

NetSDK_JAVA

编程指导手册（动环主机分册）



V2.0.0

前言

目的

欢迎使用 NetSDK（以下简称 SDK）编程指导手册（动环主机分册）。

本文档描述了动环主机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》，基础核心业务流程（例如初始化、登录、普通报警和智能报警等）详见《NetSDK 编程指导手册》。

本文档提供的示例代码仅为演示接口调用方法，不保证能直接拷贝编译。

读者对象

使用 SDK 的软件开发工程师、产品经理和项目经理等。

符号约定

在本文档中可能出现下列标识，代表的含义如下。

标识	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员伤亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 防静电	表示静电敏感的设备。
 当心触电	表示高压危险。
 激光辐射	表示强激光辐射。
 风扇警告	表示危险运动部件，请远离运动风扇叶片。
 当心机械伤人	表示设备部件机械伤人。
 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	补充	2025.02
V1.0.1	全文优化语言。	2023.02
V1.0.0	首次发布。	2021.06

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
设备 ID	外接的用于监测搜集各种数据信息的设备，该 ID 唯一标示该设备。在接口的结构体中通常描述成 DeviceID。
监测点位	外接设备监测点设置，用唯一 ID 标示。在接口的结构体中通常描述成 ID。
遥信	监测点位的报警信息
遥测	监测点位监测数据上报

目录

前言	I
名词解释	II
第 1 章 内容简介	1
1.1 概述	1
1.2 适用性	1
1.3 动环主机介绍	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 设备登录	6
2.2.1 简介	6
2.2.2 接口总览	7
2.2.3 流程说明	7
2.2.4 示例代码	8
2.3 获取设备列表	10
2.3.1 简介	10
2.3.2 接口总览	10
2.3.3 流程说明	10
2.3.4 示例代码	11
2.4 获取设备点位信息	13
2.4.1 简介	13
2.4.2 接口总览	13
2.4.3 流程说明	13
2.4.4 示例代码	14
2.5 订阅监测点位报警	15
2.5.1 简介	15
2.5.2 接口总览	15
2.5.3 流程说明	15
2.5.4 示例代码	16
2.6 订阅监测点位报警信息	17
2.6.1 简介	17
2.6.2 接口总览	18
2.6.3 流程说明	18
2.6.4 示例代码	19
2.7 订阅普通报警	20
2.7.1 简介	20
2.7.2 接口总览	20
2.7.3 流程说明	20
2.7.4 示例代码	21
第 3 章 接口函数	23

3.1 SDK 初始化	23
3.1.1 SDK 初始化 CLIENT_Init.....	23
3.1.2 SDK 清理 CLIENT_Cleanup	23
3.1.3 设置断线重连回调函数 CLIENT_SetAutoReconnect.....	23
3.1.4 设置网络参数 CLIENT_SetNetworkParam	24
3.2 设备登录	24
3.2.1 用户登录设备 CLIENT_LoginWithHighLevelSecurity	24
3.2.2 用户登出设备 CLIENT_Logout	24
3.3 查询主机所接入的监测设备列表 CLIENT_QueryDevState.....	25
3.4 获取设备点位信息.....	25
3.4.1 获取点位信息	25
3.5 订阅监测点位报警.....	26
3.5.1 订阅监测点位报警 CLIENT_SCADAAlarmAttachInfo.....	26
3.5.2 停止订阅 CLIENT_SCADAAlarmDetachInfo	26
3.6 订阅监测点实时信息	26
3.6.1 订阅监测点位实时信息 CLIENT_SCADAAttachInfo	26
3.6.2 取消订阅点位信息 CLIENT_SCADADetachInfo	27
3.7 报警上报	27
3.7.1 设置报警回调函数 CLIENT_SetDVRMessCallBack	27
3.7.2 订阅报警 CLIENT_StartListenEx	27
3.7.3 停止订阅报警 CLIENT_StopListen	28
第 4 章 回调函数	29
4.1 断线回调函数 fDisConnect	29
4.2 断线重连回调函数 fHaveReConnect.....	29
4.3 数据回调函数 fSCADAAlarmAttachInfoCallBack	29
4.4 报警回调函数 fMessCallBack.....	30
附录 1 法律声明	31
附录 2 网络安全建议	32

