NetSDK 编程指导手册

（智能楼宇分册）

V1.0.1

法律声明

商标声明

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

责任声明

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

关于本文档

- 产品请以实物为准，本文档仅供参考。
- 本文档供多个型号产品做参考，每个产品的具体操作不一一例举，请用户根据实际产品自行对照操作。
- 如不按照本文档中的指导进行操作，因此而造成的任何损失由使用方自己承担。
- 如获取到的 PDF 文档无法打开，请将阅读工具升级到最新版本或使用其他主流阅读工具。
- 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。产品部分功能在更新前后可能存在细微差异。
- 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。

前言

概述

欢迎使用 NetSDK（以下简称 SDK）编程指导手册。

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

本文档描述了智能楼宇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、结构体等请参考《网络 SDK 开发手册》。

本文档提供的示例代码仅为演示接口调用方法，不保证能直接拷贝编译。

读者对象

使用 SDK 的软件开发工程师、项目经理和产品经理。

符号约定

在本文档中可能出现下列标志，代表的含义如下。

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

修订记录

编号	版本号	修订内容	发布日期
1	V1.0.0	首次发布	2017.12
2	V1.0.0	删除表 1-1 中的一部分内容。	2019.1

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
防区	报警输入通道可接收外部探测的信号，每一路报警输入通道即成为一个防区。
布防和撤防	● 布防：设备已布防的防区处于警戒状态，接收外部信号、处理、记录和转发。 ● 撤防：设备已撤防的防区不会接收外部报警信号、也不会处理、记录和转发。
旁路	在设备处于布防时，某个防区对外部探测器仍然监听，也会做记录，但不会转发给用户。在设备撤防后，原来旁路的防区会转变成正常状态，当再次布防时，该防区布防成功。
消警	设备发出报警后，通常会执行一些联动操作，比如蜂鸣、报警键盘提示等，这些操作往往会持续一段时间，消警是使这些联动操作提前结束。
即时防区	在设备布防状态下，该防区触发报警时，立刻记录并转发报警信号。
延时防区	防区类型为延时防区时，可设置进入延时或者退出延时。 进入延时，即用户在延时时间内从非布防区域进入布防区域时产生报警，但不联动报警，延时时间结束后，如果没撤防，此时会联动报警；如果撤防了，就不会联动报警。设置退出延时后，设备会在退出延时结束后再进入布防状态。
24 小时防区	一经配置成 24 小时防区，立即生效，且布撤防动作对其无效，可应用于火警等场景。
情景模式	报警主机现有情景模式两种，分别为“外出模式”和“在家模式”，每个模式会有相关的配置，用户选择哪种情景模式，即可使相应配置生效。
在家模式/外出模式	当情景模式切换到“在家模式”或“外出模式”时，会使该情景模式下预设的防区布防，其余防区变为旁路状态。
隔离	一种对入侵报警探测回路的设置，处于此状态的入侵报警探测回路不能通告报警。此状态一直保持到人为复位。
模拟量报警通道(模拟量防区)	设备有多个报警输入通道，接收外部探测的信号。通道类型为模拟量时，则称为模拟量报警通道，可连接模拟量探测器，用于采集模拟量数据。
胁迫卡	门禁卡的一种，用户被胁迫时使用胁迫卡开门，门禁系统识别出是胁迫卡刷卡，并发出报警信号。

目录

1 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：通用功能、报警主机、门禁等功能。

根据环境不同，开发包包含的文件会不同，具体如下所示。

- Windows 开发包所包含的文件，请参见表 1-1。

表1-1 Windows 开发包包括的文件

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

- Linux 开发包所包含的文件，请参见表 1-2。

表1-2 Linux 开发包包括的文件

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

说明

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

1.2 适用性

1.2.1 适用系统

- 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。

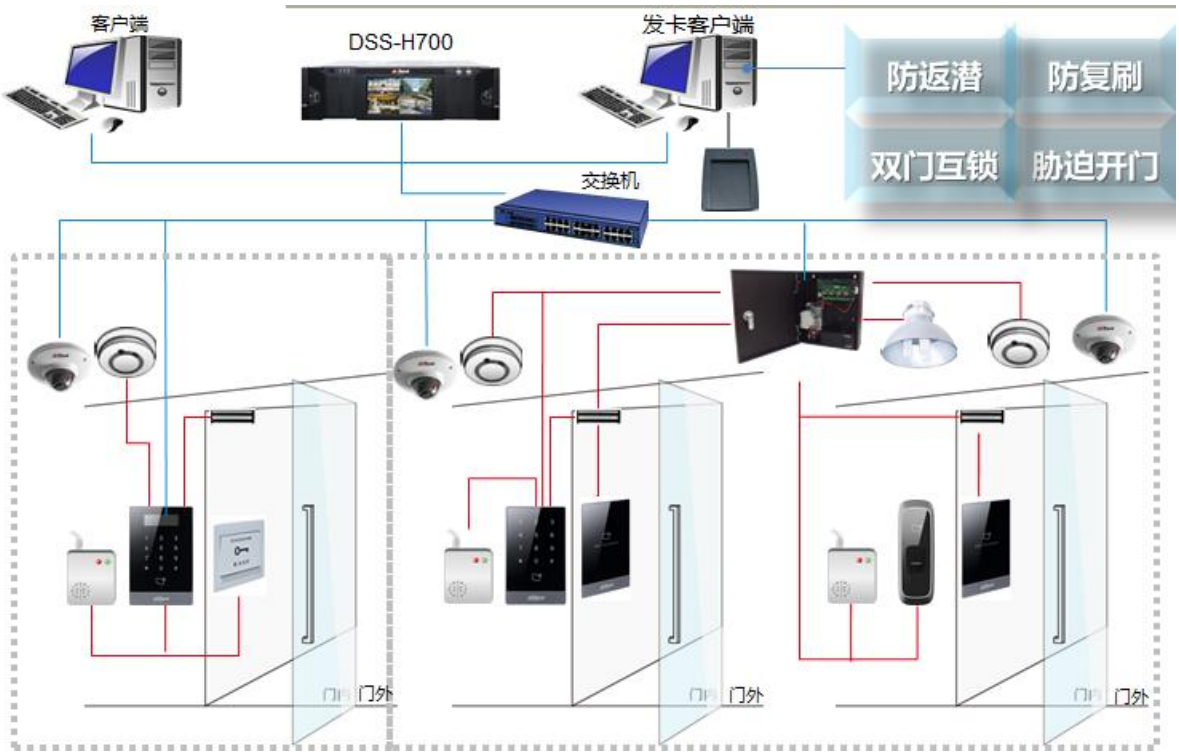
1.2.2 支持设备

- 门禁设备
  DH-BSC1221A 、 DH-BSC1201B 、 DH-BSC1201C 、 DH-BSC1201C-S 、 DH-BSC1204B 、 DH-BSC1204C 、 DH-BSC1204C-S 、 DH-BSR1100A 、 DH-BSR1101A 、 DH-BSR1100B 、 DH-BSR1102A、DH-BSM100B、DH-BSM102A、AL-280(LED)、AL-500(LED)、AL-280L、AL-280PU、AL-280Z、AL-500L、AL-500PU、AL-500Z、AL-100、FS-078、DH-SL5101CG、DH-SL5101CR、DH-SL5101CS、DH-SL5102RR、DH-SL5102RS、DH-SL5102RG
- 可视对讲设备
  VTO1210B-X、VTO1220B、VTO1210C-X、VTO9231D、VTO9241D、VTO6210B、VTO2000A、VTO6100C、VTO2111D、VTH1550CH、VTH1510CH、VTH1510A、VTH5221D、VTH5241D、VTT201、VTT2610C、VTA8111A、VTS8240B、VTS8420B、VTS5420B、VTS1500A
- 报警主机
  ARC2008C 、 ARC2008C-G 、 DH-ARC2008C 、 DH-ARC2008C-G 、 DHI-ARC2008C 、 DHI-ARC2008C-G、OEM-ARC2008C、OEM-ARC2008C-G、ARC2016C、ARC2016C-G、ARC2016C-G、DH-ARC2016C、DH-ARC2016C-G、DHI-ARC2016C、DHI-ARC2016C-G、OEM-ARC2016C 、 OEM-ARC2016C-G 、 ARC9016C 、 ARC9016C-G 、 DH-ARC9016C 、 DH-ARC9016C-G、DHI-ARC9016C、DHI-ARC9016C-G、OEM-ARC9016C、OEM-ARC9016C-G、ARC5808C-C 、 ARC5808C 、 DH-ARC5808C-E 、 DH-ARC5808C-C 、 DH-ARC5808C 、 DHI-ARC5808C 、 DHI-ARC5808C-C 、 OEM-ARC5808C-E 、 OEM-ARC5808C-C 、 OEM-ARC5808C 、 ARC5408C-C 、 ARC5408C 、 DH-ARC5408C-E 、 DH-ARC5408C-C 、 DH-ARC5408C、DHI-ARC5408C、DHI-ARC5408C-C、OEM-ARC5408C-E、OEM-ARC5408C-C、OEM-ARC5408C

