NetSDK 编程指导手册

（存储分册）



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 NVR、EVS、HDCVI 等存储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、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
V2.0.0	基础核心内容参考《NetSDK 编程指导手册》	2025.02
V1.0.5	全文优化语言。	2023.02
V1.0.4	● 新增 NVR6 系等设备登录业务注意事项。 ● 修改全文英文字体为 Myriad Pro。 ● 删除 fisheye 鱼眼矫正库。	2021.05
V1.0.3	● 删除 avnetsdk 依赖库信息。 ● 新增 StreamConvertor 依赖库。	2021.03
V1.0.2	修改登录设备和搜索设备接口函数等。	2020.03
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
NVR	Network Video Recorder 的缩写，即网络硬盘录像机。
EVS	Embedded Video Server 的缩写，即嵌入式视频服务器。
HDCVI	High Definition Composite Video Interface 的缩写，即高清视频复合接口。
主码流	视频码流类型的一种，一般是分辨率比较高，清晰度、画质更好的码流，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。
分辨率	分辨率包括显示分辨率和图像分辨率。显示分辨率指单位面积显示像素的数量；图像分辨率指图像中存储的信息量，指每英寸图像内有多少个像素点。
帧率	用来显示视频帧数的度量，常用单位 FPS 和 Hz，帧数越高视频的流畅度越高。一般帧率高于 24FPS 的会让人感觉画面是连贯的。
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，通道号从 0 开始。若 SDK 与单个前端设备直连，则视频通道号一般为 0。
动态检测报警	检测到画面上有移动物体时，会有动态检测报警上报。
硬盘故障报警	硬盘故障时，会有硬盘故障报警上报。
视频丢失报警	此报警只针对模拟通道，当某个模拟通道没有录像了，会上报此报警。数字通道判断视频丢失，一般使用下面的 IPC 断网报警。
坏硬盘报警	硬盘有损坏时，会有此报警上报。
IPC 断网报警	IPC 断网时，会有 IPC 断网报警上报。
外部报警	即 NVR 本地报警。NVR 报警端子连接报警器时，会有外部报警上报。
IPC 外部报警	IPC 上报警线短接报警器时，会有 IPC 外部报警上报。

目录

前言 II
名词解释 III
第 1 章 产品概述 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 9
2.3.1 简介 9
2.3.2 接口总览 10
2.3.3 流程说明 10
2.3.4 示例代码 11
2.4 实时预览 12
2.4.1 简介 12
2.4.2 接口总览 12
2.4.3 流程说明 12
2.4.4 示例代码 16
2.5 录像回放 17
2.5.1 简介 17
2.5.2 接口总览 17
2.5.3 流程说明 17
2.5.4 示例代码 19
2.6 录像下载 20
2.6.1 简介 20
2.6.2 接口总览 21
2.6.3 流程说明 21
2.6.4 示例代码 24
2.7 云台控制 31
2.7.1 简介 31
2.7.2 接口总览 31
2.7.3 流程说明 31
2.7.4 示例代码 33
2.8 语音对讲 34

2.8.1 简介 34
2.8.2 接口总览 34
2.8.3 流程说明 34
2.8.4 示例代码 36
2.9 视频抓图 37
2.9.1 简介 37
2.9.2 接口总览 37
2.9.3 流程说明 37
2.9.4 示例代码 39
2.10 报警上报 40
2.10.1 简介 40
2.10.2 接口总览 40
2.10.3 流程说明 40
2.10.4 示例代码 41
2.11 存储 43
2.11.1 简介 43
2.11.2 接口总览 43
2.11.3 流程说明 43
2.11.4 示例代码 47
第 3 章 接口函数 56
3.1 SDK 初始化 56
3.1.1 SDK 初始化 CLIENT_Init 56
3.1.2 SDK 清理 CLIENT_Cleanup 56
3.1.3 设置断线重连回调函数 CLIENT_SetAutoReconnect 56
3.1.4 设置网络参数 CLIENT_SetNetworkParam 57
3.2 设备初始化 57
3.2.1 搜索设备 CLIENT_StartSearchDevicesEx 57
3.2.2 设备初始化 CLIENT_InitDevAccount 57
3.2.3 获取密码重置信息 CLIENT_GetDescriptionForResetPwd 58
3.2.4 检验安全码是否有效 CLIENT_CheckAuthCode 58
3.2.5 重置密码 CLIENT_ResetPwd 59
3.2.6 获取密码规则 CLIENT_GetPwdSpecification 59
3.2.7 停止搜索设备 CLIENT_StopSearchDevices 60
3.3 设备登录 60
3.3.1 高安全级别登录 CLIENT_LoginWithHighLevelSecurity 60
3.3.2 用户登出设备 CLIENT_Logout 61
3.4 实时预览 62
3.4.1 打开预览 CLIENT_RealPlayEx 62
3.4.2 关闭预览 CLIENT_StopRealPlayEx 63
3.4.3 保存预览数据 CLIENT_SaveRealData 63
3.4.4 停止保存预览数据 CLIENT_StopSaveRealData 63
3.4.5 设置预览数据回调 CLIENT_SetRealDataCallBackEx2 64
3.5 回放 64
3.5.1 按时间方式回放 CLIENT_PlayBackByTimeEx2 64
3.5.2 设置工作模式 CLIENT_SetDeviceMode 65
3.5.3 停止录像回放 CLIENT_StopPlayBack 65
3.5.4 获取回放 OSD 时间 CLIENT_GetPlayBackOsdTime 66

3.5.5 暂停或恢复录像回放 CLIENT_PausePlayBack 66
3.6 录像下载 67
3.6.1 查询时间段内的所有录像文件 CLIENT_QueryRecordFile 67
3.6.2 打开录像查询句柄 CLIENT_FindFile 68
3.6.3 查找录像文件 CLIENT_FindNextFile 68
3.6.4 关闭录像查询句柄 CLIENT_FindClose 69
3.6.5 按文件下载录像 CLIENT_DownloadByRecordFileEx 69
3.6.6 按时间下载录像 CLIENT_DownloadByTimeEx 70
3.6.7 查询录像下载进度 CLIENT_GetDownloadPos 71
3.6.8 停止录像下载 CLIENT_StopDownload 71
3.7 云台控制 72
3.7.1 云台控制 CLIENT_DHPTZControlEx2 72
3.8 语音对讲 75
3.8.1 开启对讲 CLIENT_StartTalkEx 75
3.8.2 关闭对讲 CLIENT_StopTalkEx 75
3.8.3 关闭录音 CLIENT_RecordStopEx 75
3.8.4 开启录音 CLIENT_RecordStartEx 76
3.8.5 发送语音 CLIENT_TalkSendData 76
3.8.6 解码语音 CLIENT_AudioDecEx 76
3.9 视频抓图 77
3.9.1 设备抓图 CLIENT_SnapPictureToFile 77
3.9.2 抓图 CLIENT_CapturePictureEx 77
3.10 报警上报 78
3.10.1 设置报警回调函数 CLIENT_SetDVRMessCallBack 78
3.10.2 订阅报警 CLIENT_StartListenEx 78
3.10.3 停止订阅报警 CLIENT_StopListen 79
3.11 存储 79
3.11.1 直接获取远程设备连接状态 CLIENT_QueryDevState 79
3.11.2 查询设备信息 CLIENT_QueryDevInfo 80
3.11.3 订阅远程设备状态 CLIENT_AttachCameraState 80
3.11.4 停止订阅远程设备状态 CLIENT_DetachCameraState 81
3.11.5 获取远程设备信息 CLIENT_MatrixGetCameras 81
3.11.6 获取通道名称 CLIENT_QueryChannelName 81
3.11.7 获取通道名称 CLIENT_GetNewDevConfig 82
3.11.8 解析数据 CLIENT_ParseData 83
第 4 章 回调函数 84
4.1 搜索设备回调函数 fSearchDevicesCB 84
4.2 异步搜索设备回调函数 fSearchDevicesCBEx 84
4.3 断线回调函数 fDisConnect 84
4.4 断线重连回调函数 fHaveReConnect 85
4.5 实时预览数据回调函数 fRealDataCallBackEx2 85
4.6 音频数据回调函数 pfAudioDataCallBack 86
4.7 回放及按文件下载进度回调函数 fDownLoadPosCallBack 87
4.8 回放及下载数据回调函数 fDataCallBack 87
4.9 按时间下载回调函数 fTimeDownLoadPosCallBack 88
4.10 报警回调函数 fMessCallBack 88
4.11 远程设备状态回调函数 fCameraStateCallBack 89

附录 1 法律声明 91
附录 2 网络安全建议 92

第 1 章 产品概述

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、录像回放、录像下载、语音对讲、视频抓图、报警上报和存储。

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

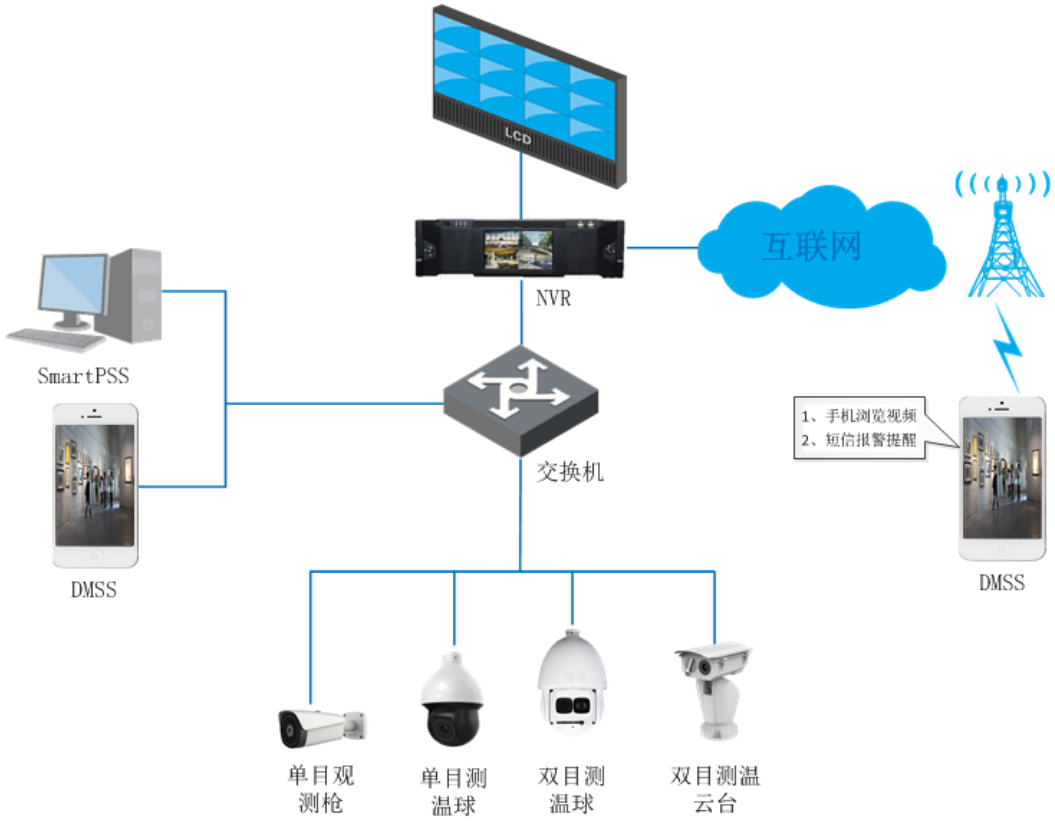
    Red Hat/SUSE 等通用 Linux 系统。
- 适用的设备如下：

  EVS5016/5024/5036/5048 系列、EVS7024/7036/7048/7064/7072 系列、MCS7024、NVR4832-4KS2、NVR4832、NVR3XX、NVR5XX 和 NVR724-256 等。

1.3 应用场景

可作为存储设备（下图以 NVR 为例）的视频输入通道，用户可以通过 SDK 访问 NVR 的对应通道获取网络摄像机的预览；基于预览，存储设备将视频存储，可以供后续的回放下载使用。

图1-1 应用场景



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

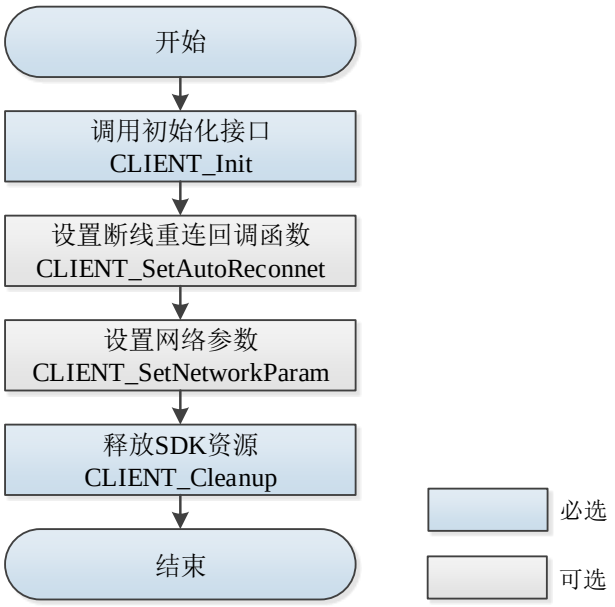
表2-1 SDK 初始化接口信息

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

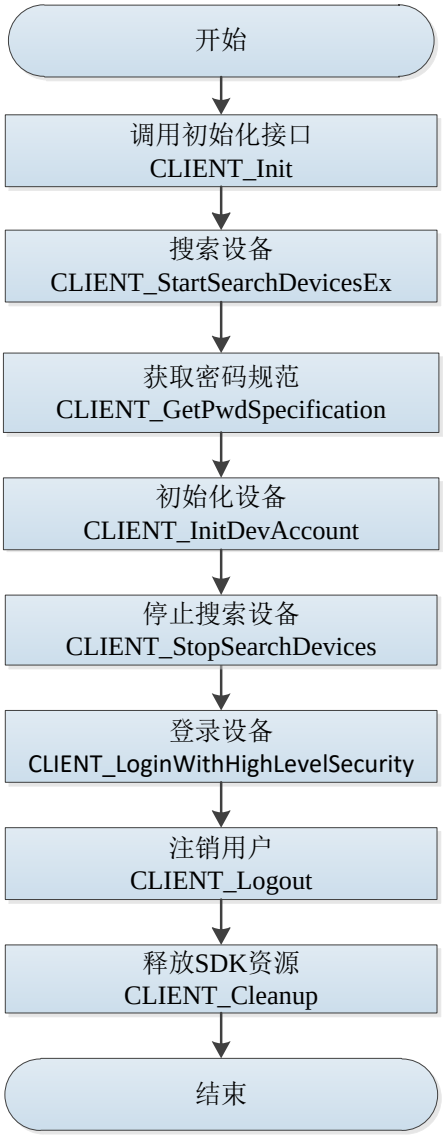
接口	说明
CLIENT_StartSearchDevicesEx	搜索局域网内的设备，找到未初始化设备
CLIENT_InitDevAccount	设备初始化接口
CLIENT_GetDescriptionForResetPwd	获取密码重置信息：手机号、邮箱和二维码信息
CLIENT_CheckAuthCode	校验安全码是否有效
CLIENT_ResetPwd	重置密码
CLIENT_GetPwdSpecification	获取密码规则
CLIENT_StopSearchDevices	停止搜索设备