第1章 内容简介

1.1 概述

本文档主要介绍 **SDK** 接口参考信息，包括主要功能、接口函数和回调函数。

主要功能包括：**SDK** 初始化、设备登录、获取设备列表、订阅点位信息、订阅监测点位实时信息、订阅设备状态报警等。

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 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	dhconfigsdk.dll	库文件
转码库	StreamConvertor.dll	库文件

- **Linux** 开发包所包含的文件，请参见表 1-2。

表1-2 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	libdhconfigsdk.so	库文件
	dhconfigsdk.h	头文件
转码库	StreamConvertor.so	库文件

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。

1.2 适用性

- 推荐内存：不低于 512M。
- **Jdk** 使用版本：jdk1.6; jdk1.8。
- **SDK** 支持的系统：
 - ◊ **Windows**
Windows 10/Windows 8.1/Windows 7/vista/XP/2000 以及 Windows Server 2008/2003。
 - ◊ **Linux**
Red Hat/SUSE 等通用 Linux 系统。

1.3 动环主机介绍

动环主机是专为动力、环境监控领域设计的一款优秀的数字监控产品。采用嵌入式 **LINUX** 操作系统，系统运行稳定；支持防区报警输入及输出、4~20mA 电流型传感器和 RS485 总线制传感器接入，依托公司视频监控的基础，结合通用的 **H.264** 的视频压缩与 **G.711** 音频压缩技术，实现报警管理、动力环境采集与控制、视频监控、语音对讲与语音投放、网络交换以及光纤环网等为一体的、具有先进的控制技术和强大的网络数据传输能力的全方位监控方案。

产品采用嵌入式设计，安全性高、可靠性好。既可本地独立工作，也可连网组成一个强大的安全监控网，配合使用专业网络视频监控平台（网络）软件，可充分体现其强大的组网和远程监控能力。

可应用于能源、天然气、矿业、电信、电力、农业、交通、智能小区、工厂、仓库、资源、水利设施等各项领域、各部门的安全防范。

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 SDK 清理接口以释放资源。

2.1.2 接口总览

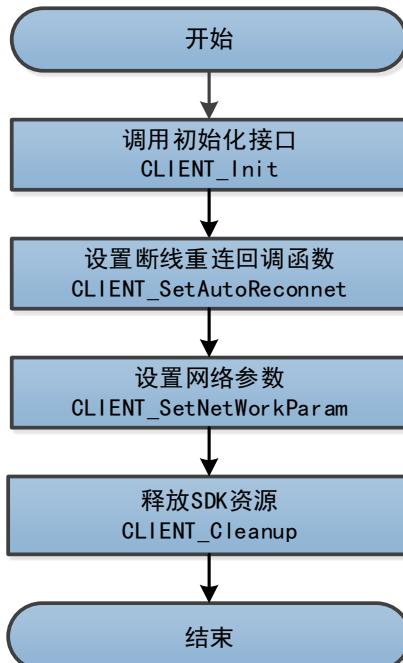
表2-1 SDK 初始化接口信息

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和录像回放业务, 其他业务无法恢复。
- 加载动态库: 如果加载动态库遇到报错如 “`Unable to load library ./wrongpath/libs/win64/dhnetsdk`: 找不到指定的模块”, 通常是路径不匹配, 需要根据报错信息调整动态库的位置或者修改代码。此问题多见于打包整个工程为 `jar` 包提供给其他项目时。由于此问题跟平台和工程使用方式相关, 不能一概而论, 需要具体分析。比如在 `linux` 平台下直接使用工程可以通过以下方式将动态库路径加载到动态库搜索路径中。
 - 1.在终端输入: `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/XXX` 当前终端生效。
 - 2.修改`~/.bashrc` 或`~/.bash_profile`, 最后一行添加 `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/XXX`, 保存之后, 使用 `source .bashrc` 执行该文件, 当前用户生效。
 - 3.修改`/etc/profile`, 添加内容如第 2 条, 同样保存后用 `source` 执行该文件, 所有用户生效。

