量统计信息。
步骤5　调用 CLIENT_StopFindNumberStat 函数停止记录查询。
步骤6　业务使用完后，调用 CLIENT_Logout 函数退出设备。
步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

2.7.4 示例代码

2.7.4.1 实时上报客流量统计

```
//
#include "stdafx.h"
#include "dhnetsdk.h"
#pragma comment(lib,"dhnetsdk.lib")
//回调函数中根据需要处理收到的数据
void _stdcall VideoStatSumCallback(LLONG lAttachHandle, NET_VIDEOSTAT_SUMMARY* pBuf, DWORD dwBufLen, LDWORD dwUser)
{
	printf("Infomation:\n");
	printf("通道号：%d;\n 规则名称：%s;\n 进入小计：%d;\n 出去小计：%d\n",pBuf->nChannelID,pBuf->szRuleName,pBuf->stuEnteredSubtotal.nTotal,pBuf->stuExitedSubtotal.nTotal);
}
int main(int argc, char* argv[])
{
	// 初始化
	CLIENT_Init(NULL,0);
	  .....
	// 登录
	NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY stInparam;
	memset(&stInparam, 0, sizeof(stInparam));
	stInparam.dwSize = sizeof(stInparam);
	strncpy(stInparam.szIP, g_szDevIp, sizeof(stInparam.szIP) - 1);
	strncpy(stInparam.szPassword, g_szPasswd, sizeof(stInparam.szPassword) - 1);
	strncpy(stInparam.szUserName, g_szUserName, sizeof(stInparam.szUserName) - 1);
	stInparam.nPort = g_nPort;
	stInparam.emSpecCap = EM_LOGIN_SPEC_CAP_TCP;
	LLONG lLoginID = CLIENT_LoginWithHighLevelSecurity(&stInparam, &stOutparam);
	if (lLoginID)
	{
		NET_IN_ATTACH_VIDEOSTAT_SUM InParam={sizeof(NET_IN_ATTACH_VIDEOSTAT_SUM)};
		InParam.nChannel=0;
		InParam.cbVideoStatSum=VideoStatSumCallback;
		NET_OUT_ATTACH_VIDEOSTAT_SUM OutParam={0};
```

```
            OutParam.dwSize=sizeof(OutParam);
            int nWaitTime=5000;   //wait time
            LLONG attachHnd = 0;
// 订阅客流量统计
attachHnd = CLIENT_AttachVideoStatSummary(lLoginID,&InParam,&OutParam,nWaitTime)
                  if(attachHnd)
            {
                  printf("CLIENT_AttachVideoStatSummary sucess\n");
            }
            else
            {
                  printf("error number:%x",CLIENT_GetLastError());
            }
      }
      else
      {
            printf("login fail\n");
      }
.....
// 取消订阅客流量统计
CLIENT_DetachVideoStatSummary(attachHnd);
      // 退出设备
CLIENT_Logout(lLoginHandle));
      CLIENT_Cleanup();
      return 0;
}
```