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。

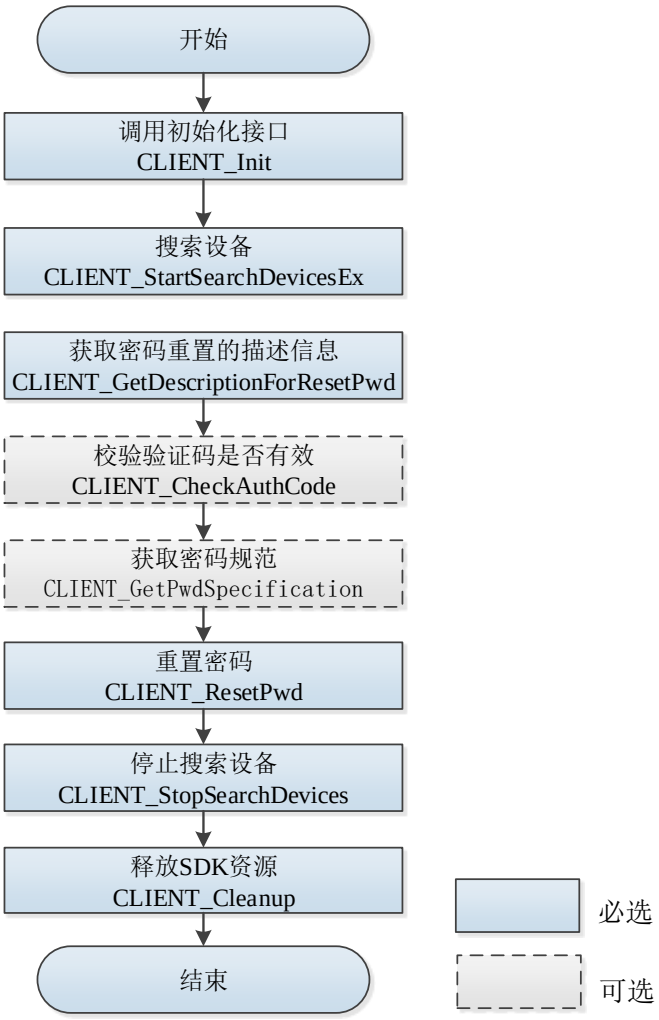
注意事项

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_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
```

2.3 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

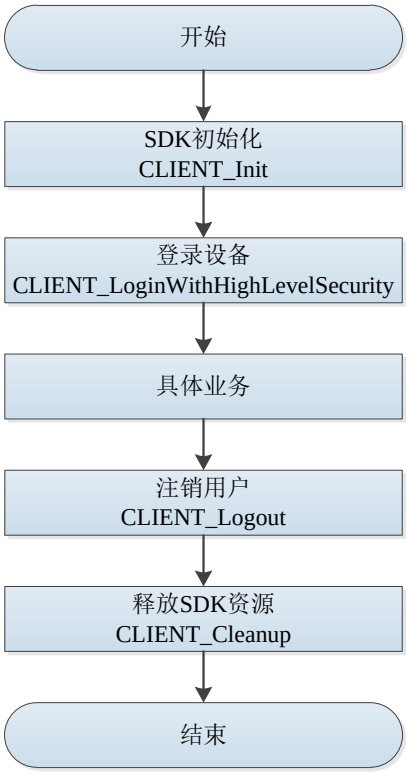
表2-3 设备登录接口信息

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。
- NVR6 系等设备（支持 16 个及以上硬盘），由于硬盘个数较多，会导致登录业务耗时较长。在进行设备登录业务前，推荐使用 CLIENT_SetOptimizeMode 接口优化获取硬盘信息。登录接口返回的硬盘个数参数，在设置上述优化后将无效，可以通过 CLIENT_QueryDevState（DH_DEVSTATE_DISK）接口获取。优化获取硬盘信息，示例代码如下：

```
int opt = OPTTYPE_MOBILE_DISK_INFO;
CLIENT_SetOptimizeMode(EM_OPT_TYPE_MOBILE_OPTION, &opt);
```

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_DEVICEINFO_Ex stDevInfo = {0};
int nError = 0;
// 登录设备
NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY stInparam;
memset(&stInparam, 0, sizeof(stInparam));
stInparam.dwSize = sizeof(stInparam);
```

```
strncpy(stInparam.szIP, csIp.GetBuffer(0), sizeof(stInparam.szIP) - 1);
strncpy(stInparam.szPassword, csPwd.GetBuffer(0), sizeof(stInparam.szPassword) - 1);
strncpy(stInparam.szUserName, csName.GetBuffer(0), sizeof(stInparam.szUserName) - 1);
stInparam.nPort = sPort;
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);
// 退出设备
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

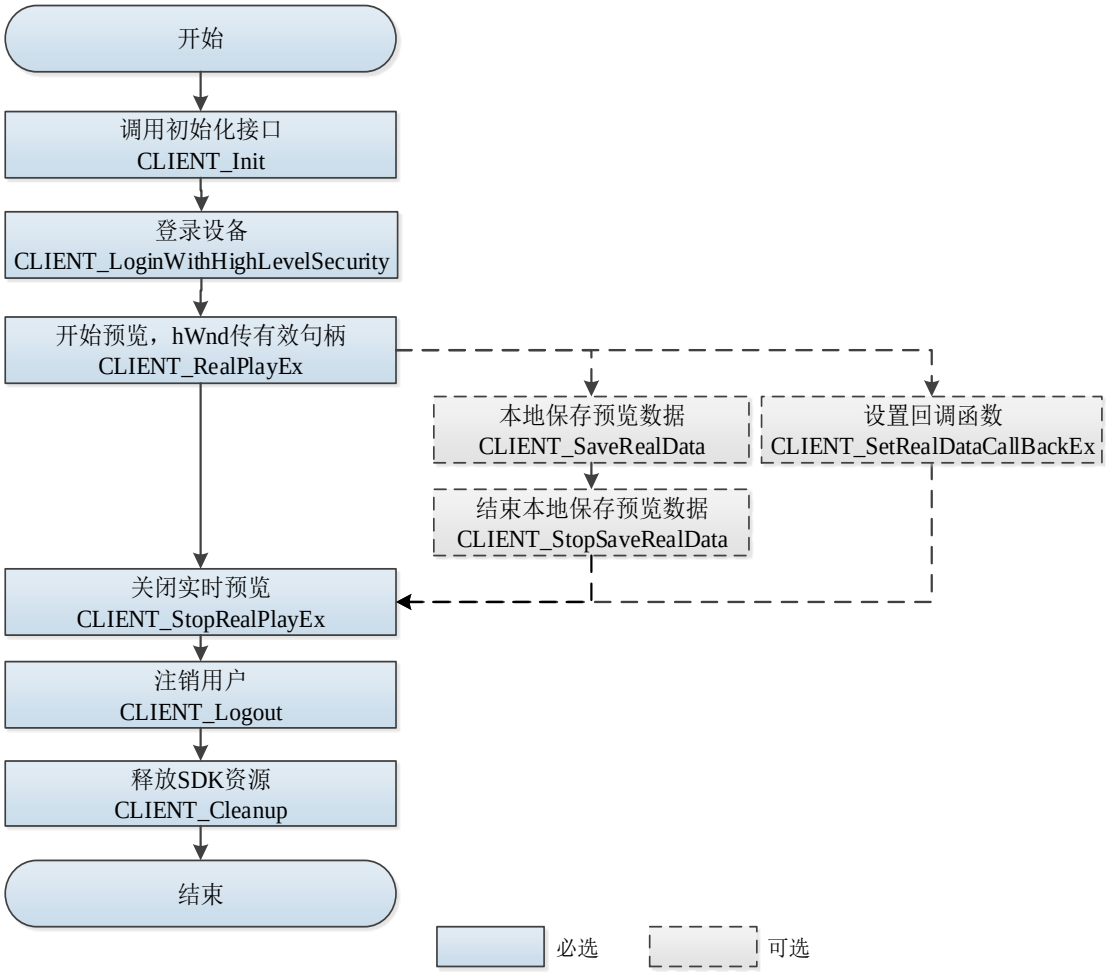
2.4.3 流程说明

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；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：

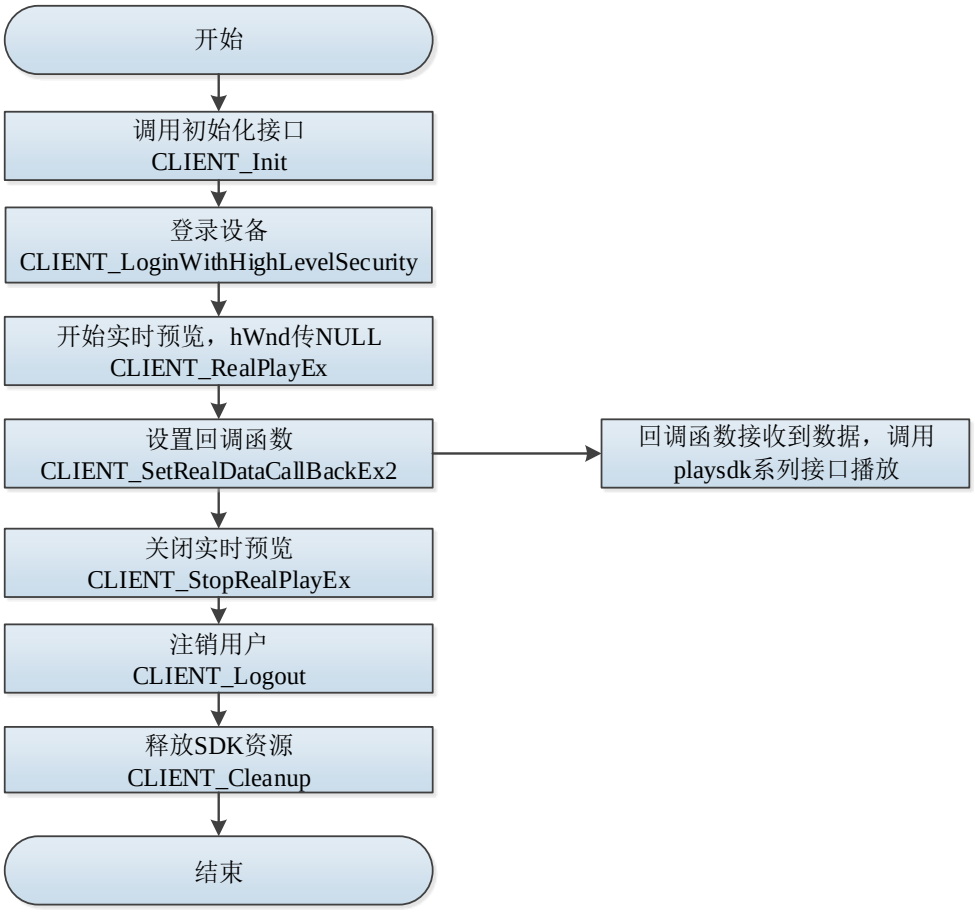
```
NET_PARAM stuNetParam = {0};
stuNetParam. nGetConnInfoTime = 5000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- 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 录像回放

2.5.1 简介

录像回放是有针对地对系统中的一些通道进行特定时间段的视频回放操作，以实现在海量的视频信息中找到目标视频进而进行调查。

回放功能主要包括：开始回放、暂停回放、恢复回放和停止回放。

2.5.2 接口总览

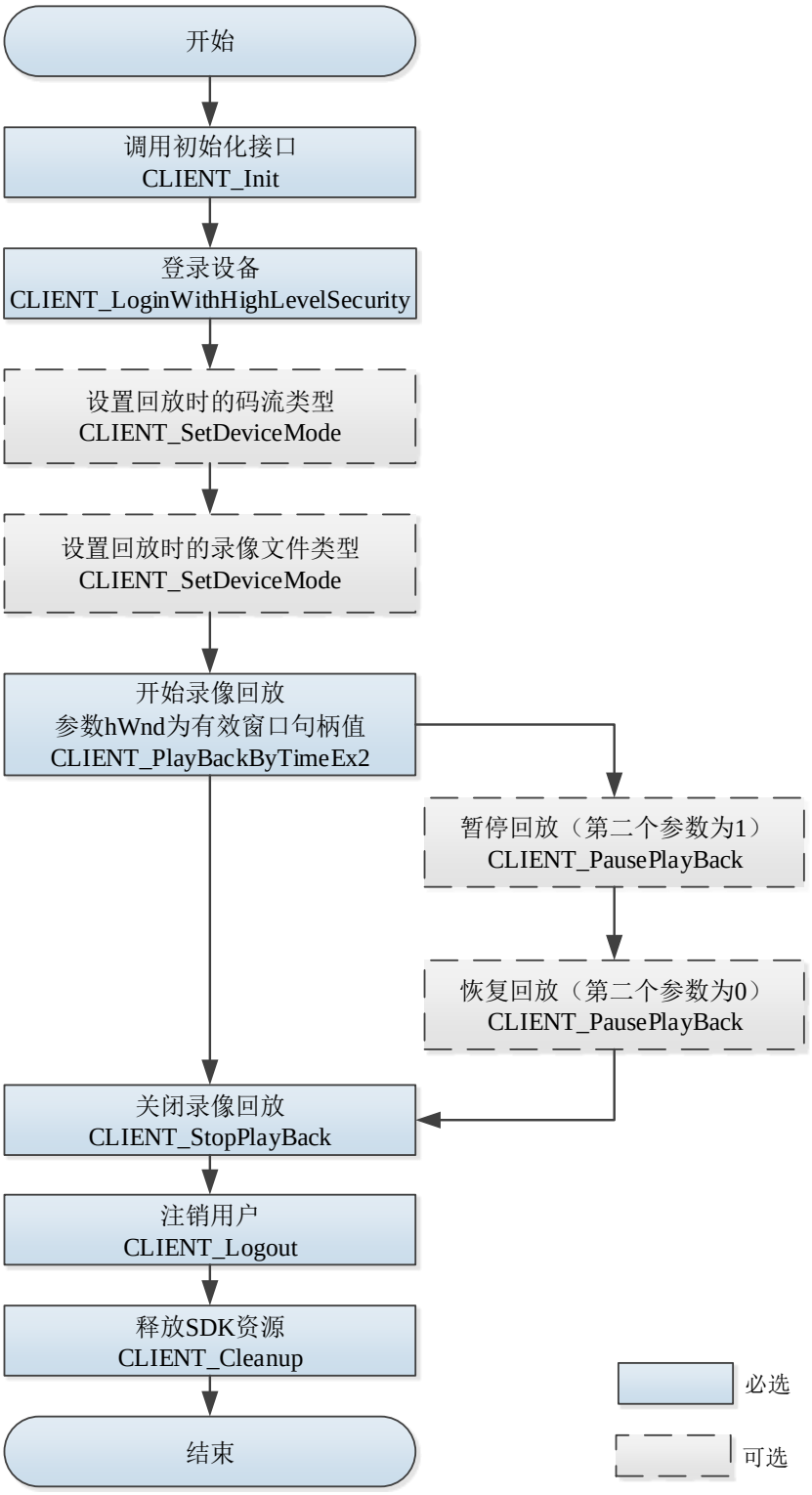
表2-6 录像回放的接口信息

接口	说明
CLIENT_PlayBackByTimeEx2	按时间方式回放扩展接口
CLIENT_SetDeviceMode	设置设备语音对讲、回放、权限等工作模式接口
CLIENT_StopPlayBack	停止录像回放接口
CLIENT_GetPlayBackOsdTime	获取回放 OSD 时间接口
CLIENT_PausePlayBack	暂停或恢复录像回放

2.5.3 流程说明

SDK 初始化之后，用户只需输入通道号、录像的起始时间、结束时间和有效窗口句柄，即可实现所需录像段录像回放。接口调用流程如图 2-7 所示。

图2-7 SDK 解码回放流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

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

步骤3　（可选）调用两次 CLIENT_SetDeviceMode，设置回放时的码流类型参数 **emType** 为 **DH_RECORD_STREAM_TYPE**，回放时的录像文件类型参数 **emType** 为 **DH_RECORD_TYPE**。

步骤4　调用 CLIENT_PlayBackByTimeEx2 启动回放，hWnd 参数为有效窗口句柄值。
步骤5　（可选）调用 CLIENT_PausePlayBack，第二个参数为 1 时暂停回放。
步骤6　（可选）调用 CLIENT_PausePlayBack，第二个参数为 0 时恢复回放。
步骤7　回放使用完后，调用 CLIENT_StopPlayBack 停止回放。
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。

2.5.4 示例代码

2.5.4.1 回放和停止回放

```
// 设置回放时的码流类型，此处设置成主码流
int nStreamType = 0; // 0-主辅码流,1-主码流,2-辅码流
CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_STREAM_TYPE, &nStreamType);

// 设置回放时的录像文件类型，此处设置成所有录像
NET_RECORD_TYPE emFileType = NET_RECORD_TYPE_ALL; // 所有录像
CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_TYPE, &emFileType);

//开启录像回放
int nChannelID = 0; // 通道号
NET_TIME stuStartTime = {0}; // 录像开始时间
stuStartTime.dwYear = 2015;
stuStartTime.dwMonth = 11;
stuStartTime.dwDay = 20;

NET_TIME stuStopTime = {0}; // 录像结束时间
stuStopTime.dwYear = 2015;
stuStopTime.dwMonth = 11;
stuStopTime.dwDay = 21;

NET_IN_PLAY_BACK_BY_TIME_INFO stIn = {0};
NET_OUT_PLAY_BACK_BY_TIME_INFO stOut = {0};
memcpy(&stIn.stStartTime, &stuStartTime, sizeof(stuStartTime));
memcpy(&stIn.stStopTime, &stuStopTime, sizeof(stuStopTime));
stIn.hWnd = hWnd;
stIn.fDownLoadDataCallBack = DataCallBack;
stIn.dwDataUser = NULL;
stIn.cbDownLoadPos = NULL;
stIn.dwPosUser = NULL;
stIn.nPlayDirection = emDirection;
stIn.nWaittime = 10000;

LLONG lPlayHandle = CLIENT_PlayBackByTimeEx2(lLoginHandle, nChannelID, &stIn, &stOut);
if (0 == lPlayHandle)
```

```
{
    printf("CLIENT_PlayBackByTimeEx2: failed! Error code: %x.\n", CLIENT_GetLastError());
}
printf("input any key to quit!\n");
getchar();

// 停止回放
if (0 != lPlayHandle)
{
    if (FALSE == CLIENT_StopPlayBack(lPlayHandle))
    {
        printf("CLIENT_StopPlayBack Failed, lRealHandle[%x]!Last Error[%x]\n" , lPlayHandle,
CLIENT_GetLastError());
    }
    else
    {
        lPlayHandle = 0;
    }
}
```