2.1.4 示例代码

```
import java.io.File;

import main.java.com.netsdk.lib.NetSDKLib;
import main.java.com.netsdk.lib.NetSDKLib.LLong;
import main.java.com.netsdk.lib.ToolKits;

import com.sun.jna.ptr.IntByReference;

/**
 * 登录接口实现
 * 主要有：初始化、登录、登出功能
 */
public class LoginModule {
```

```

public static NetSDKLib netsdk      = NetSDKLib.NETSDK_INSTANCE;
public static NetSDKLib configsdk   = NetSDKLib.CONFIG_INSTANCE;

// 登录句柄
public static LLong m_hLoginHandle = new LLong(0);

private static boolean bInit      = false;
private static boolean bLogopen   = false;

//初始化
public static boolean init(NetSDKLib.fDisConnect      disConnect,
                           NetSDKLib.fHaveReConnect haveReConnect) {
    bInit = netsdk.CLIENT_Init(disConnect, null);
    if(!bInit) {
        System.out.println("Initialize SDK failed");
        return false;
    }

    //打开日志，可选
    NetSDKLib.LOG_SET_PRINT_INFO      setLog      = new
    NetSDKLib.LOG_SET_PRINT_INFO();
    File path = new File("./sdklog/");
    if (!path.exists()) {
        path.mkdir();
    }
    String logPath = path.getAbsolutePath().getParent() + "\\sdklog\\" + ToolKits.getDate()
+ ".log";
    setLog.nPrintStrategy = 0;
    setLog.bSetFilePath = 1;
    System.arraycopy(logPath.getBytes(),      0,      setLog.szLogFilePath,      0,
logPath.getBytes().length);
    System.out.println(logPath);
    setLog.bSetPrintStrategy = 1;
    bLogopen = netsdk.CLIENT_LogOpen(setLog);
    if(!bLogopen ) {
        System.err.println("Failed to open NetSDK log");
    }

    // 设置断线重连回调接口, 设置过断线重连成功回调函数后, 当设备出现断线情况, SDK

```

```

内部会自动进行重连操作
    // 此操作为可选操作，但建议用户进行设置
    netsdk.CLIENT_SetAutoReconnect(haveReConnect, null);

    //设置登录超时时间和尝试次数，可选
    int waitTime = 5000; //登录请求响应超时时间为 5S
    int tryTimes = 1;    //登录时尝试建立链接 1 次
    netsdk.CLIENT_SetConnectTime(waitTime, tryTimes);

    // 设置更多网络参数，NET_PARAM 的 nWaittime，nConnectTryNum 成员与
    CLIENT_SetConnectTime

    // 接口设置的登录设备超时时间和尝试次数意义相同,可选
    NetSDKLib.NET_PARAM netParam = new NetSDKLib.NET_PARAM();
    netParam.nConnectTime = 10000;      // 登录时尝试建立链接的超时时间
    netParam.nGetConnInfoTime = 3000;   // 设置子连接的超时时间
    netsdk.CLIENT_SetNetworkParam(netParam);

    return true;
}

//清除环境
public static void cleanup() {
    if(bLogopen) {
        netsdk.CLIENT_LogClose();
    }

    if(bInit) {
        netsdk.CLIENT_Cleanup();
    }
}
}

```

2.2 设备登录

2.2.1 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。

2.2.2 接口总览

表2-2 设备登录接口信息

接口	说明
CLIENT_LoginWithHighLevelSecurity	高安全级别登录接口
CLIENT_Logout	登出接口

2.2.3 流程说明

登录业务流程如图 2-2 所示。

图2-2 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。
- **句柄重复：**登录句柄有可能与存在过的句柄相同，属于正常现象。例如登录设备 A 获得 loginID，

将 loginID 注销，再次进行登录操作，可能又获取到 loginID。但是在句柄的整个生命周期内，不会出现重复的句柄。

- 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。常见错误码请参见表 2-3。
- 多设备登录：SDK 初始化后，可以登录多台设备，但是相应的登录句柄、登录信息需要调整。

表2-3 常见错误码及含义

error 的错误码	含义
1	密码不正确
2	用户名不存在
3	登录超时。规避示例代码如下： NET_PARAM stuNetParam = new NET_PARAM(); stuNetParam.nWaittime = 8000; // unit ms CLIENT_SetNetworkParam (stuNetParam);
4	账号已登录
5	账号已被锁定
6	账号被列为禁止名单
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