1.3 应用场景

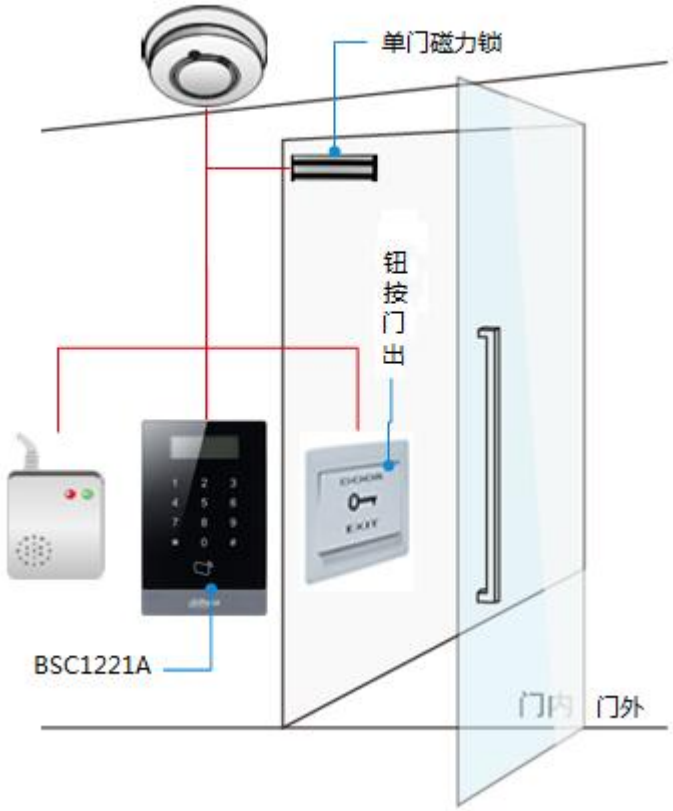
- 典型场景

图1-1 典型场景



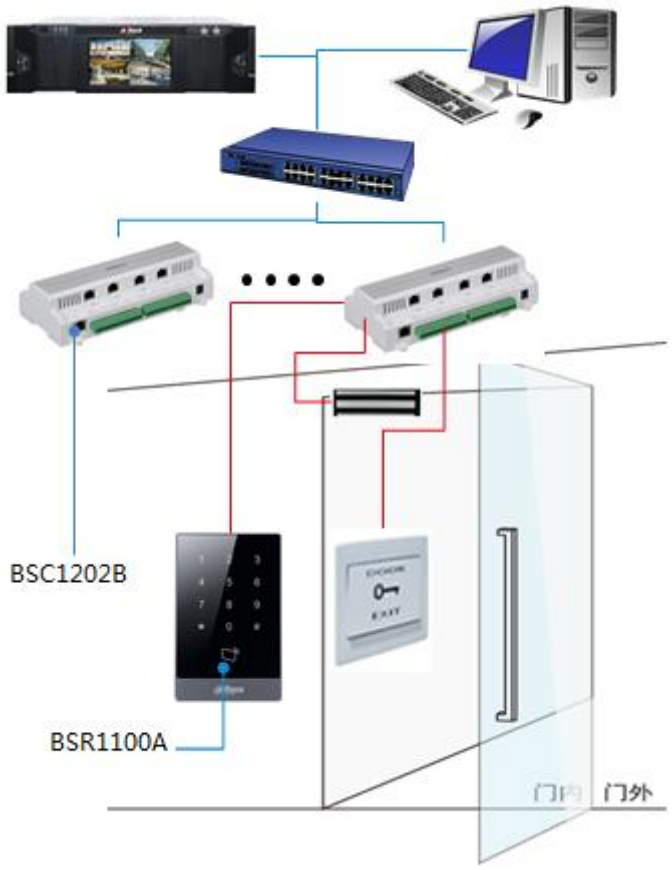
- 微型门禁适用于小型办公

图1-2 微型门禁



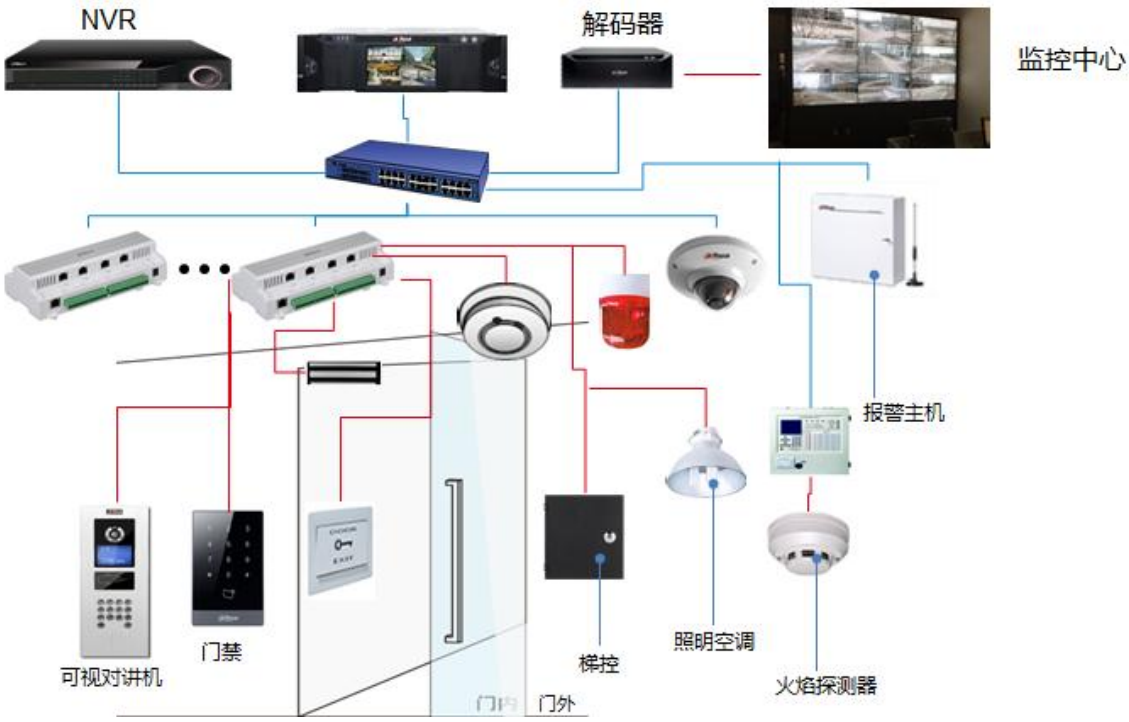
- 网络型门禁适用于中小型以上智能楼宇项目及金库和监所项目

图1-3 网络型门禁



- 增强型门禁

图1-4 增强型门禁



- 智能办公门锁适用于高档写字楼、政府办公、别墅房间门等

图1-5 智能办公门锁

2 通用功能

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

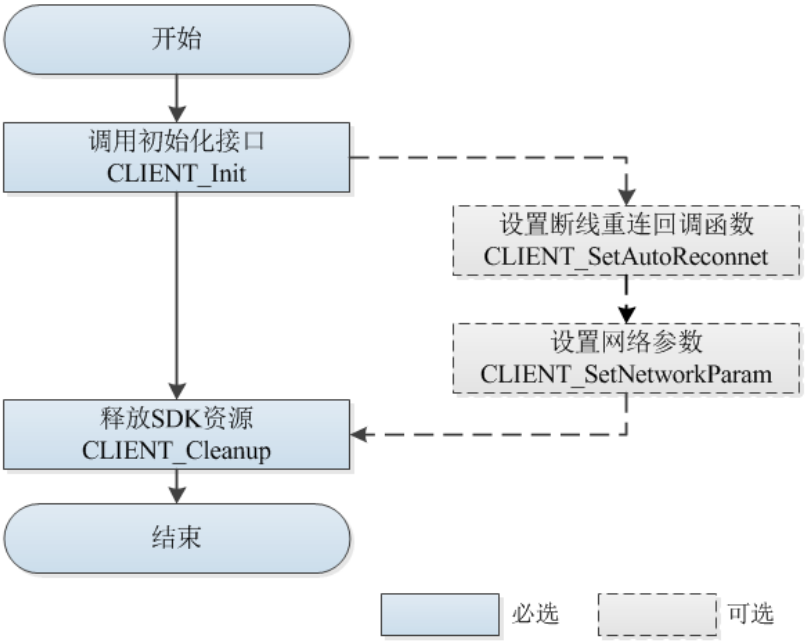
表2-1 SDK 初始化接口信息

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	设置登录网络环境接口

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、实时监视和报警订阅等。
- 断线重连：SDK 可以设置断线重连功能，当遇到一些特殊情况（例如断网、断电等）设备断线时，在 SDK 内部会定时持续不断地进行登录操作，直至成功登录设备。断线重连后可以恢复实时监视、报警和智能图片订阅业务，其他业务无法恢复。