2.5.4.2 暂停回放和恢复暂停（可选）

```
// 暂停回放
BOOL bSuccess = CLIENT_PausePlayBack(m_hPlayBack,TRUE);
if (!bSuccess)
{
    printf("CLIENT_PausePlayBack Failed, lPlayHandle[%x]!Last Error[%x]\n" , lPlayHandle,
CLIENT_GetLastError());
}
// 恢复回放
bSuccess = CLIENT_PausePlayBack(m_hPlayBack, FALSE);
if (!bSuccess)
{
    printf("CLIENT_PausePlayBack Failed, lPlayHandle[%x]!Last Error[%x]\n" , lPlayHandle,
CLIENT_GetLastError());
}
```

2.6 录像下载

2.6.1 简介

录像下载，即用户通过 SDK 获取存储设备上存有的录像并保存到本地的过程。允许用户对当前所选通道的录像进行下载，并可将视频导出到本地硬盘或者外接设备 U 盘等。

2.6.2 接口总览

表2-7 录像下载的接口信息

接口	说明
CLIENT_SetDeviceMode	设置设备语音对讲、回放和权限等工作模式接口。
CLIENT_QueryRecordFile	查询时间段内的所有录像文件的接口。
CLIENT_FindFile	打开录像查询句柄接口。
CLIENT_FindNextFile	查找录像文件接口。
CLIENT_FindClose	关闭录像查询句柄接口。
CLIENT_DownloadByRecordFileEx	按文件下载录像扩展接口。
CLIENT_DownloadByTimeEx	按时间下载录像扩展接口。
CLIENT_GetDownloadPos	查询录像下载进度。
CLIENT_StopDownload	停止录像下载接口。

2.6.3 流程说明

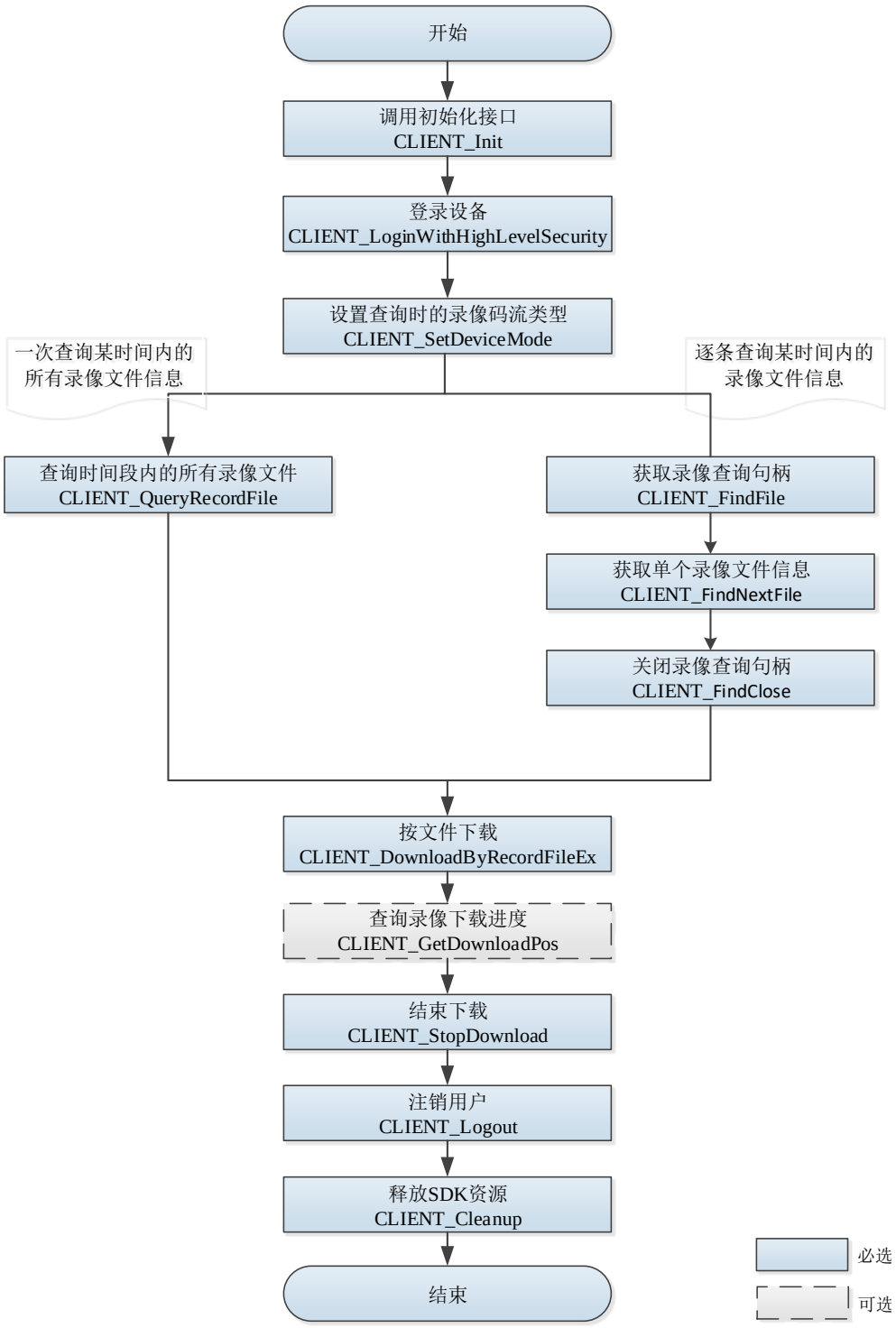
录像下载方式包括按文件下载和按时间下载。

2.6.3.1 按文件下载

即用户传入需要下载的录像文件信息，SDK 可将指定的录像文件下载并保存到用户指定的文件中。同时用户也可以提供一个回调函数的指针，SDK 将指定的录像文件的数据通过回调函数回调给用户，由用户自行处理。

按文件下载录像流程如图 2-8 所示。

图2-8 按文件下载录像流程图



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_SetDeviceMode 设置录像查询时的录像码流类型，对应 **emType** 为 **DH_RECORD_STREAM_TYPE**，建议设置为“0-主辅码流”，否则在少数设备上会无法获得结果。如果只需要主码流录像，可以自行在结果中滤除辅码流录像信息。

步骤4　可通过以下两种方式查询录像文件：

- 调用 CLIENT_FindFile 函数获取录像查询句柄，再循环调用 CLIENT_FindNextFile 逐次获取录像文件信息，最后调用 CLIENT_FindClose 关闭录像查询句柄。
- 调用 CLIENT_QueryRecordFile 一次性获取某时间段内的所有录像文件信息。

步骤5 获取到录像文件信息后，调用 CLIENT_DownloadByRecordFileEx 开始录像文件下载，形参 sSavedFileName 和 fDownLoadDataCallBack 中至少有一个需为有效值。可调用 cbDownLoadPos 实时获取下载进度，如果不适用置为 NULL 即可。

步骤6 （可选）下载过程中，根据用户需求调用 CLIENT_GetDownloadPos 查询录像下载进度。

步骤7 录像下载完毕后，调用 CLIENT_StopDownload 停止下载。

步骤8 业务使用完后，调用 CLIENT_Logout 登出设备。

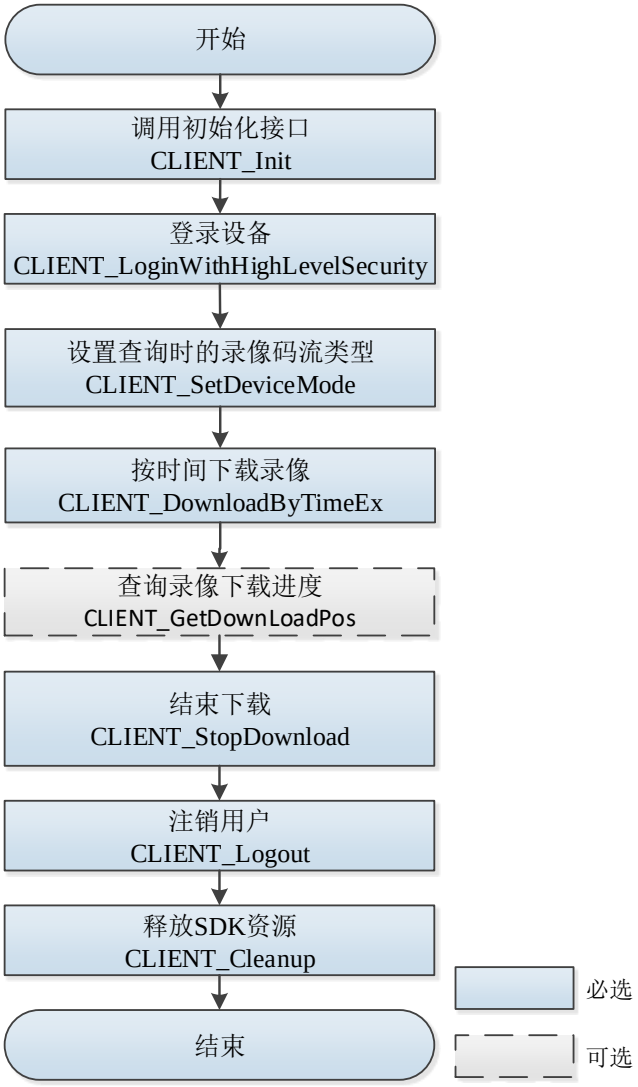
步骤9 SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

2.6.3.2 按时间下载

即用户传入需要下载的起始时间和结束时间，SDK 可将指定时间段内的录像文件下载并保存到用户指定的文件中。同时用户也可提供一个回调函数的指针，SDK 将指定时间段内的录像数据通过回调函数回调给用户，由用户自行处理。

按时间下载流程如图 2-9 所示。

图2-9 按时间下载流程图



流程说明

步骤1　调用 CLENT_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_SetDeviceMode 设置录像查询时的录像码流类型，对应 **emType** 为 **DH_RECORD_STREAM_TYPE**。

步骤4　调用 CLIENT_DownloadByTimeEx 接口函数开始按时间下载，形参 **sSavedFileName** 和 **fDownLoadDataCallBack** 中至少有一个需为有效值。**cbDownLoadPos** 用户可自行决定是否使用，如果不适用置为 **NULL** 即可。

步骤5　（可选）下载过程中，根据用户需求调用 CLIENT_GetDownloadPos 查询录像下载进度。同时，步骤 4 的 **cbDownLoadPos** 可以在下载的过程中实时获取下载进度，用户可根据需要选择两种获取方式。

步骤6　录像下载完毕后，调用 CLIENT_StopDownload 停止下载。根据需要可以等文件下载完成后关闭下载，也可以下载一部分就关闭下载。

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。

2.6.4 示例代码

2.6.4.1 按文件下载

```
// 回调函数声明
// 回放/下载 进度回调函数
// 不建议在该回调函数中调用 SDK 接口
// dwDownLoadSize: -1 时表示本次回放/下载结束，-2 表示写文件失败，其他值表示有效数据
// 通过 CLIENT_DownloadByRecordFileEx 设置该回调函数，当 SDK 收到回放/下载数据时，SDK 会调用该函数
void CALLBACK DownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, LDWORD dwUser);

// 回放/下载 数据回调函数
// 不建议在该回调函数中调用 SDK 接口
// 回放时：参数返回，0：表示本次回调失败，下次回调会返回相同的数据，1：表示本次回调成功，下次回调会返回后续的数据
// 下载时：不管回调函数返回值为多少都认为回调成功，下次回调会返回后续的数据
// 通过 CLIENT_DownloadByRecordFileEx 设置该回调函数，当 SDK 收到回放/下载数据时，SDK 会调用该函数
int CALLBACK DataCallBack(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser);

// 录像文件查询
    // 设置查询时的录像码流类型，此处设置码流类型为主辅码流
    int nStreamType = 0; // 0-主辅码流,1-主码流,2-辅码流
    CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_STREAM_TYPE, &nStreamType);
```

```
// 录像查询有两种实现方式：1，一次取完时间段内的所有录像文件；2，分次取时间段内所有录像文件。
// 方式 1：一次取完时间段内的所有录像文件
NET_TIME StartTime = {0};
NET_TIME StopTime = {0};
StartTime.dwYear = 2015;
StartTime.dwMonth = 9;
StartTime.dwDay = 20;
StartTime.dwHour = 0;
StartTime.dwMinute = 0;
StopTime.dwYear = 2015;
StopTime.dwMonth = 9;
StopTime.dwDay = 21;
StopTime.dwHour = 15;
NET_RECORDFILE_INFO netFileInfo[30] = {0};
NET_RECORDFILE_INFO stuNetFileInfo;
int nFileCount = 0;
//录像文件查询
if(!CLIENT_QueryRecordFile(lLoginHandle, nChannelID, (int)EM_RECORD_TYPE_ALL, &StartTime, &StopTime, NULL, &netFileInfo[0], sizeof(netFileInfo), &nFileCount,5000, FALSE))
{
printf("CLIENT_QueryRecordFile: failed! Error code: %x.\n", CLIENT_GetLastError());
}
else
{
// 将查询过来的第一个文件设置为下载文件
        memcpy(&stuNetFileInfo, (void *)&netFileInfo[0], sizeof(stuNetFileInfo));
}

    // 方式 2：分次取时间段内所有录像文件
    int nChannelID = 0; // 通道号
    NET_TIME stuStartTime = {0};
    stuStartTime.dwYear = 2015;
    stuStartTime.dwMonth = 9;
    stuStartTime.dwDay = 20;

    NET_TIME stuStopTime = {0};
    stuStopTime.dwYear = 2015;
    stuStopTime.dwMonth = 9;
    stuStopTime.dwDay = 30;
    int lFindHandle = CLIENT_FindFile(lLoginHandle, nChannelID, 0, NULL, &stuStartTime, &stuStopTime, FALSE, 5000);
    if (0 == lFindHandle)
    {
        printf("CLIENT_FindFile Failed!Last Error[%x]\n",CLIENT_GetLastError());
        return;
```

```
    }
    // demo 的示例代码，以最大支持 nMaxRecordFileCount 录像文件为例。
    std::vector<NET_RECORDFILE_INFO> bufFileInfo(nMaxRecordFileCount);

    for (int nFileIndex = 0; nFileIndex < nMaxRecordFileCount; ++nFileIndex)
    {
        int result = CLIENT_FindNextFile(lFindHandle, &bufFileInfo[nFileIndex]);
        if (0 == result)// 录像文件信息数据取完
        {
            break;
        }
        else if (1 != result)// 参数出错
        {
            printf("CLIENT_FindNextFile Failed!Last Error[%x]\n",CLIENT_GetLastError());
            break;
        }
    }

    //停止查找
    if(0 != lFindHandle)
    {
        CLIENT_FindClose(lFindHandle);
    }
    // 将查询过来的第一个文件设置为下载文件
    NET_RECORDFILE_INFO stuNetFileInfo;
    if (nFileIndex > 0)
    {
        memcpy(&stuNetFileInfo, (void *)&bufFileInfo[0], sizeof(stuNetFileInfo));
    }
    else
    {
        printf("no record, return\n");
        return;
    }

// 录像文件下载
    // 开启录像下载
    // 函数形参 sSavedFileName 和 fDownLoadDataCallBack 至少有一个为有效值
    // 实际应用中，一般根据需求选择直接保存至 sSavedFileName 或回调处理数据两者之一
    lDownloadHandle = CLIENT_DownloadByRecordFileEx(lLoginHandle, &stuNetFileInfo,
"test.dav", DownLoadPosCallBack, NULL, DataCallBack, NULL);
    if (0 == lDownloadHandle)
    {
        printf("CLIENT_DownloadByRecordFileEx: failed! Error code: %x.\n",
CLIENT_GetLastError());
    }
```

```
}

// 关闭下载，可在下载结束后调用，也可在下载中调用。
    if (0 != lDownloadHandle)
    {
        if (FALSE == CLIENT_StopDownload(lDownloadHandle))
        {
    printf("CLIENT_StopDownload Failed, lDownloadHandle[%x]!Last Error[%x]\n" ,
lDownloadHandle, CLIENT_GetLastError());
        }
        else
        {
            lDownloadHandle = 0;
        }
    }

//回调函数定义
void CALLBACK DownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD
dwDownLoadSize, LDWORD dwUser)
{
    // 若多个回放/下载使用相同的进度回调函数，则用户可通过 lPlayHandle 进行一一对应
    if (lPlayHandle == lDownloadHandle)
    {
        printf("lPlayHandle[%p]\n", lPlayHandle);
        printf("dwTotalSize[%d]\n", dwTotalSize);
        printf("dwDownLoadSize[%d]\n", dwDownLoadSize);
        printf("dwUser[%p]\n", dwUser);
        printf("\n");
    }
}

int CALLBACK DataCallBack(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD
dwBufSize, LDWORD dwUser)
{
    int nRet = 0;
    printf("call DataCallBack\n");
    // 若多个回放/下载使用相同的数据回调函数，则用户可通过 lRealHandle 进行一一对应
    if(lRealHandle == lDownloadHandle)
    {
        printf("lPlayHandle[%p]\n", lRealHandle);
        printf("dwDataType[%d]\n", dwDataType);
        printf("pBuffer[%p]\n", pBuffer);
        printf("dwBufSize[%d]\n", dwBufSize);
        printf("dwUser[%p]\n", dwUser);
        printf("\n");
        switch(dwDataType)
```

```
        {
        case 0:
            //Original data
            // 用户在此处保存码流数据，离开回调函数后再进行解码或转发等一系列处理
            nRet = 1;

            break;
        case 1:
            //Standard video data

            break;
        case 2:
            //yuv data

            break;
        case 3:
            //pcm audio data

            break;
        default:
            break;
        }
    }
    return nRet;
}
```