2.2.4 示例代码

```
import java.io.File;

import main.java.com.netsdk.lib.NetSDKLib;
import main.java.com.netsdk.lib.NetSDKLib.LLong;
import main.java.com.netsdk.lib.ToolKits;

import com.sun.jna.ptr.IntByReference;

public class LoginModule {

    public static NetSDKLib netsdk      = NetSDKLib.NETSDK_INSTANCE;
    public static NetSDKLib configsdk   = NetSDKLib.CONFIG_INSTANCE;
    //SDK 初始化， SDK 清理省略
    // 设备信息
```

```

    public static NetSDKLib.NET_DEVICEINFO_Ex m_stDeviceInfo = new
NetSDKLib.NET_DEVICEINFO_Ex();
    //登录句柄
    public static LLong m_hLoginHandle = new LLong(0);

    //登录设备
    public static boolean login(String m_strIp, int m_nPort, String m_strUser, String
m_strPassword) {
        //入参
        NET_IN_LOGIN_WITH_HIGHLVEL_SECURITY pstInParam=
new NET_IN_LOGIN_WITH_HIGHLVEL_SECURITY();
        pstInParam.szIP= m_strIp;
        pstInParam.nport= m_nPort;
        pstInParam.szUserName= m_strUser;
        pstInParam.szPassword= m_strPassword;
        //出参
        NET_OUT_LOGIN_WITH_HIGHLVEL_SECURITY pstOutParam=
new NET_OUT_LOGIN_WITH_HIGHLVEL_SECURITY();
        m_hLoginHandle =
netsdk.CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIGHLVEL_SECURIT
Y pstInParam, NET_OUT_LOGIN_WITH_HIGHLVEL_SECURITY pstOutParam);

        if(m_hLoginHandle.longValue() == 0) {
            System.err.printf("Login Device[%s] Port[%d]Failed. %s\n", m_strIp, m_nPort,
ToolKits.getErrorCodePrint());
        } else {
            System.out.println("Login Success ");
        }

        return m_hLoginHandle.longValue() == 0? false:true;
    }

    //登出设备
    public static boolean logout() {
        if(m_hLoginHandle.longValue() == 0) {
            return false;
        }

        boolean bRet = netsdk.CLIENT_Logout(m_hLoginHandle);
        if(bRet) {