2.1.4 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

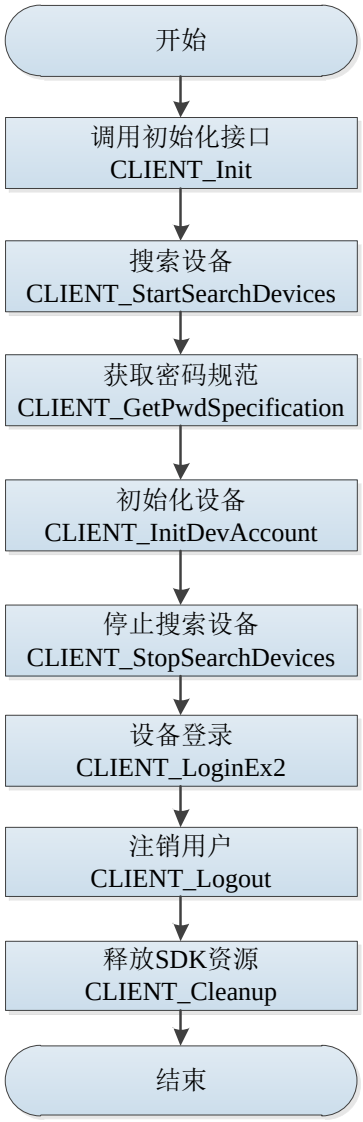
接口	说明
CLIENT_StartSearchDevices	搜索局域网内的设备，找到未初始化设备
CLIENT_InitDevAccount	设备初始化接口
CLIENT_GetDescriptionForResetPwd	获取密码重置信息：手机号、邮箱和二维码信息
CLIENT_CheckAuthCode	校验安全码是否有效
CLIENT_ResetPwd	重置密码
CLIENT_GetPwdSpecification	获取密码规则
CLIENT_StopSearchDevices	停止搜索设备

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_StartSearchDevices 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_LoginEx2，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。

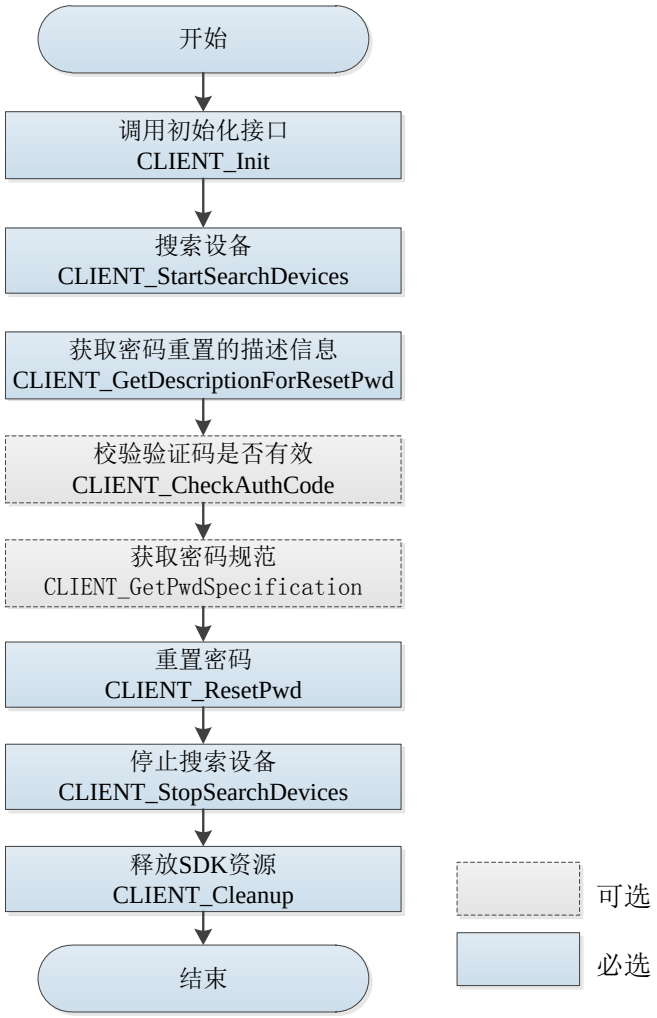
注意事项

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_StartSearchDevices 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_LoginEx2，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_StartSearchDevices ，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_StartSearchDevices，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 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 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 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

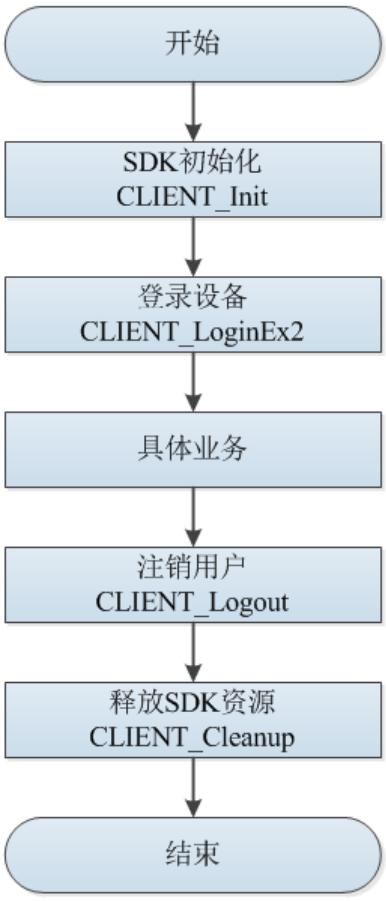
表2-3 设备登录接口信息

接口	说明
CLIENT_LoginEx2	登录扩展接口 2
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_LoginEx2 登录设备。
步骤3　登录成功后，用户可以实现需要的业务功能。
步骤4　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- 登录句柄：登录成功时接口返回值非 0（即句柄可能小于 0，也属于登录成功）；同一设备登

录多次，每次的登录句柄不一样。如果无特殊业务，建议只登录一次，登录的句柄可以重复用于其他各种业务。

- 登出：接口内部会释放登录会话中已打开的业务，但建议用户不要依赖登出接口的清理功能。例如打开监视后，在不需要使用监视时，用户应该调用结束监视的接口。
- 登录与登出配对使用，登录会消耗一定的内存和 socket 信息，在登出后释放资源。
- 登录失败：建议通过登录接口的 error 参数（登录错误码）初步排查。常见错误码如表 2-4 所示。