2.6.4.2 按时间下载

```
// 回调函数声明
// 按时间回放进度回调函数
// 不建议在该回调函数中调用 SDK 接口
// dwDownLoadSize: -1 时表示本次回放/下载结束，-2 表示写文件失败，其他值表示有效数据
// 通过 CLIENT_DownloadByTimeEx 设置该回调函数，当 SDK 收到回放/下载数据时，SDK 会调用该函数
void CALLBACK TimeDownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, int index, NET_RECORDFILE_INFO recordfileinfo, LDWORD dwUser);

// 回放/下载 数据回调函数
// 不建议在该回调函数中调用 SDK 接口
// 回放时：参数返回，0：表示本次回调失败，下次回调会返回相同的数据，1：表示本次回调成功，下次回调会返回后续的数据
// 下载时：不管回调函数返回值为多少都认为回调成功，下次回调会返回后续的数据
// 通过 CLIENT_DownloadByTimeEx 设置该回调函数，当 SDK 收到回放/下载数据时，SDK 会调用该函数
int CALLBACK DataCallBack(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser);
```

```
// 设置查询时的录像码流类型，此处设置码流类型为主辅码流
    int nStreamType = 0; // 0-主辅码流,1-主码流,2-辅码流
    CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_STREAM_TYPE, &nStreamType);
    // 设置下载开始和结束时间
    int nChannelID = 0; // 通道号
    NET_TIME stuStartTime = {0};
    stuStartTime.dwYear = 2015;
    stuStartTime.dwMonth = 9;
    stuStartTime.dwDay = 17;

    NET_TIME stuStopTime = {0};
    stuStopTime.dwYear = 2015;
    stuStopTime.dwMonth = 9;
    stuStopTime.dwDay = 18;

// 开启录像下载
    // 函数形参 sSavedFileName 和 fDownLoadDataCallBack 需至少有一个为有效值，否则入参有误
    lDownloadHandle = CLIENT_DownloadByTimeEx(lLoginHandle, nChannelID, EM_RECORD_TYPE_ALL, &stuStartTime, &stuStopTime, "test.dav", TimeDownLoadPosCallBack, NULL, DataCallBack, NULL);
    if (lDownloadHandle == 0)
    {
        printf("CLIENT_DownloadByTimeEx: failed! Error code: %x.\n", CLIENT_GetLastError());
    }
}

// 关闭下载，可在下载结束后调用，也可在下载中调用
    if (0 != lDownloadHandle)
    {
        if (FALSE == CLIENT_StopDownload(lDownloadHandle))
        {
            printf("CLIENT_StopDownload Failed, lDownloadHandle[%x]!Last Error[%x]\n" , lDownloadHandle, CLIENT_GetLastError());
        }
        else
        {
            lDownloadHandle = 0;
        }
    }

// 回调函数定义
void CALLBACK TimeDownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, int index, NET_RECORDFILE_INFO recordfileinfo, LDWORD dwUser)
```

```
{
    // 若多个回放/下载使用相同的进度回调函数，则用户可通过 lPlayHandle 进行一一对应
    if (lPlayHandle == lDownloadHandle)
    {
        printf("lPlayHandle[%p]\n", lPlayHandle);
        printf("dwTotalSize[%d]\n", dwTotalSize);
        printf("dwDownLoadSize[%d]\n", dwDownLoadSize);
        printf("index[%d]\n", index);
        printf("dwUser[%p]\n", dwUser);
        printf("\n");
    }
}

int CALLBACK DataCallBack(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD
dwBufSize, LDWORD dwUser)
{
    int nRet = 0;
    printf("call DataCallBack\n");
    // 若多个回放/下载使用相同的数据回调函数，则用户可通过 lRealHandle 进行一一对应
    if(lRealHandle == lDownloadHandle)
    {
        printf("lPlayHandle[%p]\n", lRealHandle);
        printf("dwDataType[%d]\n", dwDataType);
        printf("pBuffer[%p]\n", pBuffer);
        printf("dwBufSize[%d]\n", dwBufSize);
        printf("dwUser[%p]\n", dwUser);
        printf("\n");
        switch(dwDataType)
        {
        case 0:
            //Original data
            // 用户在此处保存码流数据，离开回调函数后再进行解码或转发等一系列处理
            nRet = 1;//

            break;
        case 1:
            //Standard video data

            break;
        case 2:
            //yuv data

            break;
        case 3:
            //pcm audio data
```

```
                break;
            default:
                break;
            }
        }
        return nRet;
}
```

2.7 云台控制

2.7.1 简介

云台是指承载摄像设备及防护罩并能够远程进行全方位控制的机械平台。云台实质上是由两个电机组成，可以实现水平和垂直的运动，从而给摄像机设备全方位、多角度的视野。

本节主要指导用户如何通过 SDK 实现方向控制（简称八方位控制，具体包括上、下、左、右，左上、左下、右上和右下）、聚焦、变倍、光圈、快速定位和精确三维定位功能。

2.7.2 接口总览

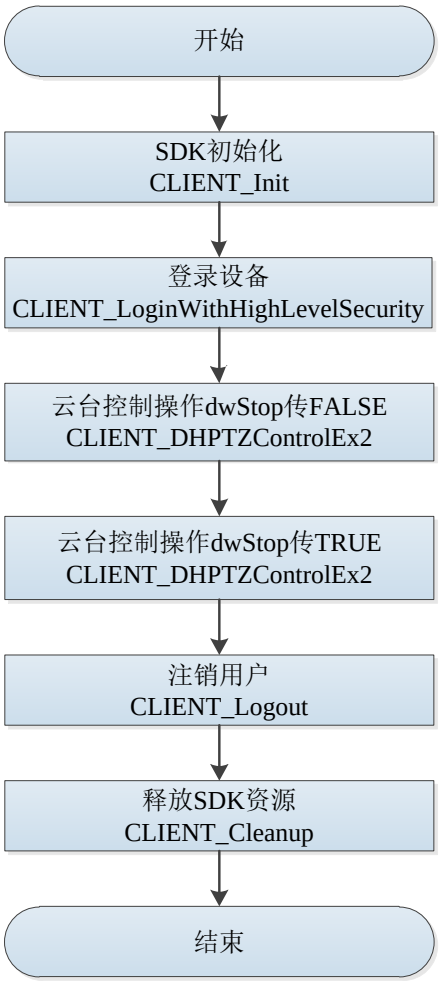
表2-8 云台控制接口信息

接口	说明
CLIENT_DHPTZControlEx2	云台控制扩展接口

2.7.3 流程说明

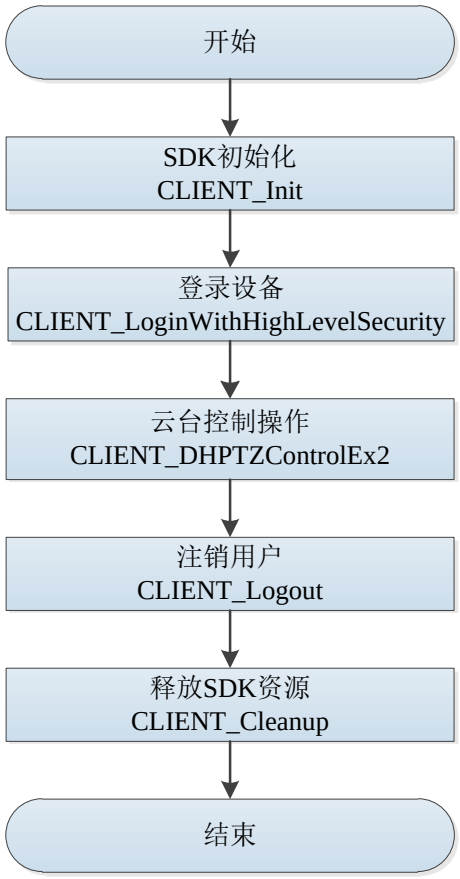
方向控制、聚焦、变倍和光圈属于持续的动作，SDK 提供启动和停止的接口，由用户自行控制启动停止的时机。流程如图 2-10 所示。

图2-10 云台控制流程图（持续类操作）



快速定位与精确三维定位属于一次动作，只需调用一次云台控制接口，流程如图 2-11 所示。

图2-11 云台控制流程图（一次性操作）



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　根据需求调用 CLIENT_DHPTZControlEx2 接口操作云台。不同的云台命令可能需要不同的参数，部分操作命令需要调用相应的停止命令，比如左右移动操作，具体请参见“2.7.4 示例代码”。

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

- 快速定位：以球机当前预览画面中心为原点，水平坐标和垂直坐标有效范围均为[-8191，8191]。例如传入水平坐标 2000，垂直坐标 2000，球机将向右上方转动并以新的点为坐标原点，即每次指定的坐标都是相对与当前位置的。
- 三维精确定位：首先球机有一个初始位置，以角度为单位，水平坐标[0，3600]，垂直坐标[-1800，1800]，每次指定的坐标都是绝对的坐标，和前一次球机画面所处位置无关。
- 更多示例源码见官网发布包。

2.7.4 示例代码

```
LONG lParam1 = 0; // 水平方向转动速度
```

```
LONG lParam2 = 4; // 垂直方向转动速度
LONG lParam3 = 0;
// 持续动作的调用方法以向上转动为例
// 开始向上转动
BOOL bRet = CLIENT_DHPTZControlEx2(lLoginHandle, nChannelId, DH_PTZ_UP_CONTROL, lParam1, lParam2, lParam3, FALSE, NULL);
// 结束向上转动
bRet = CLIENT_DHPTZControlEx2(lLoginHandle, nChannelId, DH_PTZ_UP_CONTROL, lParam1, lParam2, lParam3, TRUE, NULL);
// 一次动作的调用方法以快速定位为例
lParam1 = 2000;   // 水平坐标,有效范围[-8191,8191]
lParam2 = 2000;   // 垂直坐标,有效范围[-8191,8191]
lParam3 = 1;      // 变倍,有效范围(-16 ～ 16),1 表示仅转动, 不变倍
bRet = CLIENT_DHPTZControlEx2(lLoginHandle, nChannelId, DH_EXTPTZ_FASTGOTO, lParam1, lParam2, lParam3, FALSE, NULL);
```

2.8 语音对讲

2.8.1 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。

本章主要介绍用户如何使用 SDK 实现与前端设备的语音对讲。

2.8.2 接口总览

表2-9 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
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）

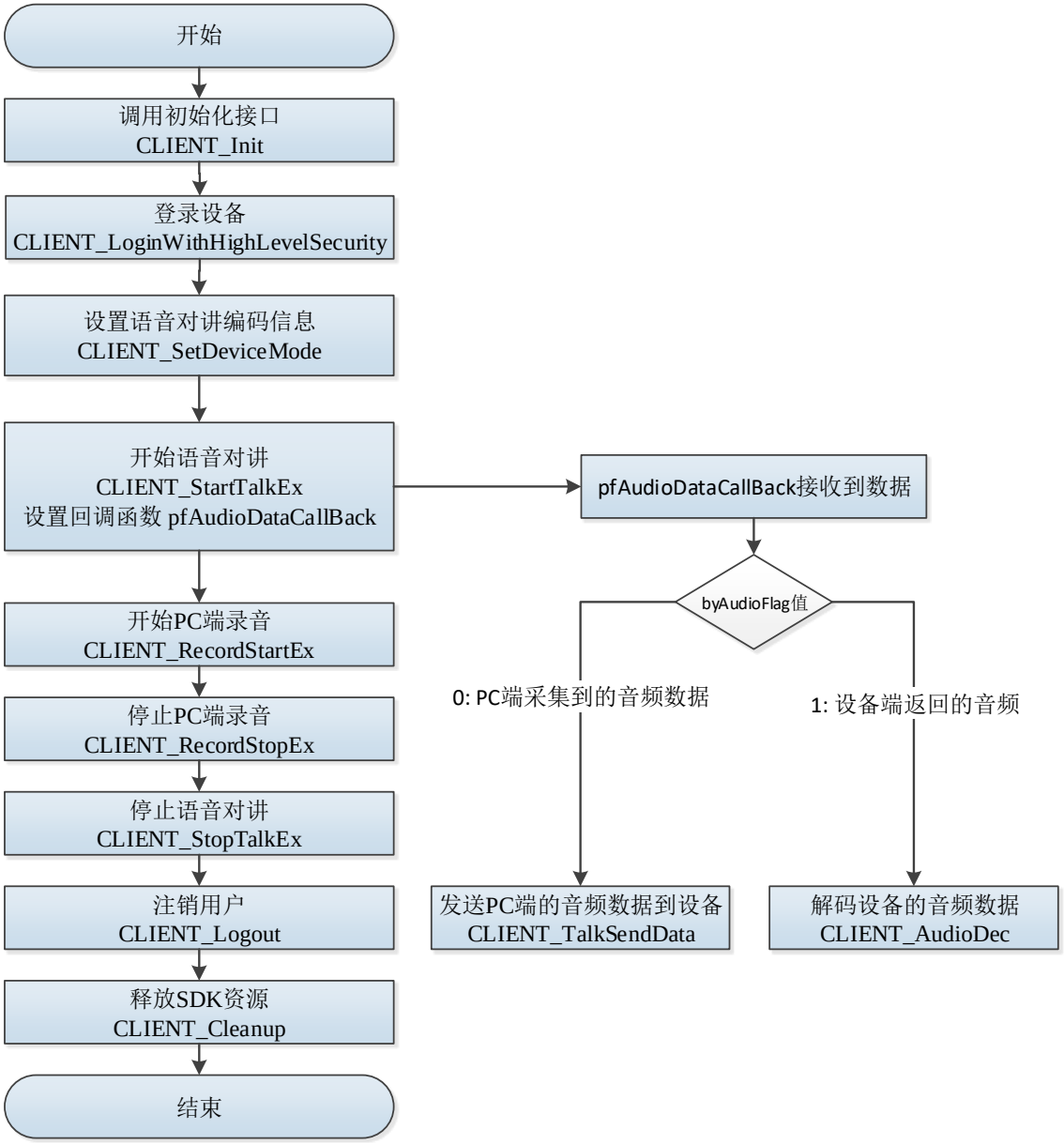
2.8.3 流程说明

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。

该模式只在 Windows 平台下有效。流程如图 2-12 所示。

图2-12 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_SetDeviceMode 设置语音对讲编码信息，参数 **emType** 设置为 **DH_TALK_ENCODE_TYPE**。

步骤4　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5　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6　对讲功能使用完毕后，调用 CLIENT_RecordStopEx 停止 PC 端录音。