2.7.4.2 查询历史客流量统计

```
// AttachVideoStatSummary.cpp : 定义控制台应用程序的入口点。
//
#include "stdafx.h"
#include "dhnetsdk.h"
#pragma comment(lib,"dhnetsdk.lib")
//回调函数中根据需要处理收到的数据
void _stdcall VideoStatSumCallback(LLONG lAttachHandle, NET_VIDEOSTAT_SUMMARY* pBuf, DWORD dwBufLen, LDWORD dwUser)
{
```

```
    printf("Infomation:\n");
    printf("通道号：%d;\n 规则名称：%s;\n 进入小计：%d;\n 出去小
计：%d\n",pBuf->nChannelID,pBuf->szRuleName,pBuf->stuEnteredSubtotal.nTotal,pBuf->stuExitedSubt
otal.nTotal);
}
int main(int argc, char* argv[])
{
    // 初始化
    CLIENT_Init(NULL,0);
     .....
    // 登录
    NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY stInparam;
    memset(&stInparam, 0, sizeof(stInparam));
    stInparam.dwSize = sizeof(stInparam);
    strncpy(stInparam.szIP, g_szDevIp, sizeof(stInparam.szIP) - 1);
    strncpy(stInparam.szPassword, g_szPasswd, sizeof(stInparam.szPassword) - 1);
    strncpy(stInparam.szUserName, g_szUserName, sizeof(stInparam.szUserName) - 1);
    stInparam.nPort = g_nPort;
    stInparam.emSpecCap = EM_LOGIN_SPEC_CAP_TCP;
    LLONG lLoginID = CLIENT_LoginWithHighLevelSecurity(&stInparam, &stOutparam);
    if (lLoginID == 0)
    {
            CLIENT_Cleanup();
        return 0;
    }
        NET_IN_FINDNUMBERSTAT inParam ;
    inParam.dwSize = (uint)Marshal.SizeOf(inParam);
    inParam.nChannelID = nChannelID; // 要进行查询的通道号
    // 查询起始时间, 结束时间设置暂时精确到小时
    . . . . . .
    inParam.nGranularityType = 1; // 查询粒度 0:分钟,1:小时,2:日,3:周,4:月,5:季,6:年
    inParam.nWaittime = 5000; // 等待接收数据的超时时间

    NET_OUT_FINDNUMBERSTAT outParam;
```

```
    outParam.dwSize = sizeof(outParam);
    LLONG findHnd = m_FindHandle = CLIENT_StartFindNumberStat(pLoginHandle, &inParam, &outParam);
    //
    if (findHand == 0)
    {
        printf("find number stat failed! \n");
        goto e_clear;;
    }
    NET_IN_DOFINDNUMBERSTAT inDoFind;
    inDoFind.dwSize = sizeof(inDoFind);
    inDoFind.nBeginNumber = 0; // 从 0 开始查询
    inDoFind.nCount = 10; // 每次查询 10 条
    inDoFind.nWaittime = 5000; //接口超时时间 5s

    NET_OUT_DOFINDNUMBERSTAT outStuDoFindNumStat = {sizeof(NET_OUT_DOFINDNUMBERSTAT)};
    outStuDoFindNumStat.pstuNumberStat = new DH_NUMBERSTAT[10];
    for (int i = 0; i < 10 ; i++)
    {
        outStuDoFindNumStat.pstuNumberStat[i].dwSize = sizeof(DH_NUMBERSTAT);
    }
    outStuDoFindNumStat.nBufferLen = 10 * sizeof(DH_NUMBERSTAT);
    int index = 0;
    do
    {
        if (CLIENT_DoFindNumberStat(findHand, &inDoFind, &outStuDoFindNumStat) > 0)
        {
            for (int i = 0; i < outStuDoFindNumStat.nCount; i++, index++)
            {
                // 查询结果
            }
            // 查询下一次
            inDoFind.nBeginNumber += inDoFind.nCount;// 从上一次结束地方开始查询
```

```
            }
            else
            {
                printf("find error: \n");
                break;
            }
        } while (inDoFind.nBeginNumber >= outParam.dwTotalCount);
            ........
        // 停止查询客流量
        CLINET_StopFindNumberStat(findHand);
e_clear:
        // 退出设备
CLIENT_Logout(lLoginID))
        CLIENT_Cleanup();
        return 0;
}
```

第 3 章 接口说明

3.1 SDK 初始化

3.1.1 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	

3.1.2 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

3.1.3 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
	[in]dwUser	断线重连回调函数的用户参数

选项	说明
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

3.1.4 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	

3.2 设备初始化

3.2.1 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	

3.2.2 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	

3.2.3 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	

3.2.4 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

选项	说明	
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	

3.2.5 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	

3.2.6 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

选项	说明	
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	

3.2.7 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	

3.3 设备登录

3.3.1 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
		[in] szIP	设备 IP
		[in] nPort	设备端口
		[in] szUserName	用户名

选项	说明		
		[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	成功返回登录句柄，失败返回 0。 登录成功之后对设备的操作都可以通过登录句柄配合 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

3.3.2 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	

3.4 实时预览

3.4.1 打开预览 CLIENT_RealPlayEx

表3-15 打开预览 CLIENT_RealPlayEx

选项	说明	
描述	打开实时预览	
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	实时预览-主码流，等同于 DH_RType_Realplay
DH_RType_Realplay_1	实时预览-从码流 1
DH_RType_Realplay_2	实时预览-从码流 2
DH_RType_Realplay_3	实时预览-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

3.4.2 关闭预览 CLIENT_StopRealPlayEx

表3-17 关闭预览 CLIENT_StopRealPlayEx

选项	说明	
描述	关闭实时预览	
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	

3.4.3 保存预览数据 CLIENT_SaveRealData

表3-18 保存预览数据 CLIENT_SaveRealData

选项	说明	
描述	保存实时预览数据为文件	
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	

3.4.4 停止保存预览数据 CLIENT_StopSaveRealData

表3-19 停止保存预览数据 CLIENT_StopSaveRealData

选项	说明	
描述	停止保存实时预览数据为文件	
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	

3.4.5 设置预览数据回调 CLIENT_SetRealDataCallBackEx2

表3-20 设置预览数据回调 CLIENT_SetRealDataCallBackEx2

选项	说明	
描述	设置实时预览数据回调	
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	预览数据流回调函数
	[in] dwUser	预览数据流回调函数的参数
	[in] dwFlag	回调中预览数据的类型，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