表2-4 常见错误码

error 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_LoginEx2 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
LLONG lLoginHandle = CLIENT_LoginEx2(szDevIp, nPort, szUserName, szPasswd,
EM_LOGIN_SPEC_CAP_TCP, NULL, &stDevInfo, &nError);
// 登出设备
CLIENT_Logout(lLoginHandle);
```

2.4 实时监视

2.4.1 简介

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

SDK 登录设备后，可向设备获取主码流和辅码流。

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

2.4.2 接口总览

表2-5 实时监视接口信息

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

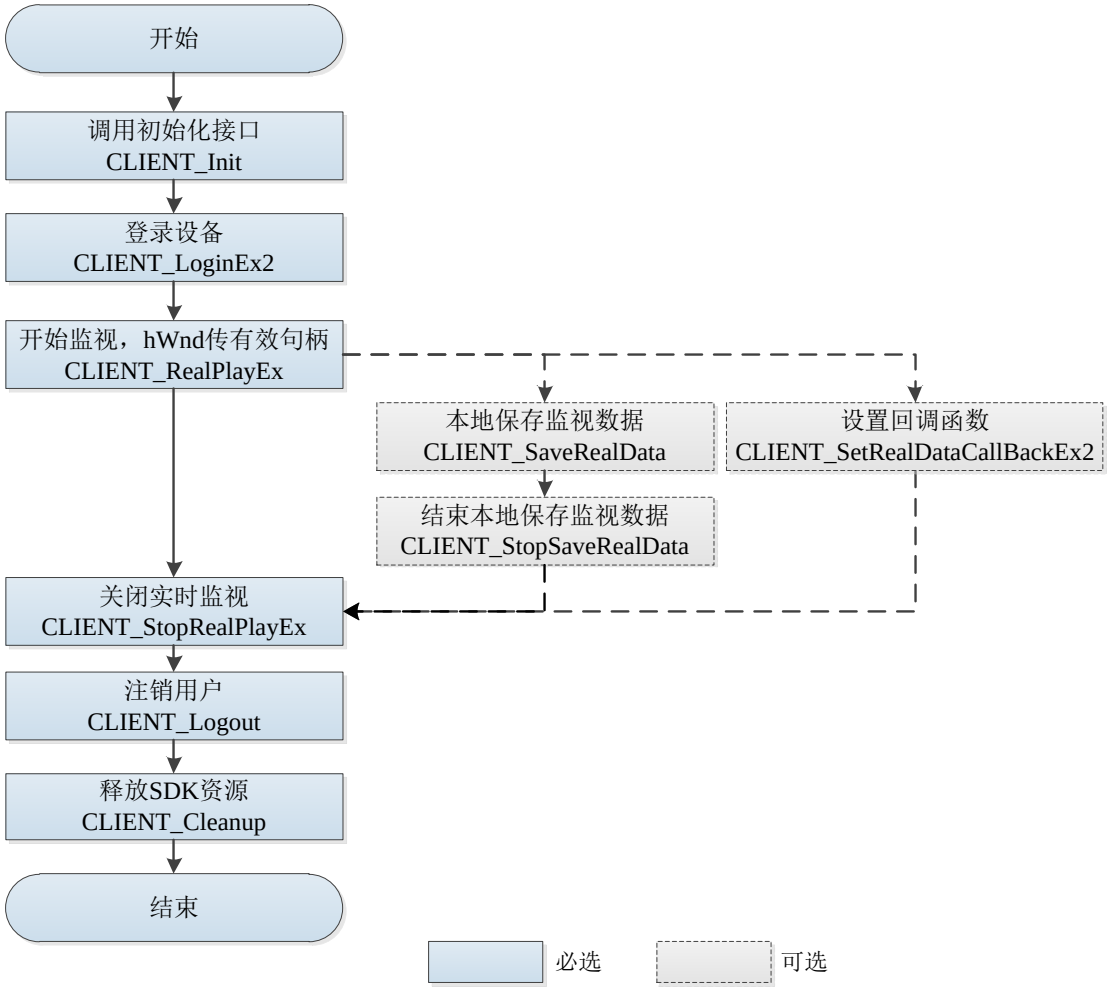
2.4.3 流程说明

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_LoginEx2 登录设备。
步骤3　调用 CLIENT_RealPlayEx 启动实时监视，参数 **hWnd** 为有效窗口句柄。
步骤4　（可选）调用 CLIENT_SaveRealData 开始保存监视数据。
步骤5　（可选）调用 CLIENT_StopSaveRealData 结束保存，生成本地视频文件。
步骤6　（可选）若调用 CLIENT_SetRealDataCallBackEx2，用户可将视频数据选择保存或转发。若保存成文件，与步骤 4、5 效果相同。
步骤7　实时监视使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时监视。
步骤8　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤9　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- SDK 解码播放只支持 Windows 系统，非 Windows 系统需要用户获取码流后自己调用解码显示。
- 多线程调用：同一个登录会话内的业务，不支持多线程调用；可以多个线程处理不同的登录会话中的业务，但不建议这样调用。
- 超时：接口内申请监视资源需和设备做一些约定，然后才请求监视数据，过程中有一些超时时间的设定（请参见 NET_PARAM 结构体），其中与监视相关的字段为 **nGetConnInfoTime**。

如果实际使用中（如网络状况不良）有超时现象，可将 **nGetConnInfoTime** 的值修改大一些。示例代码如下，在 CLIENT_Init 函数后调用，调用一次即可：

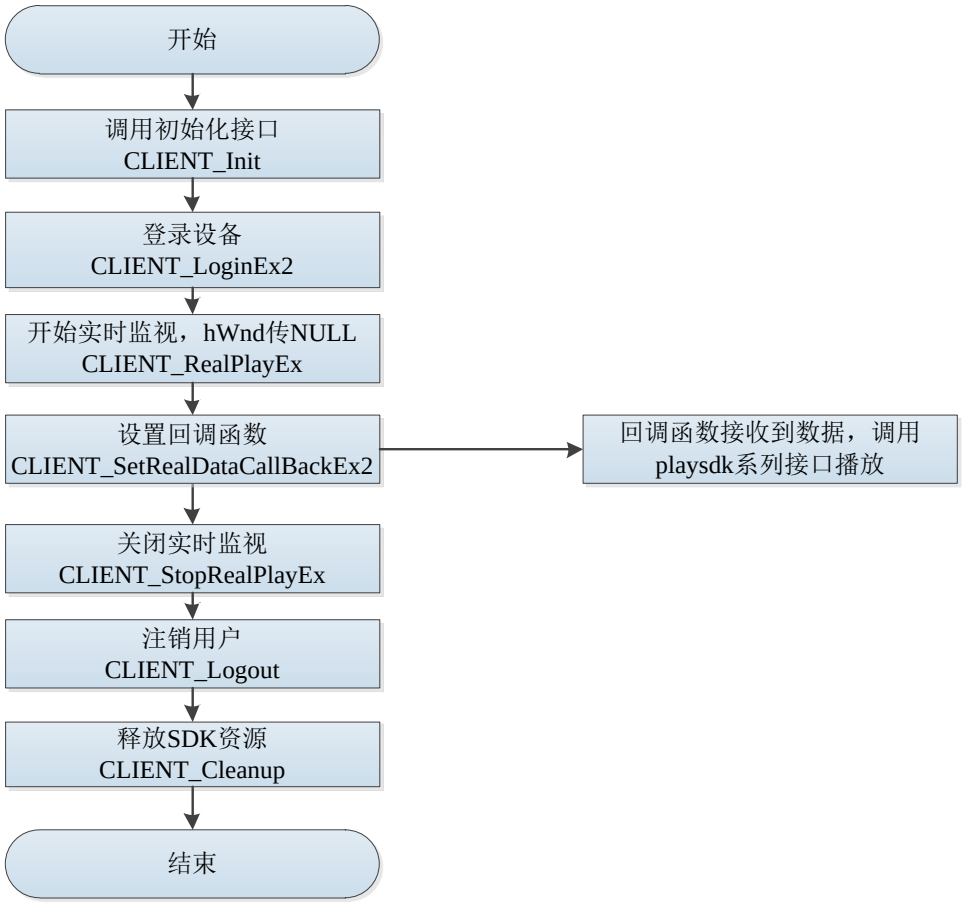
```
NET_PARAM stuNetParam = {0};
stuNetParam. nGetConnInfoTime = 5000; // 单位 ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- 重复打开失败：部分设备不支持同一次登录下同一个通道多次打开，当重复打开同一通道的监视，可能会出现第一次打开成功，后续打开失败的现象。建议：
  - ◇ 将已打开的通道先关闭。例如已经开启通道一的主码流视频，希望再打开通道一的辅码流视频时，可先关闭通道一的主码流视频，再开启通道一的辅码流视频。
  - ◇ 登录两次设备获取两个登录句柄，分别处理主码流和辅码流业务。
- 接口成功无画面：SDK 内部解码需要使用到 dhplay.dll，建议查看运行目录下是否缺少 dhplay.dll 及其依赖的辅助库，具体请参见表 1-1。
- 系统资源不足的情况下，设备可能返回错误而不恢复码流，可以在报警回调函数（即 CLIENT_SetDVRMessCallBack 中设置的回调函数）收到事件 DH_REALPLAY_FAILD_EVENT，该事件包含了详细的错误码，请参见《网络 SDK 开发手册》中的“DEV_PLAY_RESULT 结构体”。
- 32 路限制：解码显示比较消耗资源，特别是高分辨率视频，考虑到客户端硬件资源有限，一般同时解码显示的通道数有限，所以该方式暂时限定为最多 32 路，如超过 32 路，建议使用“2.4.3.2 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 回调实时监视码流给用户，用户调用 PlaySDK 进行解码播放。用户调用第三方解码播放流程如图 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_LoginEx2 登录设备。
步骤3　登录成功后，调用 CLIENT_RealPlayEx 启动实时监视，参数 **hWnd** 为 **NULL**。
步骤4　调用 CLIENT_SetRealDataCallBackEx2 设置实时数据回调函数。
步骤5　在回调函数中将数据传给 PlaySDK 完成解码。
步骤6　实时监视使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时监视。
步骤7　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤8　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