步骤7　调用 CLIENT_StopTalkEx 停止语音对讲。

步骤8　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

- 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。

2.8.4 示例代码

```
// 设置语音对讲编码信息,以 PCM 为例
DHDEV_TALKDECODE_INFO curTalkMode;
curTalkMode.encodeType = DH_TALK_PCM;
curTalkMode.nAudioBit = 16;
curTalkMode.dwSampleRate = 8000;
curTalkMode.nPacketPeriod = 25;
CLIENT_SetDeviceMode(lLoginHandle, DH_TALK_ENCODE_TYPE, &curTalkMode);
// 开始语音对讲
lTalkHandle = CLIENT_StartTalkEx(lLoginHandle, AudioDataCallBack, (LDWORD)NULL);
if(0 != lTalkHandle)
{
    BOOL bSuccess = CLIENT_RecordStartEx(lLoginHandle);
}

// 停止本地录音
if (!CLIENT_RecordStopEx(lLoginHandle))
{
printf("CLIENT_RecordStop Failed!Last Error[%x]\n", CLIENT_GetLastError());
}
// 停止语音对讲
if (0 != lTalkHandle)
{
CLIENT_StopTalkEx(lTalkHandle);
}
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
        LONG lSendLen = CLIENT_TalkSendData(lTalkHandle, pDataBuf, dwBufSize);
        if(lSendLen != (LONG)dwBufSize)
        {
            printf("CLIENT_TalkSendData Failed!Last Error[%x]\n" , CLIENT_GetLastError());
```

```
        }
    }
    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);
    }
}
```

2.9 视频抓图

2.9.1 简介

视频抓图，即获取前端产品当前画面的图片数据。本章介绍以下两种抓图方式：

- 网络抓图：用户调用 SDK 接口，接口内部发送抓图命令给设备，设备抓取当前画面并通过网络发送给 SDK，SDK 将接收到的图片数据返回给用户。
- 本地抓图：用户在已打开预览的前提下，可以将监控中的数据保存为图片，该图片是从显示画面中保存帧信息，与设备之间没有网络交互。

2.9.2 接口总览

表2-10 视频抓图的接口信息

接口	说明
CLIENT_SnapPictureToFile	抓图同步接口，将图片数据直接返回给用户
CLIENT_CapturePictureEx	本地抓图，参数可以是预览或回放的句柄

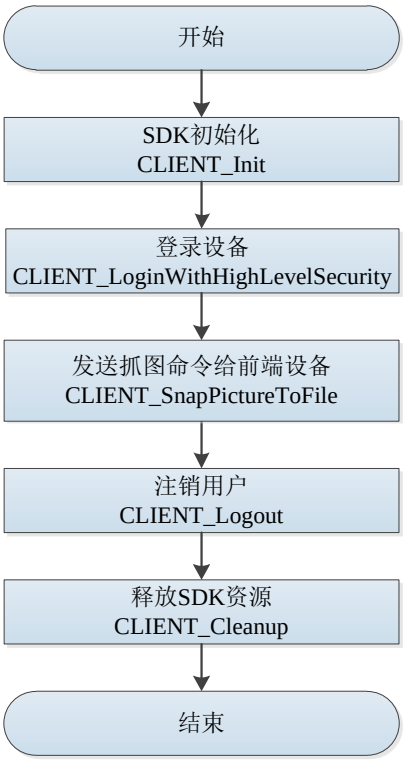
2.9.3 流程说明

视频抓图可分为网络抓图和本地抓图两种。

2.9.3.1 网络抓图

网络抓图流程如图 2-13 所示。

图2-13 网络抓图流程图



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_SnapPictureToFile 获取图片信息。
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。

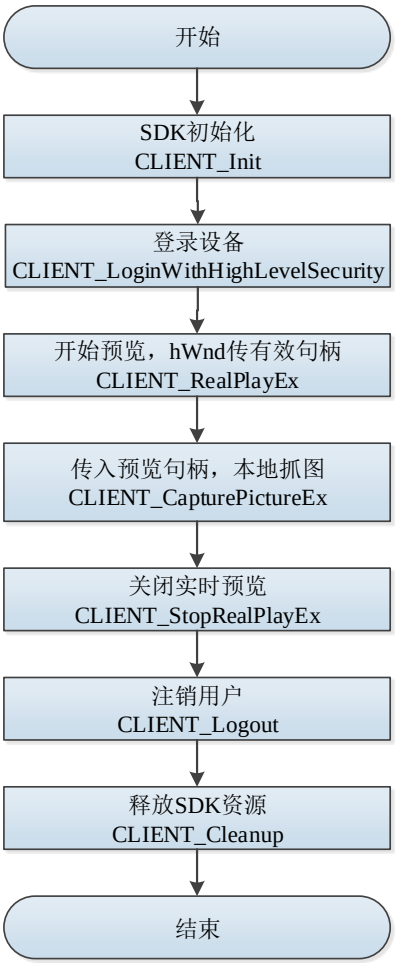
注意事项

- 多线程调用：同一个登录会话内的业务，不支持多线程调用。
- 抓图配置：网络抓图的图片质量、分辨率等可以设置的。但是如果默认配置效果能满足，建议不用修改，示例源码详见官网发布包。
- 图片保存形式：图片数据以内存的形式返回，同时接口支持保存成文件（前提是用户已设置 NET_IN_SNAP_PIC_TO_FILE_PARAM 的 **szFilePath** 字段）。

2.9.3.2 本地抓图

本地抓图流程如图 2-14 所示。

图2-14 本地抓图流程图



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_RealPlayEx 开启预览，获取预览句柄。
步骤4　调用 CLIENT_CapturePictureEx 传入预览句柄。
步骤5　调用 CLIENT_StopRealPlayEx 关闭实时预览。
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。

2.9.4 示例代码

```
// 网络抓图示例
NET_IN_SNAP_PIC_TO_FILE_PARAM stuInParam = {sizeof(stuInParam)};
NET_OUT_SNAP_PIC_TO_FILE_PARAM stuOutParam = {sizeof(stuOutParam)};
SNAP_PARAMS stuSnapParams = {0};
stuSnapParams.Channel = 0;     // 以第一个通道为例
int nBufferLen = 2*1024*1024;
char* pBuffer = new char[nBufferLen]; // 图片缓存
memset(pBuffer, 0, nBufferLen);
```

```
stuOutParam.szPicBuf = pBuffer;
stuOutParam.dwPicBufLen = nBufferLen;
if (FALSE == CLIENT_SnapPictureToFile(lLoginHandle, &stuSnapParams))
{
printf("CLIENT_SnapPictureEx Failed!Last Error[%x]\n", CLIENT_GetLastError());
}
delete[] pBuffer;

// 本地抓图示例，hPlayHandle 为打开预览获取的句柄
if (FALSE == CLIENT_CapturePictureEx(hPlayHandle, “test.jpg”, NET_CAPTURE_JPEG))
{
printf("CLIENT_CapturePictureEx Failed!Last Error[%x]\n", CLIENT_GetLastError());
}
```

2.10 报警上报

2.10.1 简介

报警上报实现方式为：通过 SDK 登录设备并向设备订阅报警功能，设备检测到报警事件立即发送给 SDK，通过报警回调函数即可获取相应报警信息。

2.10.2 接口总览

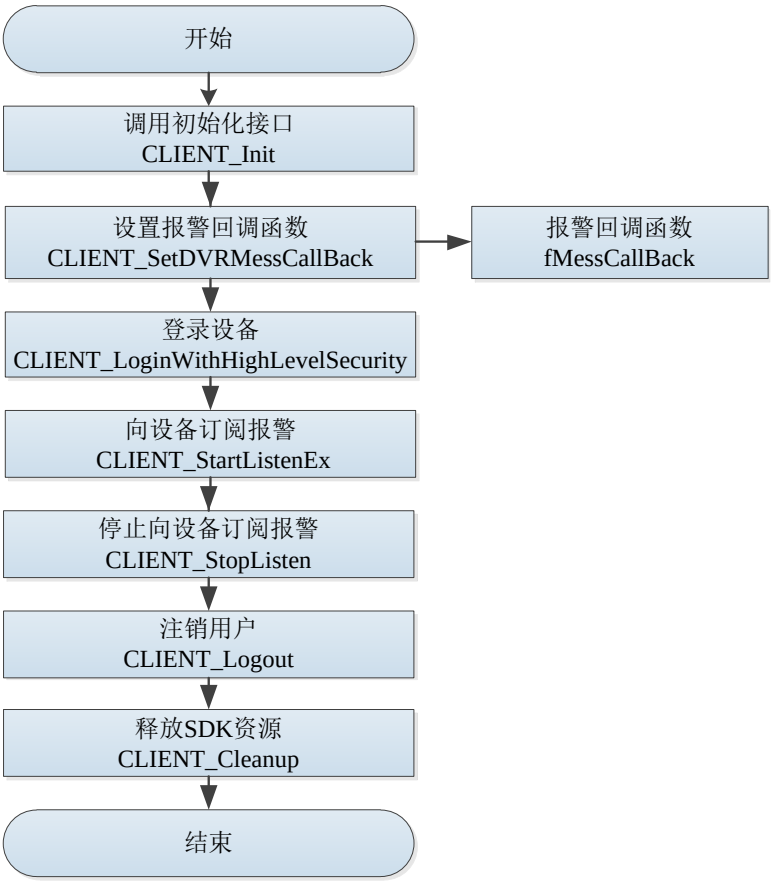
表2-11 报警上报的接口信息

接口	说明
CLIENT_SetDVRMessCallBack	设置报警回调函数接口
CLIENT_StartListenEx	订阅报警扩展接口
CLIENT_StopListen	停止订阅报警

2.10.3 流程说明

报警上报流程如图 2-15 所示。

图2-15 报警上报流程



流程说明

步骤1　调用 CLIENT_Init 函数，完成 SDK 初始化流程。

步骤2　调用 CLIENT_SetDVRMessCallBack 函数，设置报警回调函数，该接口需在报警订阅之前调用。

步骤3　调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。

步骤4　调用 CLIENT_StartListenEx 函数，向设备订阅报警。订阅成功后，设备上报的报警事件通过 CLIENT_SetDVRMessCallBack 函数设置的回调函数通知用户。

步骤5　报警上报功能使用完毕后，调用 CLIENT_StopListen 函数停止向设备订阅报警。

步骤6　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

如果之前能上报的报警突然不再上报了，则需排查下设备是否断线。如果断线，设备自动重连后是不会再有报警上报的，需要取消订阅后重新订阅。

2.10.4 示例代码

```
// 报警回调
int CALLBACK    afMessCallBack(LONG lCommand, LLONG lLinID, char *pBuf, DWORD
dwBufLen,
```

```
char *pchDVRIP, LONG nDVRPort, LDWORD dwUser)
{
if(lCommand == DH_ALARM_STORAGE_FAILURE_EX)//硬盘故障报警
{
        printf("硬盘故障报警上报");
ALARM_STORAGE_FAILURE_EX* pstStorageFailureInfo = (ALARM_STORAGE_FAILURE_EX*)pBuf;
    //接下来可以通过 pstStorageFailureInfo 取对应的报警信息。
}
if(lCommand == DH_DISKERROR_ALARM_EX) // 坏硬盘报警
    {
printf("硬盘损坏报警上报\n");
//视频丢失报警不携带任何报警信息
    }
}

// 设置报警回调
CLIENT_SetDVRMessCallBack(&afMessCallBack,0);
// 订阅报警
BOOL bRet = CLIENT_StartListenEx(lLoginHandle);
if(!bRet)
{
    printf("CLIENT_StartListenEx Failed, Last Error[%x]\n" ,
    CLIENT_GetLastError());
}
else
{
printf("listen succeed.\n");
}

// 停止报警订阅
BOOL bRet = CLIENT_StopListen(lLoginHandle);
if(!bRet)
{
    printf("CLIENT_StopListen Failed, Last Error[%x]\n" ,
    CLIENT_GetLastError());
}
else
{
printf("stop listen succeed.\n");
}
```

2.11 存储

2.11.1 简介

存储接口，即与存储设备对接常用的接口，主要包括远程设备信息获取、远程设备连接状态查询订阅及远程通道名称获取修改等。

2.11.2 接口总览

表2-12 存储接口信息

接口	说明
CLIENT_QueryDevState	直接获取远程设备连接状态，每个通道单独获取。 参数 type 为 DH_DEVSTATE_VIRTUALCAMERA。
CLIENT_QueryDevInfo	直接获取远程设备连接状态，所有通道统一获取。 参数 nQueryType 为 NET_QUERY_GET_CAMERA_STATE。
CLIENT_AttachCameraState	订阅远程设备状态。 当远程设备状态发生变化时，会有相应设备信息上报。
CLIENT_DetachCameraState	停止订阅远程设备状态。 与 CLIENT_AttachCameraState 配对使用。
CLIENT_MatrixGetCameras	获取远程设备信息，如设备型号、IP 等信息。
CLIENT_QueryChannelName	获取通道名称。
CLIENT_GetNewDevConfig	获取通道名称。 参数 szCommand 为 CFG_CMD_VIDEOIN 或 CFG_CMD_CHANNELTITLE。
CLIENT_ParseData	解析数据，与 CLIENT_GetNewDevConfig 配对使用。

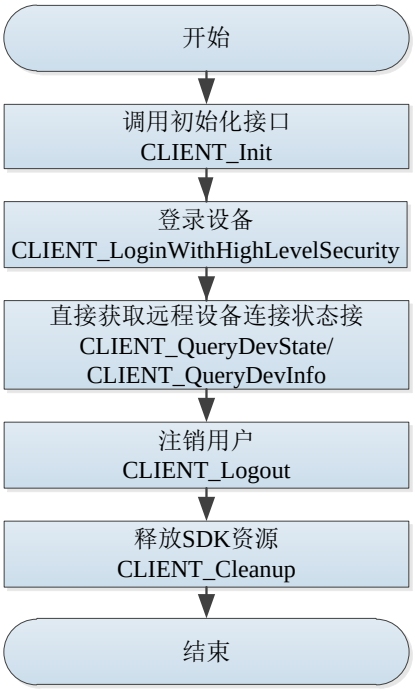
2.11.3 流程说明

存储设备主要涉及的业务为：直接获取远程设备连接状态、订阅远程设备连接状态、获取远程设备信息和获取通道名称。下面分别介绍各业务流程。

2.11.3.1 直接获取远程设备连接状态

直接获取远程设备连接状态流程如图 2-16 所示。

图2-16 直接获取远程设备连接状态流程图



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　直接获取远程设备连接状态。直接获取远程设备状态目前有两个接口，分别是 CLIENT_QueryDevState (**type** 为 **DH_DEVSTATE_VIRTUALCAMERA**)和 CLIENT_QueryDevInfo(**QueryType** 为 **NET_QUERY_GET_CAMERA_STATE**)。

说明

设备不同，两个接口支持的情况不同。建议使用前自行测试，根据支持情况选择合适的接口。

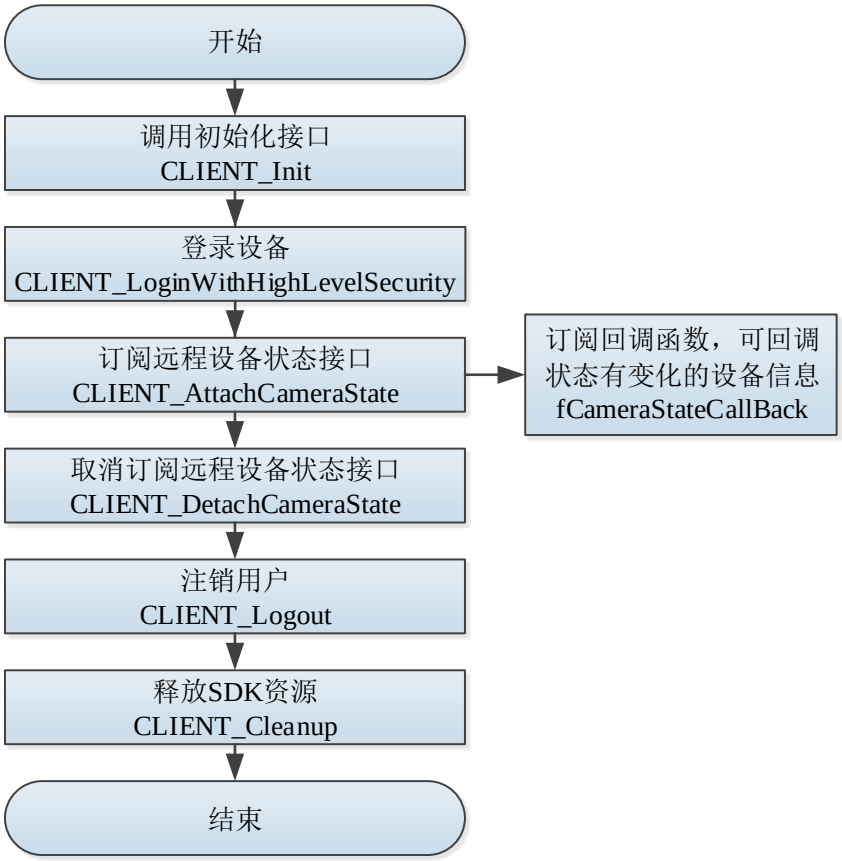
步骤4　调用 CLIENT_Logout 登出设备。

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

2.11.3.2 订阅远程设备状态

订阅远程设备状态流程如图 2-17 所示。

图2-17 订阅远程设备状态流程图



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_AttachCameraState 订阅远程设备连接状态。订阅之后，如果远程设备状态有变化，则会通过回调函数 fCameraStateCallBack 上报设备信息。