```

```
    m_hLoginHandle.setValue(0);
}

return bRet;
}
```

2.3 获取设备列表

2.3.1 简介

获取主机上连接的设备的 ID。

2.3.2 接口总览

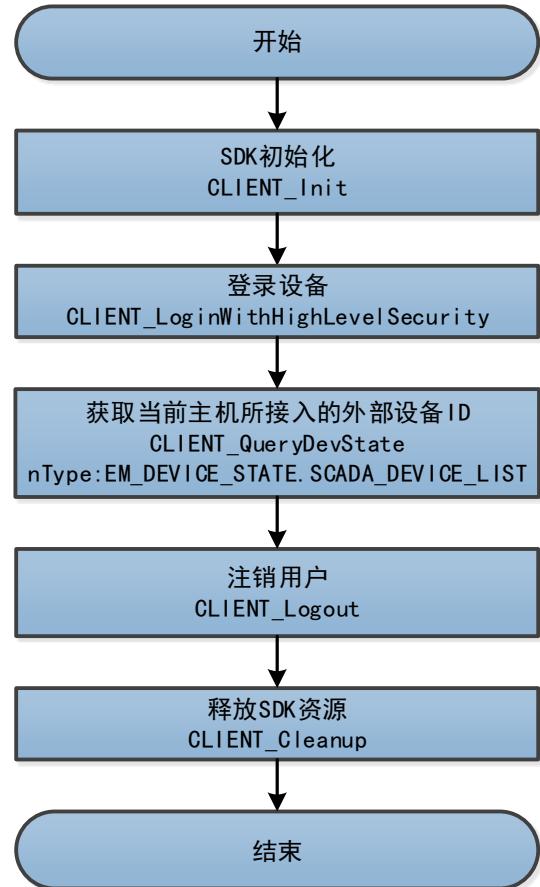
表2-4 获取设备列表接口信息

接口	说明
CLIENT_QueryDevState	获取当前主机所接入的外部设备 ID

2.3.3 流程说明

NetSDK 获取设备列表流程如图 2-3 所示。

图2-3 获取设备列表业务流程



流程说明

- 步骤1 调用 `CLIENT_Init` 完成 NetSDK 初始化流程。
- 步骤2 调用 `CLIENT_LoginWithHighLevelSecurity` 登录设备。
- 步骤3 调用 `CLIENT_QueryDevState` 获取设备列表，参数 `nType` 值
`EM_DEVICE_STATE_SCADA_DEVICE_LIST` 对应结构体
`NET_SCADA_DEVICE_LIST`。
- 步骤4 业务使用完后，调用 `CLIENT_Logout` 登出设备。
- 步骤5 NetSDK 功能使用完后，调用 `CLIENT_Cleanup` 释放 NetSDK 资源。

2.3.4 示例代码

```
/*
 * 获取当前主机接入的外部设备 ID
 */
public boolean queryDeviceIdList() {
    m_lstDeviceID.clear(); // 清空外部设备 ID 列表

    int nCount = deviceinfo.byChanNum; // 设备的通道总数

    NET_SCADA_DEVICE_ID_INFO[] stuDeviceIDList = new
```

```

NET_SCADA_DEVICE_ID_INFO[nCount];
    for (int i = 0; i < stuDeviceIDList.length; ++i) {
        stuDeviceIDList[i] = new NET_SCADA_DEVICE_ID_INFO();
    }

    NET_SCADA_DEVICE_LIST          stuSCADADeviceInfo      =      new
NET_SCADA_DEVICE_LIST();
    stuSCADADeviceInfo.nMax = nCount;
    int nSize = stuDeviceIDList[0].size() * nCount;
    stuSCADADeviceInfo.pstuDeviceIDInfo = new Memory(nSize); // 监测设备信息
    stuSCADADeviceInfo.pstuDeviceIDInfo.clear(nSize);
    ToolKits.SetStructArrToPointerData(stuDeviceIDList,
stuSCADADeviceInfo.pstuDeviceIDInfo);

    if           (queryDevState(NetSDKLib.NET_DEVSTATE_SCADA_DEVICE_LIST,
stuSCADADeviceInfo)) {
        System.err.println(" 获取当前主机接入的外部设备 ID 失败 " +
ToolKits.getErrorCode()); // 接口调用失败
        return false;
    }

    if (stuSCADADeviceInfo.nRet == 0) {
        System.out.println(" 当前主机接入的外部设备 ID 有效个数为 0.");
// 外部设备没有有效 ID
        return false;
    }

    // 从 Pointer 提取数据
    ToolKits.GetPointerDataToStructArr(stuSCADADeviceInfo.pstuDeviceIDInfo,
stuDeviceIDList);
    // 打印数据并更新设备 ID
    System.out.println(" 获取当前主机接入的外部设备 ID 的有效个数 : " +
stuSCADADeviceInfo.nRet);
    for (int i = 0; i < stuSCADADeviceInfo.nRet; ++i) {
        String deviceID = "";
        try {
            System.out.printf("外部设备[%d] 设备 id[%s] 设备名称[%s]\n",
new String(stuDeviceIDList[i].szDeviceID, encode).trim(),
new String(stuDeviceIDList[i].szDevName, encode).trim());
            deviceID = new String(stuDeviceIDList[i].szDeviceID, encode).trim();
        } catch (UnsupportedEncodingException e) {

```

```
        e.printStackTrace();
    }
    m_lstDeviceID.add(deviceID); // 更新外部设备 ID 列表
}
return true;
}
```