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

2.4.4 示例代码

2.4.4.1 SDK 解码播放

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

2.4.4.2 调用播放库

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

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

```
}

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

2.5 语音对讲

2.5.1 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 实现与设备的语音对讲。

2.5.2 接口总览

表2-6 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）
CLIENT_SetDeviceMode	设置设备语音对讲工作模式

2.5.3 流程说明

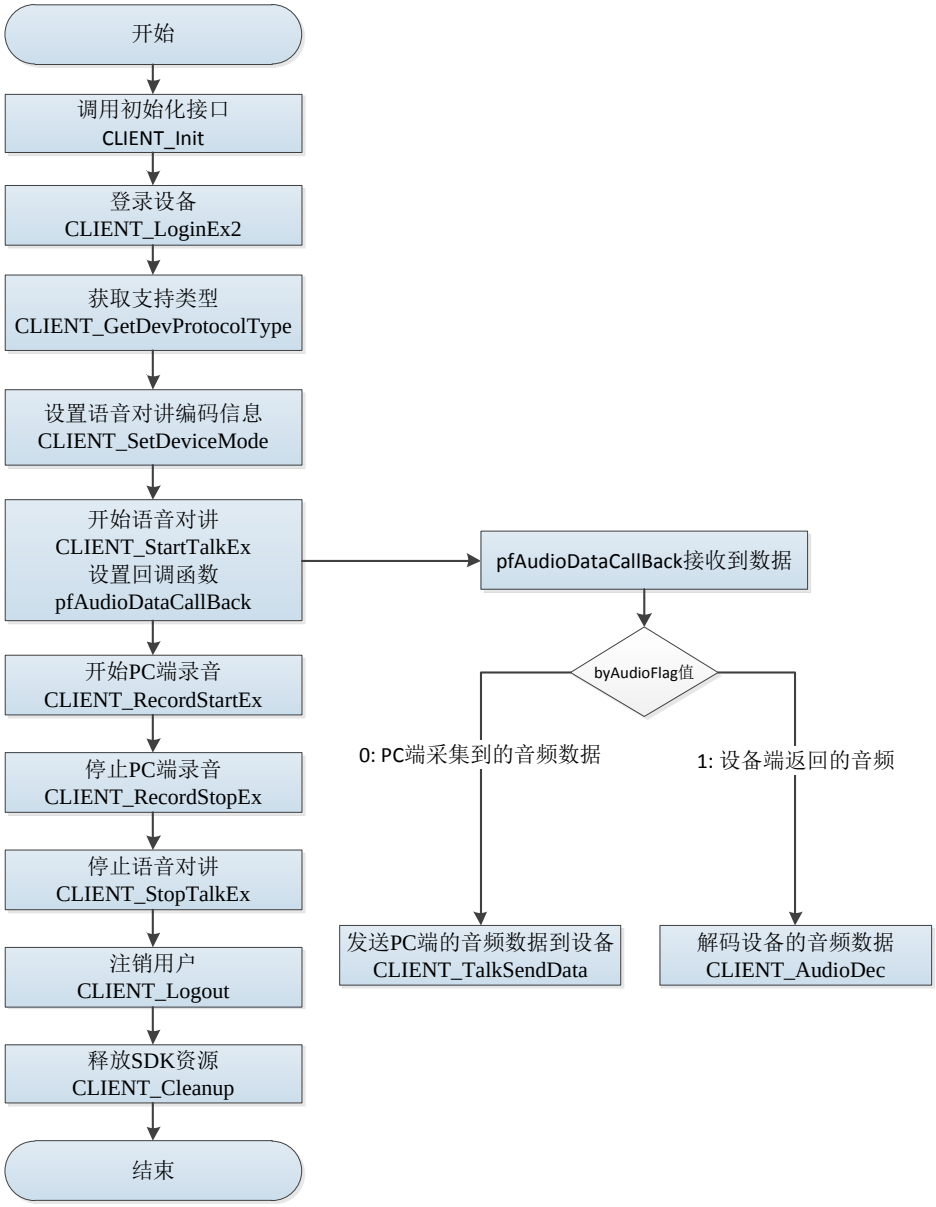
当 SDK 从本地声卡采集到音频数据或 SDK 接收到前端发送过来的音频数据时，会调用音频数据回调函数。用户可在回调函数中调用 SDK 接口将采集到的本地音频数据发送到前端设备，也可以调用 SDK 接口将接收到的前端设备的音频数据进行解码播放。语音对讲流程如图 2-7 所示。

该模式只在 Windows 平台下有效。

说明

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

图2-7 二代语音对讲流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

步骤2　初始化成功后，调用 CLIENT_LoginEx2 登录设备。

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

步骤4　调用 CLIENT_SetDeviceMode 设置语音对讲参数。

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

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

步骤6　调用 CLIENT_RecordStartEx 开始 PC 端录音，该接口调用后，CLIENT_StartTalkEx 设置

的语音对讲回调函数中才会收到本地音频数据。

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

步骤8　调用 CLIENT_StopTalkEx 停止语音对讲。

步骤9　调用 CLIENT_Logout 登出设备。

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

注意事项

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

2.5.4 示例代码

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

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

// 根据获取出的对讲类型，设置对讲参数
if (emTpye == EM_DEV_PROTOCOL_V3) //三代对讲需要设备此类参数，而二代设备不需要设置此类参数
{
    NET_TALK_EX stuTalk = {sizeof(stuTalk)};
        stuTalk.nAudioPort = RECEIVER_AUDIO_PORT;    // 用户自定义接收端口
        stuTalk.nChannel   = 0;
        stuTalk.nWaitTime  = 5000;
        CLIENT_SetDeviceMode(m_lLoginHandle, DH_TALK_MODE3, &stuTalk)
}
// 开始语音对讲
lTalkHandle = CLIENT_StartTalkEx(lLoginHandle, AudioDataCallBack, (LDWORD)NULL);
// 开始本地录音
CLIENT_RecordStartEx(lLoginHandle);
// 停止本地录音
```

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

2.6 事件侦听

2.6.1 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，用户可通过报警回调函数获取相应的报警信息。

2.6.2 接口总览

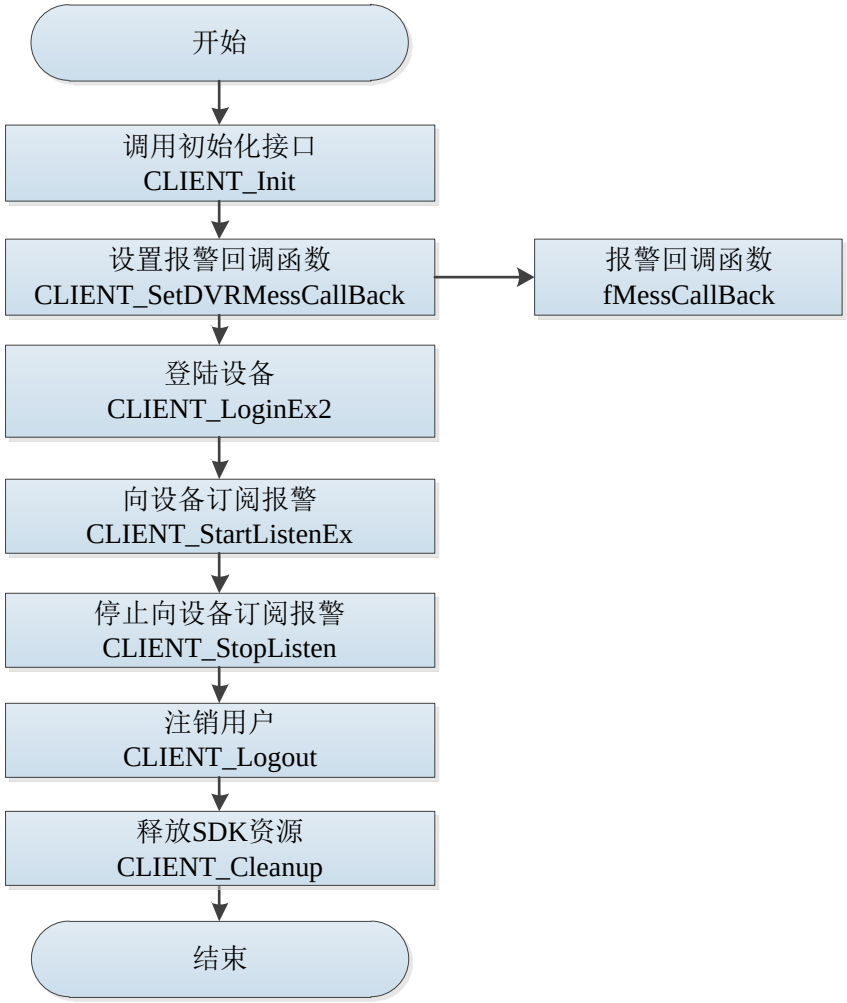
表2-7 报警侦听上报的接口信息

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

2.6.3 流程说明

报警上报流程如图 2-8 所示。

图2-8 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。

说明

该接口需在报警订阅之前调用。

步骤3　调用 CLIENT_LoginEx2 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 设置的回调函数通知用户。

说明

报警主机、门禁和语音对讲相关的报警事件，具体请参见“6.6 报警回调函数 fMessCallBack”。

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。

- 报警回调函数 fMessCallBack 中的报警信息，建议异步处理，不建议在回调里做过多操作，否则会阻塞回调。

2.6.4 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_ACCESS_CTL_EVENT) // 门禁事件，更多门禁相关事件请参见“6.6 报警回调函数 fMessCallBack”。
    {
        ALARM_ACCESS_CTL_EVENT_INFO* pstAccessInfo =
                                        (ALARM_ACCESS_CTL_EVENT_INFO*)pBuf;
         //接下来可以通过 pstAccessInfo 获取对应的报警信息。
    }
    ……..
}
// 设置报警回调
CLIENT_SetDVRMessCallBack(afMessCallBack,0);
// 订阅报警
CLIENT_StartListenEx(lLoginHandle);
// 停止报警订阅
CLIENT_StopListen(lLoginHandle);
```

3.1 布撤防

3.1.1 简介

- 布防：设备所有防区处于警戒状态，接收外部信号，并可处理、记录和转发信号。
- 撤防：设备所有防区都不会接收外部报警信号，也不会处理、记录和转发信号。

3.1.2 接口总览

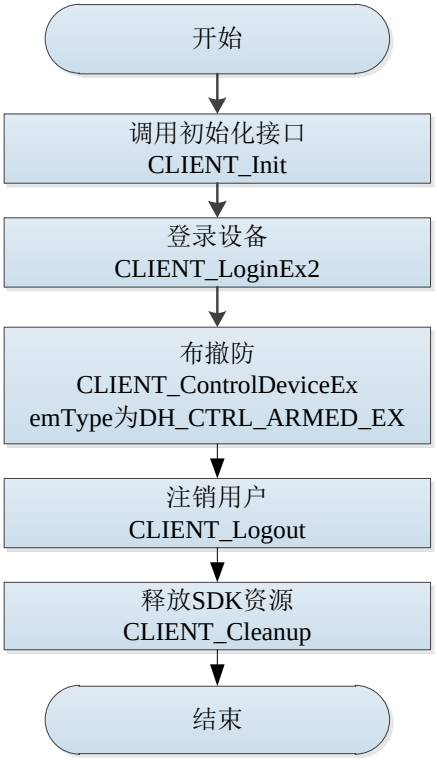
表3-1 布撤防接口信息

接口	说明
CLIENT_ControlDeviceEx	设备控制扩展接口

3.1.3 流程说明

布撤防流程如图 3-1 所示。

图3-1 布撤防业务流程



流程说明

步骤1　调用 CLIENT_Init 函数，完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginEx2 函数登录设备。
步骤3　调用 CLIENT_ControlDeviceEx 函数来控制设备布防撤防，函数参数 **emType** 值为 **DH_CTRL_ARMED_EX**。
步骤4　业务执行完成之后，调用 CLIENT_Logout 函数登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

3.1.4 示例代码