3.5 智能事件订阅

3.5.1 开始智能事件订阅 CLIENT_RealLoadPictureEx

表3-22 开始智能事件订阅 CLIENT_RealLoadPictureEx

选项	说明	
描述	开始智能事件订阅	
函数	LLONG CALL_METHOD CLIENT_RealLoadPictureEx(LLONG lLoginID, int nChannelID, DWORD dwAlarmType, BOOL bNeedPicFile, fAnalyzer DataCallBack cbAnalyzerData, LDWORD dwUser, void* Reserved);	
参数	[in] lLoginID	登录句柄

选项	说明	
	[in] nChannelID	设备通道号
	[in] dwAlarmType	订阅报警事件类型
	[in]bNeedPicFile	是否订阅图片文件
	[in]cbAnalyzerData	智能事件回调函数
	[in] dwUser	用户自定义数据类型
	[in] Reserved	保留字段
返回值	● 成功返回 LLONG 类型订阅句柄 ● 失败返回 0	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

智能报警事件类型说明请参见表 3-23。

表3-23 智能事件类型说明

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_CROSSFENCEDETECTION	0x0000011F	穿越围栏	DEV_EVENT_CROSSFENCEDETECTION_INFO
EVENT_IVS_CROSSLINEDETECTION	0x00000002	绊线入侵	DEV_EVENT_CROSSLINE_INFO
EVENT_IVS_CROSSREGIONDETECTION	0x00000003	区域入侵	DEV_EVENT_CROSSREGION_INFO
EVENT_IVS_LEFTDETECTION	0x00000005	物品遗留	DEV_EVENT_LEFT_INFO
EVENT_IVS_PRESERVATION	0x00000008	物品保全	DEV_EVENT_PRESERVATION_INFO
EVENT_IVS_TAKENAWAYDETECTION	0x00000115	物品搬移	DEV_EVENT_TAKENAWAYDETECTION_INFO
EVENT_IVS_WANDERDETECTION	0x00000007	徘徊事件	DEV_EVENT_WANDER_INFO
EVENT_IVS_VIDEOABNORMALDETECTION	0x00000013	视频异常	DEV_EVENT_VIDEOABNORMALDETECTION_INFO
EVENT_IVS_AUDIO_ABNORMALDETECTION	0x00000126	声音异常	DEV_EVENT_IVS_AUDIO_ABNORMALDETECTION_INFO
EVENT_IVS_CLIMBDETECTION	0x00000128	攀高检测	DEV_EVENT_IVS_CLIMB_INFO
EVENT_IVS_FIGHTDETECTION	0x0000000E	斗殴检测	DEV_EVENT_FLOWSTAT_INFO
EVENT_IVS_LEAVEDETECTION	0x00000129	离岗检测	DEV_EVENT_IVS_LEAVE_INFO
EVENT_IVS_PRISONERRISEDETECTION	0x0000011E	起身检测	DEV_EVENT_PRISONERRISEDETECTION_INFO
EVENT_IVS_PASTEDETECTION	0x00000004	非法黏贴物贴条检测	DEV_EVENT_PASTE_INFO

3.5.2 停止智能事件订阅 CLIENT_StopLoadPic

表3-24 停止智能事件订阅 CLIENT_StopLoadPic

选项	说明	
描述	停止智能事件订阅	
函数	BOOL CALL_METHOD CLIENT_StopRealPlayEx(LLONG lRealHandle);	
参数	[in] lRealHandle	智能事件订阅句柄
返回值	BOOL 类型 ● 成功：TRUE ● 失败：FALSE	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

3.6 客流量统计

3.6.1 开始订阅客流量统计 CLIENT_AttachVideoStatSummary

表3-25 开始订阅客流量统计 CLIENT_AttachVideoStatSummary

选项	说明	
描述	开始订阅客流量统计	
函数	LLONG CALL_METHOD CLIENT_AttachVideoStatSummary(LLONG lLoginID, Const NET_IN_ATTACH_VIDEOSTAT_SUM* pInParam, NET_OUT_ATTACH_VIDEOSTAT_SUM* pOutParam, int nWaitTime);	
参数	[in] lLoginID	登录句柄
	[in]pInParam	客流量统计输入参数
	[out] pOutParam	客流量统计输出参数
	[in]nWaitTime	超时时间
返回值	LLONG 类型 ● 成功：不等于 0 ● 失败：等于 0	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

3.6.2 停止订阅客流量统计 CLIENT_DetachVideoStatSummary

表3-26 停止订阅客流量统计 CLIENT_DetachVideoStatSummary

选项	说明
描述	停止订阅客流量统计

选项	说明	
函数	BOOL CALL_METHOD CLIENT_DetachVideoStatSummary(LLONG lAttachHandle);	
参数	[in] lAttachHandle	订阅句柄
返回值	BOOL 类型 ● 成功：TRUE ● 失败：FALSE	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