2.4 获取设备点位信息

2.4.1 简介

针对动环主机上的设备，按照各个设备 ID 获取对应的监测点位信息。

2.4.2 接口总览

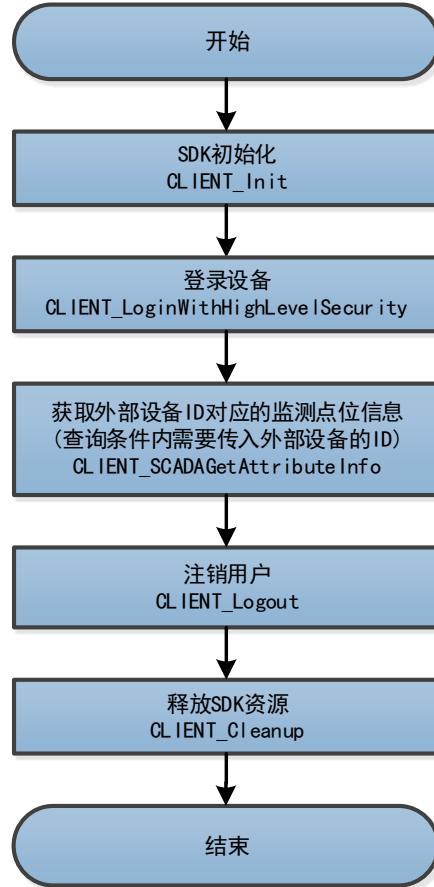
表2-5 获取监测点位信息的接口信息

接口	说明
CLIENT_SCADAGetAttributeInfo	获取设备点位信息

2.4.3 流程说明

接口调用流程如图 2-4 所示。

图2-4 获取设备点位信息业务流程



流程说明

- 步骤1 调用 NETClient_Init 完成 NetSDK 初始化流程。
 - 步骤2 初始化成功后，调用 NETClient_LoginWithHighLevelSecurity 登录设备。
 - 步骤3 调用 CLIENT_SCADAGetAttributeInfo，获取外部设备 ID 对应下的监测点位信息。其中查询条件 stuCondition 内的 szDeviceID 传入 CLIENT_QueryDevState 获取到的设备 ID。
 - 步骤4 业务使用完后，调用 CLIENT_Logout 登出设备。
- NetSDK 功能使用完后，调用 CLIENT_Cleanup 释放 NetSDK 资源。

2.4.4 示例代码

```
public void getDeviceAttribute(String devicId){  
    NET_IN_SCADA_GET_ATTRIBUTE_INFO inParam = new  
    NET_IN_SCADA_GET_ATTRIBUTE_INFO();  
    NET_OUT_SCADA_GET_ATTRIBUTE_INFO outParam = new  
    NET_OUT_SCADA_GET_ATTRIBUTE_INFO();  
    NET_GET_CONDITION_INFO conditionInfo = new NET_GET_CONDITION_INFO();  
    conditionInfo.szDeviceID = devicId.getBytes();  
    outParam.nMaxAttributeInfoNum = 20;  
    NET_ATTRIBUTE_INFO attributeInfo = new NET_ATTRIBUTE_INFO();  
    attributeInfo.write();
```

```

        outParam.pstuAttributeInfo = attributeInfo.getPointer();
        inParam.write();
        outParam.write();
        boolean ret = netsdkApi.CLIENT_SCADAGetAttributeInfo(m_hLoginHandle,
inParam.getPointer(), outParam.getPointer(), 3000);
        if (ret){
            System.out.println("SCADAGetAttributeInfo succeed!");
            outParam.read();
            int retAttributeInfoNum = outParam.nRetAttributeInfoNum;
            for (int i = 0; i < retAttributeInfoNum; i++) {
                NET_ATTRIBUTE_INFO out = new NET_ATTRIBUTE_INFO();
                ToolKits.GetPointerDataToStruct(outParam.pstuAttributeInfo, 0, out);
                System.out.println(out.emStatus);
            }
        }
    }
}