```
CTRL_ARM_DISARM_PARAM_EX stuParam = {sizeof(stuParam)};
stuParam.stuIn.dwSize = sizeof(stuParam.stuIn);
stuParam.stuOut.dwSize = sizeof(stuParam.stuOut);

stuParam.stuIn.emState = NET_ALARM_MODE_ARMING;
stuParam.stuIn.emSceneMode = NET_SCENE_MODE_OUTDOOR;
stuParam.stuIn.szDevPwd = "admin";
CLIENT_ControlDeviceEx(g_lLoginHandle, DH_CTRL_ARMED_EX, &stuParam, NULL,3000);
```

3.2 设置防区状态

3.2.1 简介

设置防区状态，控制设备报警通道的正常、旁路和隔离状态。

3.2.2 接口总览

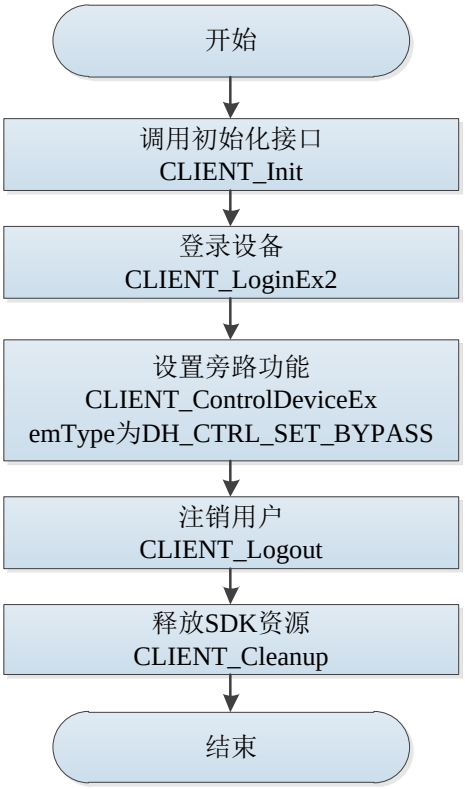
表3-2 设置防区状态接口信息

接口	说明
CLIENT_ControlDeviceEx	设备控制扩展接口

3.2.3 流程说明

设置防区状态流程如图 3-2 所示。

图3-2 设置防区状态业务流程



流程说明

步骤1　调用 CLIENT_Init 函数，完成 SDK 初始化流程。

步骤2　调用 CLIENT_LoginEx2 函数登录设备。

步骤3　调用 CLIENT_ControlDeviceEx 函数来控制设备设置防区状态，函数参数 **emType** 值为 **DH_CTRL_SET_BYPASS**。

步骤4　业务执行完成之后，调用 CLIENT_Logout 函数登出设备。

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

3.2.4 示例代码

```
NET_CTRL_SET_BYPASS stuParam = {sizeof(stuParam)};
stuParam.dwSize = sizeof(stuParam);
stuParam.emMode = NET_BYPASS_MODE_BYPASS; //  设置防区状态为旁路状态
stuParam.szDevPwd = "admin";
stuParam.nExtendedCount = 1;
int nExtendChn[1] = {1};
stuParam.pnExtended = nExtendChn;
stuParam.nLocalCount = 2;
int nLocalChn[2] = {2,3};
stuParam.pnLocal = nLocalChn;
CLIENT_ControlDeviceEx(g_lLoginHandle, DH_CTRL_SET_BYPASS, &stuParam, NULL, 3000);
```

3.3 防区状态查询

3.3.1 简介

查询防区状态，包括报警输入状态、报警输出状态、警号状态等。

3.3.2 接口总览

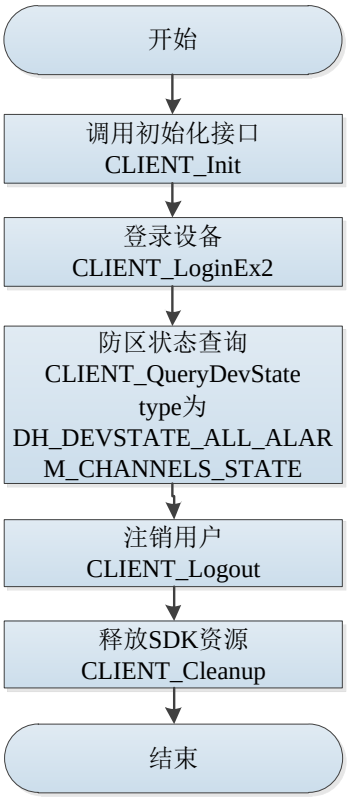
表3-3 防区状态查询接口信息

接口	说明
CLIENT_QueryDevState	状态查询接口

3.3.3 流程说明

防区状态查询流程如图 3-3 所示。

图3-3 防区状态查询业务流程



流程说明

步骤1　调用 CLIENT_Init 函数，完成 SDK 初始化流程。

步骤2　调用 CLIENT_LoginEx2 函数登录设备。

步骤3　业务实现，调用 CLIENT_QueryDevState 来实现防区状态查询，此时参数 **type** 为 **DH_DEVSTATE_ALL_ALARM_CHANNELS_STATE**。

步骤4　业务执行完成之后，调用 CLIENT_Logout 函数登出设备。

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

3.3.4 示例代码

```
NET_CLIENT_ALARM_CHANNELS_STATE stuAlarmChannelState = {sizeof(stuAlarmChannelState)};
stuAlarmChannelState.emType = NET_ALARM_CHANNEL_TYPE_ALL; // 查询所有通道状态；
int nNum = 2;

// 警号通道相关字段初始化
stuAlarmChannelState.nAlarmBellCount = nNum;
stuAlarmChannelState.pbAlarmBellState = new BOOL[stuAlarmChannelState.nAlarmBellCount];
memset(stuAlarmChannelState.pbAlarmBellState, 0, stuAlarmChannelState.nAlarmBellCount * sizeof(BOOL));

// 报警输入通道相关字段初始化
stuAlarmChannelState.nAlarmInCount = nNum;
stuAlarmChannelState.pbAlarmInState = new BOOL[stuAlarmChannelState.nAlarmInCount];
memset(stuAlarmChannelState.pbAlarmInState, 0, stuAlarmChannelState.nAlarmInCount * sizeof(BOOL));

// 报警输出通道相关字段初始化
stuAlarmChannelState.nAlarmOutCount = nNum;
stuAlarmChannelState.pbAlarmOutState = new BOOL[stuAlarmChannelState.nAlarmOutCount];
memset(stuAlarmChannelState.pbAlarmOutState, 0, stuAlarmChannelState.nAlarmOutCount * sizeof(BOOL));

// 扩展模块报警输入通道相关字段初始化
stuAlarmChannelState.nExAlarmInCount = nNum;
stuAlarmChannelState.pbExAlarmInState = new BOOL[stuAlarmChannelState.nExAlarmInCount];
memset(stuAlarmChannelState.pbExAlarmInState, 0, stuAlarmChannelState.nExAlarmInCount * sizeof(BOOL));
stuAlarmChannelState.pnExAlarmInDestionation = new int[1024];

// 扩展模块报警输出通道相关字段初始化
stuAlarmChannelState.nExAlarmOutCount = nNum;
stuAlarmChannelState.pbExAlarmOutState = new BOOL[stuAlarmChannelState.nExAlarmOutCount];
memset(stuAlarmChannelState.pbExAlarmOutState, 0, stuAlarmChannelState.nExAlarmOutCount *
sizeof(BOOL));
stuAlarmChannelState.pnExAlarmOutDestionation = new int[1024];