步骤4　调用 CLIENT_DetachCameraState 取消订阅远程设备状态。取消订阅后，远程设备状态即使有变化，也不会上报。

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。

2.11.3.3 获取远程设备信息

获取远程设备信息流程如图 2-18 所示。

图2-18 获取远程设备信息流程图



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_MatrixGetCameras 获取远程设备信息，如设备型号、IP 等信息。
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。

2.11.3.4 获取通道名称

获取通道名称流程如图 2-19 所示。

图2-19 获取通道名称流程图



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　获取通道名称目前有两种方式，第一种接口是 CLIENT_QueryChannelName，第二种接口是 CLIENT_GetNewDevConfig&&CLIENT_ParseData。其中第二种接口有两个 command 可以获取通道名称，分别为 CFG_CMD_VIDEOIN 和 CFG_CMD_CHANNELTITLE。

说明

设备不同，两个接口支持的情况不同。建议使用前自行测试，根据支持情况选择合适的接口。

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。

2.11.4 示例代码

2.11.4.1 直接获取远程设备连接状态

```
// 获取远程设备连接状态接口 1
DHDEV_VIRTUALCAMERA_STATE_INFO stInOut = {sizeof(stInOut)};
```

```
int nRetLen = 0;
for(int i=0;i< nChnCount;i++)
{
	stInOut.nChannelID = i;//按通道逐个获取
	BOOL bRet = CLIENT_QueryDevState(lLoginHandle,DH_DEVSTATE_VIRTUALCAMERA,
(char *)&stInOut,sizeof(stInOut),&nRetLen,3000);
	if(bRet)
	{
		if(stInOut.emConnectState == 0)
		{
			printf("通道%d 的通道状态为《未连接》\n",i);
		}
		else if (stInOut.emConnectState == 1)
		{
			printf("通道%d 的通道状态为《正在连接》\n",i);
		}
		else if (stInOut.emConnectState == 2)
		{
			printf("通道%d 的通道状态为《已连接》\n",i);
		}
		memset((void *)&stInOut,0,sizeof(stInOut));
		stInOut.nStructSize = sizeof(stInOut);
	}
	else
	{
		printf("get virtual camera state failed.");
	}
}
else
	{
		printf("CLIENT_QueryDevState Failed, Last Error[%x]\n",
			CLIENT_GetLastError());
}

// 获取远程设备连接状态接口 2
NET_IN_GET_CAMERA_STATEINFO stIn = {sizeof(stIn)};
	stIn.bGetAllFlag = TRUE;
	NET_OUT_GET_CAMERA_STATEINFO stOut = {sizeof(stOut)};
	stOut.pCameraStateInfo = new NET_CAMERA_STATE_INFO[64];
	memset(stOut.pCameraStateInfo,0,sizeof(*stOut.pCameraStateInfo)*64);
	stOut.nMaxNum = 64;
	BOOL bRet = CLIENT_QueryDevInfo(lLoginHandle,NET_QUERY_GET_CAMERA_STATE,
&stIn,&stOut,NULL,3000);
	if(bRet)
	{
		for(int i = 0;i<stOut.nValidNum;i++)
```

```
        {
            if(stOut.pCameraStateInfo[i].emConnectionState == 0)
            {
                cout << "当前通道"<<stOut.pCameraStateInfo[i].nChannel<<
"状态:"<<"未知"<<endl;
            }
            else if (stOut.pCameraStateInfo[i].emConnectionState == 1)
            {
                cout << "当前通道"<<stOut.pCameraStateInfo[i].nChannel<<
"状态:"<<"正在连接"<<endl;
            }
            else if (stOut.pCameraStateInfo[i].emConnectionState == 2)
            {
                cout << "当前通道"<<stOut.pCameraStateInfo[i].nChannel<<
"状态:"<<"已连接"<<endl;
            }
            else if (stOut.pCameraStateInfo[i].emConnectionState == 3)
            {
                cout << "当前通道"<<stOut.pCameraStateInfo[i].nChannel<<
"状态:"<<"未连接"<<endl;
            }
            else if (stOut.pCameraStateInfo[i].emConnectionState == 4)
            {
                cout << "当前通道"<<stOut.pCameraStateInfo[i].nChannel<<
"状态:"<<"通道未配置，无信息"<<endl;
            }
            else if (stOut.pCameraStateInfo[i].emConnectionState == 5)
            {
                cout << "当前通道"<<stOut.pCameraStateInfo[i].nChannel<<"状态:"<<
"通道有配置,但被禁用"<<endl;
            }
        }
            cout<<"succeed"<<endl;
    }
    else
    {
        printf("CLIENT_QueryDevInfo Failed, Last Error[%x]\n",
            CLIENT_GetLastError());

}
```

2.11.4.2 订阅远程设备状态

```
// 订阅远程设备状态
// 远程设备状态回调函数声明，如远程设备状态有变化，则从此回调函数返回。
    void CALLBACK AfCameraStateCallBack(LLONG lLoginID, LLONG lAttachHandle, const
```

```
NET_CB_CAMERASTATE *pBuf, int nBufLen, LDWORD dwUser);

    // 订阅远程设备状态
NET_IN_CAMERASTATE stIn1 = {sizeof(stIn1)};
    int nChannel = -1;
    stIn1.pChannels = &nChannel;
    stIn1.nChannels = 4;
    stIn1.cbCamera = AfCameraStateCallBack;
    NET_OUT_CAMERASTATE stOut1 = {sizeof(stOut1)};
    LLONG llRet = CLIENT_AttachCameraState(lLoginHandle,&stIn1,&stOut1,3000);
    if(llRet)
    {
        printf("订阅成功，接下来如有前端设备状况发生变化，会上报..... \n");
    }
    else
    {
        printf("CLIENT_AttachCameraState Failed, Last Error[%x]\n",
        CLIENT_GetLastError());
}

// 停止订阅远程设备状态
If(llRet)
{
    BOOL bRet = CLIENT_DetachCameraState(llRet);
    If(bRet)
{
        printf(“取消订阅成功。\n”);
}
else
{
                printf("CLIENT_DetachCameraState Failed, Last Error[%x]\n",
                    CLIENT_GetLastError());
}
}

// 远程设备状态回调函数定义
void CALLBACK AfCameraStateCallBack(LLONG lLoginID, LLONG lAttachHandle, const
NET_CB_CAMERASTATE *pBuf, int nBufLen, LDWORD dwUser)
{
        if(pBuf->emConnectState == 0)
        {
            printf("当前通道%d 状态为《未连接》\n", pBuf->nChannel);
        }
        else if (pBuf->emConnectState == 1)
        {
printf("当前通道%d 状态为《正在连接》\n", pBuf->nChannel);
```

```
        }
        else if (pBuf->emConnectState == 2)
        {
            printf("当前通道%d 状态为《已连接》\n", pBuf->nChannel);
        }
}
```

2.11.4.3 获取远程设备信息

```
// 获取远程设备信息接口 1
// nChanNum 为登录接口返回的通道个数
NET_IN_GET_CAMERA_STATEINFO stuInfo = { sizeof(NET_IN_GET_CAMERA_STATEINFO), TRUE };
NET_OUT_GET_CAMERA_STATEINFO stuOutInfo =
{ sizeof(NET_OUT_GET_CAMERA_STATEINFO) };
NET_CAMERA_STATE_INFO* pstuArrayStatInfo = new NET_CAMERA_STATE_INFO[nChanNum];
memset(pstuArrayStatInfo,0,sizeof(NET_CAMERA_STATE_INFO)*nChanNum);
stuOutInfo.nMaxNum = nChanNum;
stuOutInfo.pCameraStateInfo = pstuArrayStatInfo;
printf("State(0:UNKNOWN,1:CONNECTING,2:CONNECTED,3:UNCONNECT,4:EMPTY,5:DISABLE).\
n");
BOOL bRet = CLIENT_QueryDevInfo(lLoginHandle, NET_QUERY_GET_CAMERA_STATE, &stuInfo,
&stuOutInfo, NULL, 2000);
if (bRet)
{
printf("CLIENT_QueryDevInfo NET_QUERY_GET_CAMERA_STATE success.\n");
for (int i = 0; i < stuOutInfo.nValidNum; i++)
{
printf("channel:%d,Status:%d.\n",
stuOutInfo.pCameraStateInfo[i].nChannel,stuOutInfo.pCameraStateInfo[i].emConnectionState);
}
}
else
{
printf("CLIENT_QueryDevInfo Failed, Last Error[%x]\n",
CLIENT_GetLastError());
}

// 获取远程设备信息接口 2
DH_IN_MATRIX_GET_CAMERAS stuInParm = {sizeof(DH_IN_MATRIX_GET_CAMERAS)};
DH_OUT_MATRIX_GET_CAMERAS stuOutParam = {sizeof(DH_OUT_MATRIX_GET_CAMERAS)};
DH_MATRIX_CAMERA_INFO   stuAllmatrixcamerinfo[128] = {0};
stuOutParam.nMaxCameraCount = nChanNum; //这里获取到的最大数
stuOutParam.pstuCameras = stuAllmatrixcamerinfo;
for (int i=0;i< __min(stuOutParam.nMaxCameraCount,stuOutParam.nRetCameraCount);++i)
{
stuOutParam.pstuCameras[i].dwSize = sizeof(DH_MATRIX_CAMERA_INFO);
```

```
stuOutParam.pstuCameras[i].stuRemoteDevice.dwSize = sizeof(DH_REMOTE_DEVICE);
}
int iNumbers = 1;
// 获取所有有效显示源
BOOL bRet = CLIENT_MatrixGetCameras(lLoginHandle, &stuInParm, &stuOutParam);
printf("ALL the Device list Info Begin:\n");
if(bRet)
{
int iChannelNumbers =0;
      char szUserInput[32] = "";
memset(szUserInput, 0, sizeof(szUserInput));
printf("too many channels info:Input    your show numbers: ==>\n");
gets(szUserInput);
iChannelNumbers = atoi(szUserInput);
for ( int j=0;j<__min(stuOutParam.nRetCameraCount,iChannelNumbers);++j)
{
DH_MATRIX_CAMERA_INFO stuinfo = stuOutParam.pstuCameras[j];
if( TRUE)// 是否远程设备
{
      switch (stuinfo.emChannelType)
      {
      case LOGIC_CHN_REMOTE:
            {
                printf("This is LOGIC_CHN_REMOTE(远程通道):\n");
break;
}
case LOGIC_CHN_LOCAL:
      {
            printf("This is LOGIC_CHN_LOCAL(本地通道):\n");
            break;
      }
case LOGIC_CHN_COMPOSE:
      {
            printf("This is LOGIC_CHN_COMPOSE(合成通道):\n");
            break;
      }
case LOGIC_CHN_MATRIX:
      {
printf("This is LOGIC_CHN_MATRIX(模拟矩阵通道):\n");
break;
}
case LOGIC_CHN_CASCADE:
      {
            printf("This is LOGIC_CHN_CASCADE(级联通道):\n");
            break;
      }
```

```
default:
    {
        printf("This is LOGIC_CHN_UNKNOWN(未知通道):\n");
    }
}
printf("..........\n");
printf("This is the %d remote camera:\n",iNumbers++);
printf("Dev Remote ChannelID = %d,the Local
nUniqueChannel = %d.\n",stuinfo.nChannelID,stuinfo.nUniqueChannel);
printf("Dev Local szDevID = %s,
the Local szName = %s.\n",stuinfo.szDevID,stuinfo.szName);
DH_REMOTE_DEVICE stuRemoteDevice = stuinfo.stuRemoteDevice;

printf("RemoteDev IP = %s,
RemoteDev Port = %d.\n",stuRemoteDevice.szIp,stuRemoteDevice.nPort);
}
}
}
else
{
printf("CLIENT_MatrixGetCameras Failed, Last Error[%x]\n",
CLIENT_GetLastError());
}
```

2.11.4.4 获取通道名称

```
// 获取通道名称接口 1
// 此处先获取相关配置再修改配置。
// 此处只示例了 command 为 CFG_CMD_CHANNELTITLE 的情况，command 为
CFG_CMD_VIDEOIN 参考此示例开发
    #define MAX_SPACE_NUM 50
    char * szOut = new char[1024* MAX_SPACE_NUM];
    AV_CFG_ChannelName *stOut = new AV_CFG_ChannelName[MAX_SPACE_NUM];
    memset(szOut,0,1024*MAX_SPACE_NUM);//初始化
    memset(stOut,0,sizeof(*stOut)*MAX_SPACE_NUM);
    for(int i=0; i<MAX_SPACE_NUM;i++)
    {
        stOut[i].nStructSize = sizeof(stOut[i]);//结构体大小赋值
    }

    int nError = 0;
    BOOL bRet = 0;
    //获取配置
    bRet = CLIENT_GetNewDevConfig(lLoginHandle,CFG_CMD_CHANNELTITLE,-1,
        szOut,MAX_SPACE_NUM*1024,&nError,3000);
```

```
    if(bRet)
    {
        int nRetLen = 0;
        bRet= CLIENT_ParseData(CFG_CMD_CHANNELTITLE,szOut,stOut,
            MAX_SPACE_NUM*sizeof(*stOut),&nRetLen);
        if(bRet)
        {
            int nRetNum = nRetLen/sizeof(*stOut);
            for(int n=0; n<nRetNum;n++)
            {
                AV_CFG_ChannelName stTitle = stOut[n];
                printf("channel %d title is %s\n",n,stTitle.szName);
            }
            printf("get succeed\n");
        }
        else
        {
            printf("CLIENT_ParseData Failed, Last Error[%x]\n",
                CLIENT_GetLastError());
        }
    }
    else
    {
        printf("CLIENT_GetNewDevConfig Failed, Last Error[%x]\n",
            CLIENT_GetLastError());
    }

    // 修改配置
    printf("修改第一个通道的通道名称为：AAA\n");
    char szName1[CFG_MAX_CHANNEL_NAME_LEN] = {"AAA"};
    memcpy(stOut[0].szName, szName1, CFG_MAX_CHANNEL_NAME_LEN);

    bRet = CLIENT_PacketData(CFG_CMD_CHANNELTITLE,stOut,
sizeof(AV_CFG_ChannelName),szOut,1024*MAX_SPACE_NUM);
    if(bRet)
    {
        bRet = CLIENT_SetNewDevConfig(lLoginHandle,CFG_CMD_CHANNELTITLE,0,
            szOut,1024*MAX_SPACE_NUM,NULL,NULL,3000);
        //修改哪个通道的通道名称，通道号填几，通道号从 0 开始。此处修改第一个通道的名
称，所以通道号填 0
        if(bRet)
        {
            printf("set succeed\n");
        }
        else
        {
```

```
                printf("CLIENT_SetNewDevConfig Failed, Last Error[%x]\n",
                    CLIENT_GetLastError());
            }
        }
        else
        {
            printf("CLIENT_PacketData Failed, Last Error[%x]\n",
                CLIENT_GetLastError());
                }

// 获取通道名称接口 2
// 此接口适用于 DVR 模拟设备，由于市面 DVR 设备减少，此接口使用时需注意
char * szOut = new char[32*18];
int nChannelCount = 0;
BOOL bRet = CLIENT_QueryChannelName(lLoginHandle,szOut,32*16,&nChannelCount,3000);
if(bRet)
{
    for(int i = 0; i<nChannelCount; i++)
{
cout << "channel:" << i <<","<<"通道名称为:"<<szOut<<endl;
szOut = szOut +32;
}
}
else
{
    printf("CLIENT_QueryChannelName Failed, Last Error[%x]\n",
                CLIENT_GetLastError());
}
```

第 3 章 接口函数

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
返回值	无	

选项	说明
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

3.3.2 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 回放

3.5.1 按时间方式回放 CLIENT_PlayBackByTimeEx2

表3-22 按时间方式回放 CLIENT_PlayBackByTimeEx2

选项	说明	
描述	按时间方式回放	
函数	LLONG CLIENT_PlayBackByTimeEx2(LLONG lLoginID, int nChannelID, NET_IN_PLAY_BACK_BY_TIME_INFO *pstNetIn, NET_OUT_PLAY_BACK_BY_TIME_INFO *pstNetOut);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nChannelID	设备通道号
	[in] pstNetIn	查询输入条件
	[out] pstNetOut	查询输出信息

选项	说明
返回值	● 成功返回网络回放 ID ● 失败返回 0
说明	● NET_IN_PLAY_BACK_BY_TIME_INFO 中回调函数声明 fDataCallBack 和 ● fDownLoadPosCallBack 请参见“第 4 章回调函数” ● 参数 hWnd 和 fDownLoadDataCallBack 不能同时为 NULL，否则接口调用会返回失败