```

2.5 订阅监测点位报警

2.5.1 简介

监测各个监测点位的报警信息。

2.5.2 接口总览

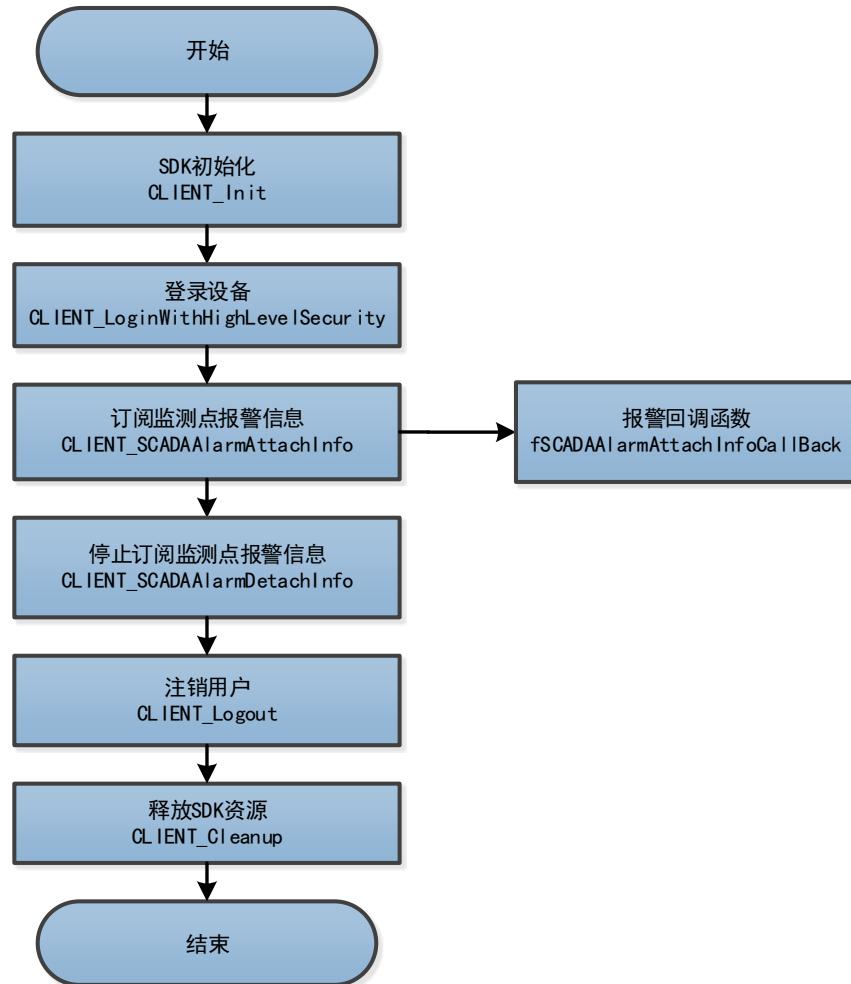
表2-6 监测点位报警接口信息

接口	说明
CLIENT_SCADAAlarmAttachInfo	订阅监测点位报警信息
CLIENT_SCADAAlarmDetachInfo	取消订阅监测点位报警信息

2.5.3 流程说明

监测点位报警流程如图 2-5 所示。

图2-5 监测点报警信息流程



流程说明

- 步骤1 调用 `CLIENT_Init` 完成 NetSDK 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_SCADAAlarmAttachInfo` 函数，向设备订阅报警。对应输入输出结构体分别为：NET_IN_SCADA_ALARM_ATTACH_INFO 和 NET_OUT_SCADA_ALARM_ATTACH_INFO。订阅成功后，设备上报的报警事件通过 `NET_IN_SCADA_ALARM_ATTACH_INFO` 结构体内设置的回调函数 `cbCallBack` 通知用户。
- 步骤4 报警上报功能使用完毕后，调用 `CLIENT_SCADAAlarmDetachInfo` 函数停止向设备订阅报警。
- 步骤5 业务使用完后，调用 `CLIENT_Logout` 登出设备。
- 步骤6 NetSDK 功能使用完后，调用 `CLIENT_Cleanup` 释放 NetSDK 资源。

2.5.4 示例代码

```
/**  
 * 订阅监测点位报警信息  
 */
```

```
public void scadaAlarmAttach() {
    // 入参
    NET_IN_SCADA_ALARM_ATTACH_INFO stIn = new
    NET_IN_SCADA_ALARM_ATTACH_INFO();
    stIn.cbCallBack = SCADAAlarmAttachInfoCallBack.getInstance();

    // 出参
    NET_OUT_SCADA_ALARM_ATTACH_INFO stOut = new
    NET_OUT_SCADA_ALARM_ATTACH_INFO();

    m_hAlarmAttachHandle = netsdkApi.CLIENT_SCADAAlarmAttachInfo(m_hLoginHandle, stIn, stOut, 3000);
    if (m_hAlarmAttachHandle.longValue() != 0) {
        System.out.println("订阅监测点位报警信息成功！");
    } else {
        System.err.println("订阅监测点位报警信息失败！ " + ToolKits.getErrorCode());
    }
}

/**
 * 取消订阅监测点位报警信息
 */
public void scadaAlarmDetachInfo() {
    if (m_hAlarmAttachHandle.longValue() != 0) {
        netsdk.CLIENT_SCADAAlarmDetachInfo(m_hAlarmAttachHandle);
        alarmAttachInfoHandle.setValue(0);
    }
}
```