int nRetLen = 0;
CLIENT_QueryDevState(g_lLoginHandle, DH_DEVSTATE_ALL_ALARM_CHANNELS_STATE,
(char*)&stuAlarmChannelState, sizeof(NET_CLIENT_ALARM_CHANNELS_STATE), &nRetLen, 3000);
```

4 门禁

4.1 门禁控制

4.1.1 简介

控制门禁的打开和关闭。

4.1.2 接口总览

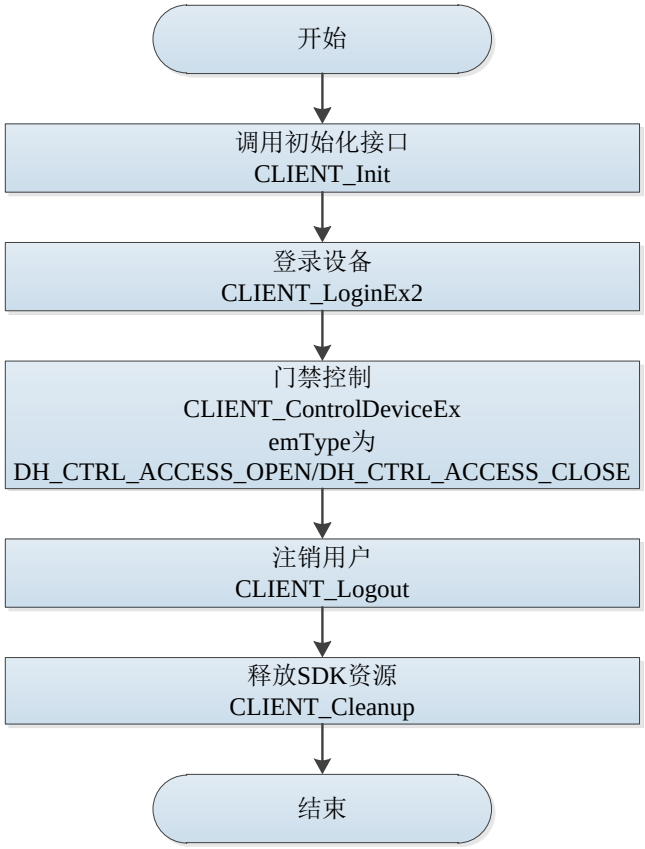
表4-1 门禁控制接口信息

接口	说明
CLIENT_ControlDeviceEx	设备控制扩展接口

4.1.3 流程说明

门禁控制流程如图 4-1 所示。

图4-1 门禁控制业务流程



流程说明

步骤1　调用 CLIENT_Init 函数，完成 SDK 初始化流程。

步骤2　调用 CLIENT_LoginEx2 函数登录设备。

步骤3　调用 CLIENT_ControlDeviceEx 函数来控制门禁。

- 打开门禁：**emType** 值为 **DH_CTRL_ACCESS_OPEN**。
- 关闭门禁：**emType** 值为 **DH_CTRL_ACCESS_CLOSE**。

步骤4　业务执行完成之后，调用 CLIENT_Logout 函数登出设备。

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

4.1.4 示例代码

```
// 打开门禁
NET_CTRL_ACCESS_OPEN stOpen = {sizeof(stOpen)};
stOpen.nChannelID = 0;
strncpy(stOpen.szUserID, “admin”, sizeof(stOpen.szUserID) - 1);
CLIENT_ControlDeviceEx((LLONG)g_lLoginHandle, DH_CTRL_ACCESS_OPEN, &stOpen, NULL, 3000);