3.6.3 开始查询客流量统计信息 CLIENT_StartFindNumberStat

表3-27 开始查询客流量统计信息 CLIENT_StartFindNumberStat

选项	说明	
描述	开始查询客流量统计信息	
函数	LLONG CALL_METHOD CLIENT_StartFindNumberStat(LLONG lLoginID, NET_IN_FINDNUMBERSTAT* pstInParam, NET_OUT_FINDNUMBERSTAT* pstOutParam);	
参数	[in] lLoginID	登录句柄
	[in] pstInParam	入参
	[out] pstOutParam	出参
返回值	LLONG 类型; ● 成功：不等于 0 ● 失败：等于 0	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

3.6.4 继续查询客流量统计信息 CLIENT_DoFindNumberStat

表3-28 继续查询客流量统计信息 CLIENT_DoFindNumberStat

选项	说明	
描述	继续查询客流量统计信息	
函数	int CALL_METHOD CLIENT_DoFindNumberStat(LLONG lFindHandle, NET_IN_DOFINDNUMBERSTAT* pstInParam, NET_OUT_DOFINDNUMBERSTAT* pstOutParam);	
参数	[in] lFindHandle	登录句柄
	[in]pstInParam	入参
	[out]pstOutParam	出参
返回值	int 类型 ● 成功：1 ● 失败：-1	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

3.6.5 停止查询客流量统计信息 CLIENT_StopFindNumberStat

表3-29 停止查询客流量统计信息 CLIENT_StopFindNumberStat

选项	说明	
描述	停止查询客流量统计信息	
函数	BOOL CALL_METHOD CLIENT_StopFindNumberStat(LLONG lFindHandle);	
参数	[in] lFindHandle	登录句柄
返回值	BOOL 类型 ● 成功:TRUE ● 失败：FALSE	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

第 4 章 回调函数

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

表4-5 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 实时预览数据回调函数 fRealDataCallBackEx2

表4-6 实时预览数据回调函数 fRealDataCallBackEx2

选项	说明
描述	实时预览数据回调函数

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	预览数据块地址
	[out] dwBufSize	预览数据块的长度，单位：字节
	[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 智能事件回调 fAnalyzerDataCallBack

表4-7 智能事件回调 fAnalyzerDataCallBack

选项	说明	
描述	智能事件回调	
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

选项	说明	
	[out]dwAlarmType	智能事件类型
	[out]pAlarmInfo	事件信息缓存
	[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	

4.7 客流量统计信息回调 fVideoStatSumCallBack

表4-8 客流量统计信息回调 fVideoStatSumCallBack

选项	说明	
描述	客流量统计信息回调	
函数	Typedef void (CALLBACK *fVideoStatSumCallBack) (LLONG lAttachHandle, NET_VIDEOSTAT_SUMMARY* pBuf, DWORD dwBufLen, LDWORD dwUser);	
参数	[in] lAttachHandle	订阅句柄
	[out] pBuf	回调数据
	[out] dwBufLen	回调数据长度
	[out] dwUser	用户自定义数据
返回值	Void	
说明	无	

附录1 法律声明

商标声明

- HDMI™ HIGH-DEFINITION MULTIMEDIA INTERFACE：本声明适用所有产品。如本产品使用 HDMI 技术，词语 HDMI、HDMI High-Definition Multimedia Interface（高清晰度多媒体接口）、HDMI 商业外观和 HDMI 徽标均为 HDMI Licensing Administrator, Inc.的商标或注册商标。本产品已经获得 HDMI Licensing Administrator, Inc.授权使用 HDMI 技术。
- 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

- 本文档供多个型号产品使用，产品外观和功能请以实物为准。
- 如果不按照本文档中的指导进行操作而造成的任何损失由使用方自己承担。
- 本文档会实时根据相关地区的法律法规更新内容，具体请参见产品的纸质、电子光盘、二维码或官网，如果纸质与电子档内容不一致，请以电子档为准。
- 本公司保留随时修改本文档中任何信息的权利，修改的内容将会在本文档的新版本中加入，恕不另行通知。
- 本文档可能包含技术上不准确的地方、或与产品功能及操作不相符的地方、或印刷错误，以公司最终解释为准。
- 如果获取到的 PDF 文档无法打开，请使用最新版本或最主流的阅读工具。

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。
   - 不包含账户名称或账户名称的倒序。
   - 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（如 NVR、DVR 和 IP 摄像机等）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. **开启账户锁定**

   出厂默认开启账户锁定功能，建议您保持开启状态，以保护账户安全。在攻击者多次密码尝试失败后，其对应账户及源 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. **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。

8. **合理分配账户及权限**

   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。

9. **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。

10. 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。

11. 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。

12. 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。

13. 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。
- 开启设备 IP/MAC 地址过滤功能，限制允许访问设备的主机范围。