3.5.2 设置工作模式 CLIENT_SetDeviceMode

表3-23 设置工作模式 CLIENT_SetDeviceMode

选项	说明	
描述	设置工作模式	
函数	BOOL CLIENT_SetDeviceMode(LLONG 1LoginID, EM_USEDEV_MODE emType, void *pValue);	
参数	[in] 1LoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] emType	工作模式枚举
	[in] pValue	相应工作模式对应的结构体
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

表3-24 工作模式枚举及结构体对照表

emType 枚举	含义	结构体
DH_RECORD_STREAM_TYPE	设置待查询及按时间回放的录像码流类型，可认为是 int 类型 ● 0：主辅码流 ● 1：主码流 ● 2：辅码流	无
DH_RECORD_TYPE	设置按时间录像回放及下载的录像文件类型	NET_RECORD_TYPE

3.5.3 停止录像回放 CLIENT_StopPlayBack

表3-25 停止录像回放 CLIENT_StopPlayBack

选项	说明	
描述	停止录像回放	
函数	BOOL CLIENT_StopPlayBack(LLONG 1PlayHandle);	
参数	[in] 1PlayHandle	回放接口返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	

选项	说明
说明	无

3.5.4 获取回放 OSD 时间 CLIENT_GetPlayBackOsdTime

表3-26 获取回放 OSD 时间 CLIENT_GetPlayBackOsdTime

选项	说明	
描述	获取回放 OSD 时间	
函数	BOOL CLIENT_GetPlayBackOsdTime(LLONG lPlayHandle, LPNET_TIME lpOsdTime, LPNET_TIME lpStartTime, LPNET_TIME lpEndTime);	
参数	[in] lPlayHandle	回放接口返回值
	[out] lpOsdTime	OSD 的时间
	[out] lpOsdTime	当前回放文件的开始时间
	[out] lpEndTime	当前回放文件的结束时间
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.5.5 暂停或恢复录像回放 CLIENT_PausePlayBack

表3-27 暂停或恢复录像回放 CLIENT_PausePlayBack

选项	说明	
描述	暂停或恢复录像回放	
函数	BOOL CLIENT_PausePlayBack(LLONG lPlayHandle, BOOL bPause);	
参数	[in] lPlayHandle	回放接口返回值
	[out] bPause	网络回放暂停与恢复播放参数 ● 1：暂停 ● 0：恢复
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	对已经打开的播放进行暂停和恢复控制	

3.6 录像下载

3.6.1 查询时间段内的所有录像文件 CLIENT_QueryRecordFile

表3-28 查询时间段内的所有录像文件 CLIENT_QueryRecordFile

选项	说明	
描述	查询时间段内的所有录像文件	
函数	BOOL CLIENT_QueryRecordFile(LLONG lLoginID, int nChannelId, int nRecordFileType, LPNET_TIME tmStart, LPNET_TIME tmEnd, char* pchCardid, LPNET_RECORDFILE_INFO nriFileinfo, int maxlen, int *filecount, int waittime=1000, BOOL bTime = FALSE);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nChannelId	设备通道号，从 0 开始
	[in] nRecordFileType	录像文件类型
	[in] tmStart	录像开始时间
	[in] tmEnd	录像结束时间
	[in] pchCardid	卡号
	[out] nriFileinfo	返回的录像文件信息，是一个 LPNET_RECORDFILE_INFO 结构数组
	[in] maxlen	nriFileinfo 缓冲的最大长度（单位：字节，建议在（100～200）*sizeof(NET_RECORDFILE_INFO)之间）
	[out] filecount	返回的文件个数，属于输出参数最大只能查到缓冲满为止的录像记录
	[in] waittime	等待时间
	[in] bTime	是否是按时间查询
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	在回放之前需要先调用本接口查询录像记录，当根据输入的时间段查询到的录像记录信息大于定义的缓冲区大小，则只返回缓冲所能存放的录像记录，可以根据需要继续查询	

表3-29 录像文件类型及卡号对照表

数值	录像文件类型	卡号
0	所有录像文件	NULL
1	外部报警	NULL
2	动态检测报警	NULL
3	所有报警	NULL

数值	录像文件类型	卡号
4	卡号查询	卡号
5	组合条件查询	卡号&&交易类型&&交易金额（如希望跳过某字段，则相应位置为空）
6	录像位置与偏移量长度	NULL
8	按卡号查询图片（目前仅 HB-U 和 NVS 特殊型号的设备支持）	卡号
9	查询图片（目前仅 HB-U 和 NVS 特殊型号的设备支持）	NULL
10	按字段查询	FELD1&&FELD2&&FELD3&&（如希望跳过某字段，则相应位置为空）

3.6.2 打开录像查询句柄 CLIENT_FindFile

表3-30 打开录像查询句柄 CLIENT_FindFile

选项	说明	
描述	打开录像查询句柄	
函数	LLONG CLIENT_FindFile(LLONG lLoginID, int nChannelId, int nRecordFileType, char* cardid, LPNET_TIME time_start, LPNET_TIME time_end, BOOL bTime, int waittime);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nChannelId	设备通道号，从 0 开始
	[in] nRecordFileType	录像文件类型（具体类型请参见表 3-29）
	[in]cardid	卡号，只针对卡号查询有效，其他的情况填 NULL
	[in] time_start	查询录像开始时间
	[in] time_end	查询录像结束时间
	[in] bTime	是否按时间查询
	[in] waittime	查询超时时间
返回值	● 成功返回查询句柄 ● 失败返回 0	
说明	无	

3.6.3 查找录像文件 CLIENT_FindNextFile

表3-31 查找录像文件 CLIENT_FindNextFile

选项	说明
描述	查找录像文件

选项	说明	
函数	int CLIENT_FindNextFile(LLONG lFindHandle, LPNET_RECORDFILE_INFO lpFindData);	
参数	[in] lFindHandle	CLIENT_FindFile（打开录像查询句柄）返回值
	[in] lpFindData	录像文件记录缓冲，用于输出已查询的录像文件记录
返回值	● 1：成功取回一条录像记录 ● 0：录像记录已取完 ● -1：参数出错	
说明	调用本接口之前应先调用 CLIENT_FindFile 以打开查询句柄	

3.6.4 关闭录像查询句柄 CLIENT_FindClose

表3-32 关闭录像查询句柄 CLIENT_FindClose

选项	说明	
描述	关闭录像查询句柄	
函数	BOOL CLIENT_FindClose(LLONG lFindHandle);	
参数	[in] lFindHandle	CLIENT_FindFile（打开录像查询句柄）返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	调用 CLIENT_FindFile 打开查询句柄，查询完毕后应调用本函数以关闭查询句柄	

3.6.5 按文件下载录像 CLIENT_DownloadByRecordFileEx

表3-33 按文件下载录像 CLIENT_DownloadByRecordFileEx

选项	说明	
描述	按文件下载录像	
函数	LLONG CLIENT_DownloadByRecordFileEx(LLONG lLoginID, LPNET_RECORDFILE_INFO lpRecordFile, char *sSavedFileName, fDownLoadPosCallBack cbDownLoadPos, LDWORD dwUserData, fDataCallBack fDownLoadDataCallBack, LDWORD dwDataUser, void* pReserved = NULL);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] lpRecordFile	录像文件信息指针
	[in] sSavedFileName	要保存的录像文件名，全路径
	[in] cbDownLoadPos	下载进度回调函数
	[in] dwUserData	下载进度回调用户自定义数据

选项	说明	
	[in] fDownLoadDataCallBack	数据回调函数
	[in] pReserved	保留参数，默认为 NULL
返回值	● 成功返回下载 ID ● 失败返回 0	
说明	● 回调函数声明 fDownLoadPosCallBack 和 fDataCallBack 请参见“第 4 章回调函数” ● sSavedFileName 不为空，录像数据写入到该路径对应的文件 ● fDownLoadDataCallBack 不为空，录像数据通过回调函数返回	

3.6.6 按时间下载录像 CLIENT_DownloadByTimeEx

表3-34 按时间下载录像 CLIENT_DownloadByTimeEx

选项	说明	
描述	按时间下载录像	
函数	LLONG CLIENT_DownloadByTimeEx(LLONG lLoginID, int nChannelId, int nRecordFileType, LPNET_TIME tmStart, LPNET_TIME tmEnd, char *sSavedFileName, fTimeDownLoadPosCallBack cbTimeDownLoadPos, LDWORD dwUserData, fDataCallBack fDownLoadDataCallBack, LDWORD dwDataUser, void* pReserved = NULL);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nChannelId	设备通道号，从 0 开始
	[in] nRecordFileType	文件查询类型 0：所有录像文件 1：外部报警 2：动态检测录像 3：所有报警 4：按卡号查询录像 5：组合条件查询 8：按卡号查询图片 9：查询图片 10：按字段查询
	[in] tmStart	下载起始时间
	[in] tmEnd	下载结束时间
	[in] sSavedFileName	要保存的录像文件名，全路径
	[in]cbTimeDownLoadPos	下载进度回调函数
	[in] dwUserData	下载进度回调用户自定义数据
	[in] fDownLoadDataCallBack	下载数据回调函数

选项	说明	
	[in] dwUserData	下载数据回调用户自定义数据
	[in] pReserved	保留参数，默认为 NULL
返回值	● 成功返回下载 ID ● 失败返回 0	
说明	● NET_IN_PLAY_BACK_BY_TIME_INFO 中回调函数声明 fDataCallBack 和 ● fDownLoadPosCallBack 请参见“第 4 章回调函数” ● sSavedFileName 不为空，录像数据写入到该路径对应的文件 ● fDownLoadDataCallBack 不为空，录像数据通过回调函数返回	

3.6.7 查询录像下载进度 CLIENT_GetDownloadPos

表3-35 查询录像下载进度 CLIENT_GetDownloadPos

选项	说明	
描述	查询录像下载进度	
函数	BOOL CLIENT_GetDownloadPos(LLONG lFileHandle, int *nTotalSize, int *nDownLoadSize);	
参数	[in] lFileHandle	下载接口返回值
	[out] nTotalSize	下载的总长度，单位：KB
	[out] nDownLoadSize	已下载的长度，单位：KB
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	● 获得下载录像的当前位置，可以用于不需要实时显示下载进度的接口，与下载回调函数的功能类似 ● 用于不打算通过回调计算进度，可定时调用本接口获取当前进度	

3.6.8 停止录像下载 CLIENT_StopDownload

表3-36 停止录像下载 CLIENT_StopDownload

选项	说明	
描述	停止录像下载	
函数	BOOL CLIENT_StopDownload(LLONG lFileHandle);	
参数	[in] lFileHandle	下载接口返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	根据实际需要，等文件下载完成后停止下载，也可以下载到一部分停止下载	

3.7 云台控制

3.7.1 云台控制 CLIENT_DHPTZControlEx2

表3-37 云台控制 CLIENT_DHPTZControlEx2

选项	说明	
描述	云台控制	
函数	BOOL CLIENT_DHPTZControlEx2(LLONG lLoginID, int nChannelID, DWORD dwPTZCommand, LONG lParam1, LONG lParam2, LONG lParam3, BOOL dwStop , void* param4);	
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] dwPTZCommand	控制命令类型
	[in] lParam1	参数 1
	[in] lParam2	参数 2
	[in] lParam3	参数 3
	[in] dwStop	停止标志。对云台八方向操作及镜头操作命令有效，进行其他操作时，本参数应填充 FALSE
	[in] param4	支持扩展控制命令参数，主要支持如下几种控制命令： DH_EXTPTZ_MOVE_ABSOLUTELY DH_EXTPTZ_MOVE_CONTINUOUSLY DH_EXTPTZ_GOTOPRESET DH_EXTPTZ_SET_VIEW_RANGE DH_EXTPTZ_FOCUS_ABSOLUTELY DH_EXTPTZ_HORSECTORSCAN DH_EXTPTZ_VERSECTORSCAN DH_EXTPTZ_SET_FISHEYE_EPTZ
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	dwPTZCommand 与 Param1、Param2 和 Param3 的关系，请参见表 3-38	

dwPTZCommand 与 Param1、Param2 和 Param3 的关系，请参见表 3-38。

表3-38 Param1、Param2 和 Param3 的关系

dwPTZCommand 宏定义	功能描述	param1	param2	param3
DH_PTZ_UP_CONTROL	上	-	垂直速度（1-8）	-
DH_PTZ_DOWN_CONTROL	下	-	垂直速度（1-8）	-

dwPTZCommand 宏定义	功能描述	param1	param2	param3
DH_PTZ_LEFT_CONTROL	左	-	水平速度（1-8）	-
DH_PTZ_RIGHT_CONTROL	右	-	水平速度（1-8）	-
DH_PTZ_ZOOM_ADD_CONTROL	变倍+	-	倍速	-
DH_PTZ_ZOOM_DEC_CONTROL	变倍-	-	倍速	-
DH_PTZ_FOCUS_ADD_CONTROL	调焦+	-	倍速	-
DH_PTZ_FOCUS_DEC_CONTROL	调焦-	-	倍速	-
DH_PTZ_APERTURE_ADD_CONTROL	光圈+	-	倍速	-
DH_PTZ_APERTURE_DEC_CONTROL	光圈-	-	倍速	-
DH_PTZ_POINT_MOVE_CONTROL	转至预置点	-	预置点值	-
DH_PTZ_POINT_SET_CONTROL	设置	-	预置点值	-
DH_PTZ_POINT_DEL_CONTROL	删除	-	预置点值	-
DH_PTZ_POINT_LOOP_CONTROL	点间轮巡	巡航线路	-	76:开始 99:自动 96:停止
DH_PTZ_LAMP_CONTROL	灯光雨刷	0x01 开启,x00 关闭	-	-
DH_EXTPTZ_LEFTTOP	左上	垂直速度（1-8）	水平速度（1-8）	-
DH_EXTPTZ_RIGHTTOP	右上	垂直速度（1-8）	水平速度（1-8）	-
DH_EXTPTZ_LEFTDOWN	左下	垂直速度（1-8）	水平速度（1-8）	-
DH_EXTPTZ_RIGHTDOWN	右下	垂直速度（1-8）	水平速度（1-8）	-
DH_EXTPTZ_ADDTOLOOP	加入预置点到巡航	巡航线路	预置点值	-
DH_EXTPTZ_DELFROMLOOP	删除巡航中预置点	巡航线路	预置点值	-
DH_EXTPTZ_CLOSELOOP	清除巡航	巡航线路	-	-
DH_EXTPTZ_STARTPANCRUISE	开始水平旋转	-	-	-
DH_EXTPTZ_STOPPANCRUISE	停止水平旋转	-	-	-
DH_EXTPTZ_SETLEFTBORDER	设置左边界	-	-	-
DH_EXTPTZ_RIGHTBORDER	设置右边界	-	-	-
DH_EXTPTZ_STARTLINESCAN	开始线扫	-	-	-
DH_EXTPTZ_CLOSELINESCAN	停止线扫	-	-	-
DH_EXTPTZ_SETMODESTART	设置模式开始	模式线路	-	-
DH_EXTPTZ_SETMODESTOP	设置模式结束	模式线路	-	-
DH_EXTPTZ_RUNMODE	运行模式	模式线路	-	-

dwPTZCommand 宏定义	功能描述	param1	param2	param3
DH_EXTPTZ_STOPMODE	停止模式	模式线路	-	-
DH_EXTPTZ_DELETEMODE	清除模式	模式线路	-	-
DH_EXTPTZ_REVERSECOMM	翻转命令	-	-	-
DH_EXTPTZ_FASTGOTO	快速定位	水平坐标（0-8192）	垂直坐标（0-8192）	变倍（4）
DH_EXTPTZ_AUXIOPEN	辅助开关开	辅助点	-	-
DH_EXTPTZ_AUXICLOSE	辅助开关关	辅助点	-	-
DH_EXTPTZ_OPENMENU	打开球机菜单	-	-	-
DH_EXTPTZ_CLOSEMENU	关闭菜单	-	-	-
DH_EXTPTZ_MENUOK	菜单确定	-	-	-
DH_EXTPTZ_MENUCANCEL	菜单取消	-	-	-
DH_EXTPTZ_MENUUP	菜单上	-	-	-
DH_EXTPTZ_MENUDOWN	菜单下	-	-	-
DH_EXTPTZ_MENULEFT	菜单左	-	-	-
DH_EXTPTZ_MENURIGHT	菜单右	-	-	-
DH_EXTPTZ_ALARMHANDLE	报警联动云台	报警输入通道	报警联动类型： 预置点 线扫 巡航	联动值，如预置点号
DH_EXTPTZ_MATRIXSWITCH	矩阵切换	预览器号（视频输出号）	视频输入号	矩阵号
DH_EXTPTZ_LIGHTCONTROL	灯光控制器	参考 DH_PTZ_LAMP_CONTROL	-	-
DH_EXTPTZ_EXACTGOTO	三维精确定位	水平角度（0～3600）	垂直坐标（0～900）	变倍（1～128）
DH_EXTPTZ_RESETZERO	三维定位重设零位		-	-
DH_EXTPTZ_UP_TELE	上+TELE	速度（1-8）	-	-
DH_EXTPTZ_DOWN_TELE	下+TELE	速度（1-8）	-	-
DH_EXTPTZ_LEFT_TELE	左+TELE	速度（1-8）	-	-
DH_EXTPTZ_RIGHT_TELE	右+TELE	速度（1-8）	-	-
DH_EXTPTZ_LEFTUP_TELE	左上+TELE	速度（1-8）	-	-
DH_EXTPTZ_LEFTDOWN_TELE	左下+TELE	速度（1-8）	-	-
DH_EXTPTZ_TIGHTUP_TELE	右上+TELE	速度（1-8）	-	-
DH_EXTPTZ_RIGHTDOWN_TELE	右下+TELE	速度（1-8）	-	-
DH_EXTPTZ_UP_WIDE	上+WIDE	速度（1-8）	-	-
DH_EXTPTZ_DOWN_WIDE	下+WIDE	速度（1-8）	-	-
DH_EXTPTZ_LEFT_WIDE	左+WIDE	速度（1-8）	-	-
DH_EXTPTZ_RIGHT_WIDE	右+WIDE	速度（1-8）	-	-

dwPTZCommand 宏定义	功能描述	param1	param2	param3
DH_EXTPTZ_LEFTUP_WIDE	左上+WIDE	速度（1-8）	-	-
DH_EXTPTZ_LEFTDOWN_WIDE	左下+WIDE	速度（1-8）	-	-
DH_EXTPTZ_RIGHTUP_WIDE	右上+WIDE	速度（1-8）	-	-
DH_EXTPTZ_RIGHTDOWN_WIDE	右下+WIDE	速度（1-8）	-	-