//关闭门禁
NET_CTRL_ACCESS_CLOSE stClose = {sizeof(stClose)};
CLIENT_ControlDeviceEx((LLONG)g_lLoginHandle, DH_CTRL_ACCESS_CLOSE, &stClose, NULL, 3000);
```

5 接口函数

5.1 SDK 初始化

5.1.1 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	

5.1.2 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

5.1.3 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

5.1.4 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	

5.2 设备初始化

5.2.1 搜索设备 CLIENT_StartSearchDevices

选项	说明	
描述	搜索设备信息	
函数	LLONG CLIENT_StartSearchDevices (fSearchDevicesCB cbSearchDevices, void* pUserData, char* szLocalIp=NULL);	
参数	[in]cbSearchDevices	输入参数，设备信息数据回调函数
	[out]pUserData	输入参数，用户数据
	[in]szLocalIp	● 单网卡情况可填写 NULL，表示使用本机 IP ● 多网卡情况，填写需要指定网卡 IP
返回值	搜索句柄	
说明	不支持多线程调用	

5.2.2 设备初始化 CLIENT_InitDevAccount

选项	说明
描述	初始化设备
函数	BOOL CLIENT_InitDevAccount(const NET_IN_INIT_DEVICE_ACCOUNT *pInitAccountIn, NET_OUT_INIT_DEVICE_ACCOUNT *pInitAccountOut, DWORD dwWaitTime, char *szLocalIp);

选项	说明	
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	

5.2.3 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	

5.2.4 检验安全码是否有效 CLIENT_CheckAuthCode

选项	说明
描述	检验安全码否有效
函数	BOOL CLIENT_CheckAuthCode(const NET_IN_CHECK_AUTHCODE *pCheckAuthCodeIn, NET_OUT_CHECK_AUTHCODE *pCheckAuthCodeOut, DWORD dwWaitTime, char *szLocalIp);

选项	说明	
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	

5.2.5 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	

5.2.6 获取密码规则 CLIENT_GetPwdSpecification

选项	说明	
描述	获取密码规则	
函数	BOOL CLIENT_GetPwdSpecification(const NET_IN_PWD_SPECI *pPwdSpeciIn, NET_OUT_PWD_SPECI *pPwdSpeciOut, DWORD dwWaitTime, char *szLocalIp);	
参数	[in]pPwdSpeciIn	输入参数，对应 NET_IN_PWD_SPECI 结构体
	[out]pPwdSpeciOut	输出参数，对应 NET_OUT_PWD_SPECI 结构体
	[in]dwWaitTime	超时时间

选项	说明	
	[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	

5.2.7 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	

5.3 设备登录

5.3.1 用户登录设备 CLIENT_LoginEx2

选项	说明	
描述	用户登录设备	
函数	LLONG CLIENT_LoginEx2(const char *pchDVRIP, WORD wDVRPort, const char *pchUserName, const char *pchPassword, EM_LOGIN_SPAC_CAP_TYPE emSpecCap, void* pCapParam, LPNET_DEVICEINFO_Ex lpDeviceInfo, int *error);	
参数	[in]pchDVRIP	设备 IP
	[in]wDVRPort	设备端口
	[in]pchUserName	用户名
	[in]pchPassword	密码
	[in]emSpecCap	登录类别
	[in]pCapParam	登录类别参数
	[out]lpDeviceInfo	设备信息
	[out]error	失败的错误码

选项	说明
返回值	● 成功返回非 0 ● 失败返回 0
说明	无

参数 error 的错误码及含义说明，请参见表 5-1。

表5-1 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

5.3.2 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_LoginEx2 的返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

5.4 实时监视

5.4.1 打开监视 CLIENT_RealPlayEx

选项	说明
描述	打开实时监视

选项	说明	
函数	LLONG CLIENT_RealPlayEx(LLONG lLoginID, int nChannelID, HWND hWnd, DH_RealPlayType rType);	
参数	[in]lLoginID	CLIENT_LoginEx2 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 5-2。

表5-2 预览类型说明

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

5.4.2 关闭监视 CLIENT_StopRealPlayEx

选项	说明	
描述	关闭实时监视	
函数	BOOL CLIENT_StopRealPlayEx(LLONG lRealHandle);	
参数	[in]lRealHandle	CLIENT_RealPlayEx 的返回值

选项	说明
返回值	● 成功返回 TRUE ● 失败返回 FALSE
说明	无

5.4.3 保存监视数据 CLIENT_SaveRealData

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

5.4.4 停止保存监视数据 CLIENT_StopSaveRealData

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

5.4.5 设置监视数据回调 CLIENT_SetRealDataCallBackEx2

选项	说明	
描述	设置实时监视数据回调	
函数	BOOL CLIENT_SetRealDataCallBackEx2(LLONG lRealHandle, fRealDataCallBackEx2 cbRealData, LDWORD dwUser, DWORD dwFlag);	
参数	[in]lRealHandle	CLIENT_RealPlayEx 的返回值
	[in]cbRealData	监视数据流回调函数
	[in]dwUser	监视数据流回调函数的参数

选项	说明	
	[in]dwFlag	回调中监视数据的类型，EM_REALDATA_FLAG 类型，支持或运算
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

表5-3 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

5.5 设备控制

5.5.1 设备控制 CLIENT_ControlDeviceEx

选项	说明	
描述	设备控制	
函数	BOOL CLIENT_ControlDeviceEx(LLONG lLoginID, CtrlType emType, Void *pInBuf, Void *pOutBuf = NULL, int nWaitTime = 1000);	
参数	[in]lLoginID	CLIENT_LoginEx2 的返回值
	[in]emType	控制类型
	[in]pInBuf	输入参数，因 emType 不同而不同
	[out]pOutBuf	输出参数，默认为 NULL，对于有些 emType 有相应输出结构体
	[in]waittime	超时时间，默认 1000ms，可根据需要自行设置
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

emType、pInBuf 和 pOutBuf 的对照关系请参见表 5-4。

表5-4 emType、pInBuf 和 pOutBuf 对照关系

emType	描述	pInBuf	pOutBuf
DH_CTRL_ARMED_EX	布撤防	CTRL_ARM_DISARM_PARAM	NULL
DH_CTRL_SET_BYPASS	设置旁路功能	NET_CTRL_SET_BYPASS	NULL

5.6 报警侦听

5.6.1 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	

5.6.2 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_LoginEx2 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	

5.6.3 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_LoginEx2 返回值
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

5.7 获取设备状态

5.7.1 获取设备状态 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_LoginEx2 返回值
	[in]nType	查询信息类型
	[out]pBuf	用于接收查询返回的数据缓存，根据查询类型的不同，返回数据的数据结构也不同，详细请参见表 5-5
	[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，可根据需要设置
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

表5-5 查询信息类型和结构体对应关系

查询项	nType	pBuf
查询报警通道状态	DH_DEVSTATE_ALL_ALARM_CHANNELS_STATE	NET_CLIENT_ALARM_CHANNELS_STATE
查询电源与电池信息	DH_DEVSTATE_POWER_STATE	DH_POWER_STATUS

5.8 语音对讲

5.8.1 获取设备支持对讲类型 CLIENT_GetDevProtocolType

选项	说明	
描述	获取设备支持对讲类型	
函数	BOOL CLIENT_GetDevProtocolType(LLONG lLoginID, EM_DEV_PROTOCOL_TYPE *pemProtocolType);	
参数	[in]lLoginID	CLIENT_LoginEx2 的返回值

选项	说明	
	[out]pemProtocolType	设备支持的协议类型，对应 EM_DEV_PROTOCOL_TYPE 结构体
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

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

选项	说明	
描述	设置设备语音对讲工作模式	
函数	BOOL CLIENT_SetDeviceMode(LLONG lLoginID, EM_USEDEV_MODE emType, void *pValue);	
参数	[in]lLoginID	CLIENT_LoginEx2 的返回值
	[in]emType	枚举值
	[in]pValue	与枚举值对应的结构体数据指针，请参见表 5-6
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
说明	无	

emType 和 pValue 对照关系请参见表 5-6。

表5-6 emType 和 pValue 对照关系

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

5.8.3 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_LoginEx2 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	

5.8.4 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	

5.8.5 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_LoginEx2 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	

5.8.6 关闭录音 CLIENT_RecordStopEx

选项	说明	
描述	关闭本地录音	
函数	BOOL CLIENT_RecordStopEx(LLONG lLoginID);	
参数	[in]lLoginID	CLIENT_LoginEx2 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	

5.8.7 发送语音 CLIENT_TalkSendData

选项	说明
描述	发送音频数据给设备

选项	说明	
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	

5.8.8 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	

6 回调函数

6.1 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	

6.2 断线回调函数 fDisConnect

选项	说明	
描述	断线回调函数	
函数	typedef void (CALLBACK *fDisConnect)(LLONG lLoginID, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
参数	[out]lLoginID	CLIENT_LoginEx2 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	

6.3 断线重连回调函数 fHaveReConnect

选项	说明
描述	断线重连回调函数

选项	说明	
函数	typedef void (CALLBACK *fHaveReConnect)(LLONG lLoginID, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
参数	[out]lLoginID	CLIENT_LoginEx2 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	

6.4 实时监视数据回调函数 fRealDataCallBackEx2

选项	说明	
描述	实时监视数据回调函数	
函数	typedef void (CALLBACK *fRealDataCallBackEx2)(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LLONG param, LDWORD dwUser);	
参数	[out]lRealHandle	CLIENT_RealPlayEx 的返回值
	[out]dwDataType	数据类型 ● 0 表示原始数据 ● 1 表示带有帧信息的数据 ● 2 表示 YUV 数据 ● 3 表示 PCM 音频数据
	[out]pBuffer	监视数据块地址
	[out]dwBufSize	监视数据块的长度，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	

选项	说明
说明	无

6.5 音频数据回调函数 pfAudioDataCallBack

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

6.6 报警回调函数 fMessCallBack

选项	说明	
描述	报警回调函数	
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 6-1
	[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

选项	说明	
	[out]pchDVRIP	设备 IP
	[out]nDVRPort	端口
	[out]dwUser	用户自定义数据
返回值	● 成功返回 TRUE ● 失败返回 FALSE	
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6-1 报警类型和结构体对应关系

报警所属业务	报警类型名称	lCommand	pBuf
报警主机	本地报警事件	DH_ALARM_ALARM_EX2	ALARM_ALARM_INFO_EX2
	电源故障事件	DH_ALARM_POWERFAULT	ALARM_POWERFAULT_INFO
	防拆事件	DH_ALARM_CHASSISINTRUDED	ALARM_CHASSISINTRUDED_INFO
	扩展报警输入通道事件	DH_ALARM_ALARMEXTENDED	ALARM_ALARMEXTENDED_INFO
	紧急事件	DH_URGENCY_ALARM_EX	数据为 16 个字节数组，每个字节表示一个通道状态 1 为有报警 0 为无报警
	蓄电池低电压事件	DH_ALARM_BATTERYLOWPOWER	ALARM_BATTERYLOWPOWER_INFO
	设备邀请平台对讲事件	DH_ALARM_TALKING_INVITE	ALARM_TALKING_INVITE_INFO
	设备布防模式变化事件	DH_ALARM_ARMMODE_CHANGE_EVENT	ALARM_ARMMODE_CHANGE_INFO
	防区旁路状态变化事件	DH_ALARM_BYPASSMODE_CHANGE_EVENT	ALARM_BYPASSMODE_CHANGE_INFO
	报警输入源信号事件	DH_ALARM_INPUT_SOURCE_SIGNAL	ALARM_INPUT_SOURCE_SIGNAL_INFO
	消警事件	DH_ALARM_ALARMCLEAR	ALARM_ALARMCLEAR_INFO
	子系统状态改变事件	DH_ALARM_SUBSYSTEM_STATE_CHANGE	ALARM_SUBSYSTEM_STATE_CHANGE_INFO
	扩展模块掉线事件	DH_ALARM_MODULE_LOST	ALARM_MODULE_LOST_INFO
	PSTN 掉线事件	DH_ALARM_PSTN_BREAK_LINE	ALARM_PSTN_BREAK_LINE_INFO
	模拟量报警事件	DH_ALARM_ANALOG_PULSE	ALARM_ANALOGPULSE_INFO
	报警传输事件	DH_ALARM_PROFILE_ALARM_TRANSMIT	ALARM_PROFILE_ALARM_TRANSMIT_INFO
	无线设备低电量报警事件	DH_ALARM_WIRELESSDEV_LOWPOWER	ALARM_WIRELESSDEV_LOWPOWER_INFO

报警所属业务	报警类型名称	lCommand	pBuf
	防区布撤防状态改变事件	DH_ALARM_DEFENCE_ARMMODE_CHANGE	ALARM_DEFENCE_ARMMODECHANGE_INFO
	子系统布撤防状态改变事件	DH_ALARM_SUBSYSTEM_ARMMODE_CHANGE	ALARM_SUBSYSTEM_ARMMODECHANGE_INFO
	探测器异常报警	DH_ALARM_SENSOR_ABNORMAL	ALARM_SENSOR_ABNORMAL_INFO
	病人活动状态报警事件	DH_ALARM_PATIENTDETECTION	ALARM_PATIENTDETECTION_INFO
门禁	门禁事件	DH_ALARM_ACCESS_CTL_EVENT	ALARM_ACCESS_CTL_EVENT_INFO
	门禁未关事件详细信息	DH_ALARM_ACCESS_CTL_NOT_CLOSE	ALARM_ACCESS_CTL_NOT_CLOSE_INFO
	闯入事件详细信息	DH_ALARM_ACCESS_CTL_BREAK_IN	ALARM_ACCESS_CTL_BREAK_IN_INFO
	反复进入事件详细信息	DH_ALARM_ACCESS_CTL_REPEAT_ENTER	ALARM_ACCESS_CTL_REPEAT_ENTER_INFO
	胁迫卡刷卡事件详细信息	DH_ALARM_ACCESS_CTL_DURESS	ALARM_ACCESS_CTL_DURESS_INFO
	多人组合开门事件	DH_ALARM_OPENDOORGROUP	ALARM_OPEN_DOOR_GROUP_INFO
	防拆事件	DH_ALARM_CHASSISINTRUDED	ALARM_CHASSISINTRUDED_INFO
	本地报警事件	DH_ALARM_ALARM_EX2	ALARM_ALARM_INFO_EX2
	门禁状态事件	DH_ALARM_ACCESS_CTL_STATUS	ALARM_ACCESS_CTL_STATUS_INFO
	获取指纹事件	DH_ALARM_FINGER_PRINT	ALARM_CAPTURE_FINGER_PRINT_INFO
可视对讲	直连情况下，呼叫无答应事件	DH_ALARM_CALL_NO_ANSWERED	NET_ALARM_CALL_NO_ANSWERED_INFO
	手机号码上报事件	DH_ALARM_TELEPHONE_CHECK	ALARM_TELEPHONE_CHECK_INFO
	VTS 状态上报	DH_ALARM_VTSTATE_UPDATE	ALARM_VTSTATE_UPDATE_INFO
	VTO 人脸识别	DH_ALARM_ACCESSIDENTIFY	NET_ALARM_ACCESSIDENTIFY
	设备请求对方发起对讲事件	DH_ALARM_TALKING_INVITE	ALARM_TALKING_INVITE_INFO
	设备取消对讲请求事件	DH_ALARM_TALKING_IGNORE_INVITE	ALARM_TALKING_IGNORE_INVITE_INFO
	设备主动挂断对讲事件	DH_ALARM_TALKING_HANGUP	ALARM_TALKING_HANGUP_INFO
	雷达监测超速报警事件	DH_ALARM_RADAR_HIGH_SPEED	ALARM_RADAR_HIGH_SPEED_INFO