3.8 语音对讲

3.8.1 开启对讲 CLIENT_StartTalkEx

表3-39 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	

3.8.2 关闭对讲 CLIENT_StopTalkEx

表3-40 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	

3.8.3 关闭录音 CLIENT_RecordStopEx

表3-41 关闭录音 CLIENT_RecordStopEx

选项	说明
描述	关闭本地录音

选项	说明	
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	

3.8.4 开启录音 CLIENT_RecordStartEx

表3-42 开启录音 CLIENT_RecordStartEx

选项	说明	
描述	开启本地录音	
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	

3.8.5 发送语音 CLIENT_TalkSendData

表3-43 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	

3.8.6 解码语音 CLIENT_AudioDecEx

表3-44 解码语音 CLIENT_AudioDecEx

选项	说明
描述	解码音频数据

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	

3.9 视频抓图

3.9.1 设备抓图 CLIENT_SnapPictureToFile

表3-45 设备抓图 CLIENT_SnapPictureToFile

选项	说明	
描述	设备抓图	
函数	BOOL CLIENT_SnapPictureToFile(LLONG lLoginID, const NET_IN_SNAP_PIC_TO_FILE_PARAM* pInParam, NET_OUT_SNAP_PIC_TO_FILE_PARAM* pOutParam, int nWaitTime);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in] pInParam	输入参数
	[out] pOutParam	输出参数
	[in] nWaitTime	超时时间，单位：毫秒
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	● 同步接口，由设备抓取图片，通过网络传给用户 ● 需要设备支持该功能	

3.9.2 抓图 CLIENT_CapturePictureEx

表3-46 抓图 CLIENT_CapturePictureEx

选项	说明	
描述	抓图	
函数	BOOL CLIENT_CapturePictureEx(LLONG hPlayHandle, const char *pchPicFileName, NET_CAPTURE_FORMATS eFormat);	
参数	[in] hPlayHandle	CLIENT_RealPlayEx 的返回值

选项	说明	
	[in] pchPicFileName	需要保存的文件路径，必须为绝对路径
	[in] eFormat	图片格式
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	● 同步接口，将图片数据直接写成文件 ● 图片从设备发送过来的实时预览数据流中抓取	

3.10 报警上报

3.10.1 设置报警回调函数 CLIENT_SetDVRMessCallBack

表3-47 设置报警回调函数 CLIENT_SetDVRMessCallBack

选项	说明	
描述	设置报警回调函数	
函数	void CLIENT_SetDVRMessCallBack(fMessCallBack cbMessage, LDWORD dwUser);	
参数	[in] cbMessage	● 消息回调函数，可以回调设备的状态，如报警状态 ● 当设置为 0 时表示禁止回调
	[in] dwUser	用户自定义数据
返回值	无	
说明	● 设置设备消息回调函数，用来得到设备当前状态信息，与调用顺序无关，SDK 默认不回调 ● 此回调函数 fMessCallBack 必须先调用报警消息订阅接口 CLIENT_StartListenEx 才生效	

3.10.2 订阅报警 CLIENT_StartListenEx

表3-48 订阅报警 CLIENT_StartListenEx

选项	说明	
描述	订阅报警	
函数	BOOL CLIENT_StartListenEx(LLONG lLoginID);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	订阅设备消息，得到的消息从 CLIENT_SetDVRMessCallBack 的设置值回调出来	

3.10.3 停止订阅报警 CLIENT_StopListen

表3-49 停止订阅报警 CLIENT_StopListen

选项	说明	
描述	停止订阅报警	
函数	BOOL CLIENT_StopListen(LLONG lLoginID);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.11 存储

3.11.1 直接获取远程设备连接状态 CLIENT_QueryDevState

表3-50 直接获取远程设备连接状态 CLIENT_QueryDevState

选项	说明	
描述	直接获取远程设备连接状态	
函数	BOOL CLIENT_QueryDevState(LLONG lLoginID, int nType, char *pBuf, int nBufLen, int *pRetLen, int waittime=1000);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nType	查询信息类型，当获取远程设备连接状态时 type 为 DH_DEVSTATE_VIRTUALCAMERA
	[out] pBuf	用于接收查询返回的数据的缓存，对应 DHDEV_VIRTUALCAMERA_STATE_INFO 结构体
	[in] nBufLen	缓存长度，单位：字节
	[out] pRetLen	实际返回的数据长度，单位：字节
	[in] waittime	查询状态等待时间，默认 1000ms
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.11.2 查询设备信息 CLIENT_QueryDevInfo

表3-51 查询设备信息 CLIENT_QueryDevInfo

选项	说明	
描述	查询设备信息	
函数	BOOL CALL_METHOD CLIENT_QueryDevInfo(LLONG lLoginID, int nQueryType, void* pInBuf, void* pOutBuf, void *pReserved = NULL, int nWaitTime = 1000);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] nQueryType	查询信息类型，当获取远程设备连接状态时 nQueryType 为 NET_QUERY_GET_CAMERA_STATE
	[in] pInBuf	输入缓存，当获取远程设备连接状态时，对应 NET_IN_GET_CAMERA_STATEINFO 结构体
	[out] pOutBuf	输出缓存，当获取远程设备连接状态时，对应 NET_OUT_GET_CAMERA_STATEINFO 结构体
	[in] pReserved	保留
	[in] nWaitTime	查询状态等待时间，默认 1000ms
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.11.3 订阅远程设备状态 CLIENT_AttachCameraState

表3-52 订阅远程设备状态 CLIENT_AttachCameraState

选项	说明	
描述	订阅远程设备状态	
函数	LLONG CLIENT_AttachCameraState(LLONG lLoginID, const NET_IN_CAMERASTATE* pstInParam, NET_OUT_CAMERASTATE *pstOutParam, int nWaitTime = 3000);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] pstInParam	订阅输入参数
	[out] pstOutParam	订阅输出参数
	[in] nWaitTime	查询状态等待时间，默认 3000ms
返回值	● 成功返回非 0 ● 失败返回 0	
说明	输入参数中的状态回调函数原型 fCameraStateCallBack 请参见“第 4 章回调函数”	

3.11.4 停止订阅远程设备状态 CLIENT_DetachCameraState

表3-53 停止订阅远程设备状态 CLIENT_DetachCameraState

选项	说明	
描述	停止订阅远程设备状态	
函　数	BOOL CLIENT_DetachCameraState(LLONG lAttachHandle);	
参　数	[in] lAttachHandle	订阅远程设备状态接口返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说　明	无	

3.11.5 获取远程设备信息 CLIENT_MatrixGetCameras

表3-54 获取远程设备信息 CLIENT_MatrixGetCameras

选项	说明	
描述	获取远程设备信息	
函数	BOOL CLIENT_MatrixGetCameras(LLONG lLoginID, const DH_IN_MATRIX_GET_CAMERAS* pInParam, DH_OUT_MATRIX_GET_CAMERAS* pOutParam, int nWaitTime = 1000);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] pInParam	输入参数
	[out] pOutParam	输出参数
	[in] nWaitTime	查询状态等待时间，默认 1000ms
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.11.6 获取通道名称 CLIENT_QueryChannelName

表3-55 获取通道名称 CLIENT_QueryChannelName

选项	说明	
描述	获取通道名称	
函数	BOOL CLIENT_QueryChannelName(LLONG lLoginID, char *pChannelName, int maxlen, int *nChannelCount, int waittime=1000);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值

选项	说明	
	[out] pChannelName	通道名缓冲区（一般每个通道名是 32 字节长，这里需分配≥16*32 字节长度的缓冲）
	[in] maxlen	缓冲区长度，单位：字节
	[out] nChannelCount	总共通道数
	[in] waittime	查询等待时间，默认 1000ms
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.11.7 获取通道名称 CLIENT_GetNewDevConfig

表3-56 获取通道名称 CLIENT_GetNewDevConfig

选项	说明	
描述	获取通道名称	
函数	BOOL CLIENT_GetNewDevConfig(LLONG lLoginID, char* szCommand, int nChannelID, char* szOutBuffer, DWORD dwOutBufferSize, lnt *error, int waittime=500);	
参数	[in] lLoginID	CLIENT_LoginWithHighLevelSecurity 返回值
	[in] szCommand	命令参数，获取通道名称，szCommand 为 CFG_CMD_VIDEOIN 或 CFG_CMD_CHANNELTITLE
	[in] nChannelID	设备通道号，从 0 开始
	[out] szOutBuffer	输出缓冲
	[out] dwOutBufferSize	输出缓冲大小
	[out] error	错误码 ● 0：成功 ● 1：失败 ● 2：数据不合法 ● 3：暂时无法设置 ● 4：没有权限
	[in] waittime	等待超时时间，默认 500ms
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

3.11.8 解析数据 CLIENT_ParseData

表3-57 解析数据 CLIENT_ParseData

选项	说明	
描述	解析数据	
函数	BOOL CLIENT_ParseData(char* szCommand, char* szInBuffer, LPVOID lpOutBuffer, DWORD dwOutBufferSize, void* pReserved);	
参数	[in] szCommand	命令参数，与 CLIENT_GetNewDevConfig 接口 szCommand 相同
	[in] szInBuffer	输入缓存，与 CLIENT_GetNewDevConfig 接口的 szOutBuffer 相同
	[out] lpOutBuffer	输出缓存，与 CLIENT_GetNewDevConfig 接口的 szCommand 对应，分别对应结构体 CFG_VIDEO_IN_INFO 和 AV_CFG_ChannelName
	[in] dwOutBufferSize	输出缓冲大小
	[in] pReserved	保留
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

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	异步搜索设备回调函数	
函数	typedef void(CALLBACK * fSearchDevicesCBEx)(LLONG lSearchHandle, DEVICE_NET_INFO_EX2 *pDevNetInfo, void* pUserData);	
参数	[in] lSearchHandle	CLIENT_StartSearchDevicesEx 接口返回的搜索句柄
	[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 实时预览数据回调函数 fRealDataCallBackEx2

表4-5 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 音频数据回调函数 pfAudioDataCallBack

表4-6 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

选项	说明	
	[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	

4.7 回放及按文件下载进度回调函数 fDownLoadPosCallBack

表4-7 回放及按文件下载进度回调函数 fDownLoadPosCallBack

选项	说明	
描述	回放及按文件下载进度回调函数	
函数	typedef void (CALLBACK *fDownLoadPosCallBack)(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, LDWORD dwUser);	
参数	[out]lPlayHandle	回放或下载接口返回值
	[out]dwTotalSize	总大小，单位：KB
	[out]dwDownLoadSize	已下载的大小，单位：KB ● -1：本次回放结束 ● -2：写文件失败
	[out]dwUser	用户数据
返回值	无	
说明	无	

4.8 回放及下载数据回调函数 fDataCallBack

表4-8 回放及下载数据回调函数 fDataCallBack

选项	说明	
描述	回放及下载数据回调函数	
函数	typedef int (CALLBACK *fDataCallBack)(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser);	
参数	[out]lPlayHandle	回放或下载接口返回值
	[out] dwDataType	这里为 0(原始数据)
	[out] pBuffer	数据缓冲
	[out] dwBufSize	缓冲长度，单位：字节

选项	说明	
	[out] dwUser	用户数据
返回值	● 0：表示本次回调失败，下次回调会返回相同的数据 ● 1：表示本次回调成功，下次回调会返回后续的数据	
说明	● 在 CLIENT_PlayBackByTimeEx 等录像回放接口中设置该回调函数 ● 设置该回调函数时，若对应的 hWnd 参数不为 NULL，则不管回调函数返回值为多少都认为回调成功，下次回调会返回后续的数据 ● 用户在该回调函数中可通过参数 lRealHandle 来唯一识别是哪次拉流对应的回调数据	

4.9 按时间下载回调函数 fTimeDownLoadPosCallBack

表4-9 按时间下载回调函数 fTimeDownLoadPosCallBack

选项	说明	
描述	按时间下载回调函数	
函数	typedef void (CALLBACK *fTimeDownLoadPosCallBack)(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, int index, NET_RECORDFILE_INFO recordfileinfo, LDWORD dwUser);	
参数	[out]lPlayHandle	下载接口返回值
	[out] dwTotalSize	指本次播放总大小，单位：KB
	[out]dwDownLoadSize	指已经播放的大小，单位：KB ● -1：表示本次回放结束 ● -2：表示写文件失败
	[out] index	索引
	[out] recordfileinfo	录像文件信息
	[out] dwUser	用户数据
返回值	无	
说明	无	

4.10 报警回调函数 fMessCallBack

表4-10 报警回调函数 fMessCallBack

选项	说明
描述	报警回调函数

选项	说明	
函数	BOOL (CALLBACK *fMessCallBack)(LONG lCommand, LLONG lLoginID, char *pBuf, DWORD dwBufLen, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
参数	[out] lCommand	报警类型，具体请参见表 4-11
	[out] lLoginID	登录接口返回值
	[out] pBuf	接收报警数据的缓存，根据调用的侦听接口和 lCommand 值不同，填充的数据不同
	[out] dwBufLen	pBuf 的长度，单位：字节
	[out] pchDVRIP	设备 IP
	[out] nDVRPort	端口
	[out] dwUser	用户自定义数据
返回值	无意义	
说明	一般在应用程序初始化时调用设置回调，在回调函数中根据不同的设备 ID 和命令值做出不同的处理	

表4-11 报警类型说明

报警类型	报警类型名称	pBuf
DH_MOTION_ALARM_EX	动态检测报警	数据字节数与设备视频通道个数相同，每个字节表示一个视频通道的动态检测报警状态，1 为有报警，0 为无报警
DH_ALARM_STORAGE_FAILURE	硬盘故障报警	ALARM_STORAGE_FAILURE 数组
EVENT_ALARM_VIDEOLOSS	视频丢失报警	无
DH_ALARM_FRONTDISCONNECT	IPC 断网报警	ALARM_FRONTDISCONNET_INFO
DH_ALARM_ALARM_EX	外部报警	数据字节数与设备报警通道个数相同，每个字节表示一个报警通道的报警状态，1 为有报警，0 为无报警

4.11 远程设备状态回调函数 fCameraStateCallBack

表4-12 远程设备状态回调函数 fCameraStateCallBack

选项	说明
描述	远程设备状态回调函数

选项	说明	
函数	void (CALLBACK *fCameraStateCallBack) (LLONG lLoginID, LLONG lAttachHandle, const NET_CB_CAMERASTATE *pBuf, int nBufLen, LDWORD dwUser);	
参数	[out] lLoginID	登录接口返回值
	[out] lAttachHandle	订阅接口返回值
	[out] pBuf	前端设备状态
	[out] nBufLen	返回数据长度
	[out] dwUser	用户自定义数据
返回值	无	
说明	订阅远程设备状态后，如果前端设备状态发生变化时，会上报发生变化的设备信息	

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。