2.6 订阅监测点位报警信息

2.6.1 简介

监测各个监测点位信息上报。

2.6.2 接口总览

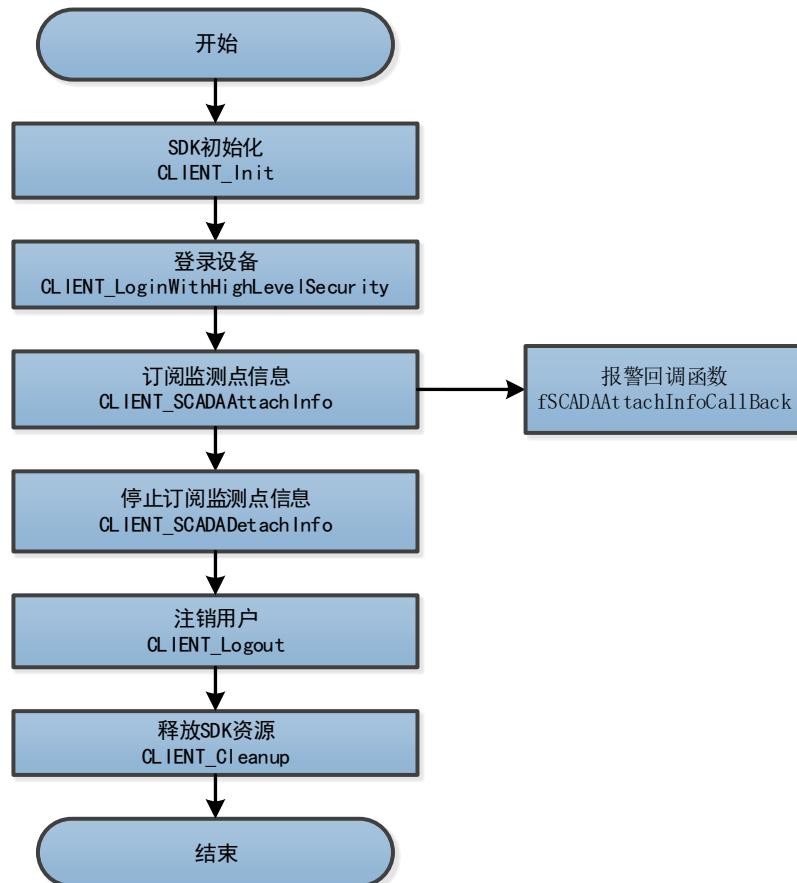
表2-7 监测点位信息

接口	说明
CLIENT_SCADAAttachInfo	订阅监测点位信息
CLIENT_SCADADetachInfo	取消监测点位信息订阅

2.6.3 流程说明

监测点位信息流程如图 2-6 所示。

图2-6 监测点位信息流程图



流程说明

- 步骤1 调用 `CLIENT_Init` 完成 NetSDK 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_SCADAAttachInfo` 函数，向设备订阅报警。对应输入输出结构体分别为：`NET_IN_SCADA_ATTACH_INFO` 和 `NET_OUT_SCADA_ATTACH_INFO`。订阅成功后，设备上报的报警事件通过 `NET_IN_SCADA_ATTACH_INFO` 结构体内设置的回调函数 `cbCallBack` 通知用户。
- 步骤4 报警上报功能使用完毕后，调用 `CLIENT_SCADADetachInfo` 函数停止向设备订阅报警。
- 步骤5 业务使用完后，调用 `CLIENT_Logout` 登出设备。
- 步骤6 NetSDK 功能使用完后，调用 `CLIENT_Cleanup` 释放 NetSDK 资源。

2.6.4 示例代码

```
/*
 * 订阅监测点位实时数据
 */
public void case_scada_real_attach() {
    // 入参
    NET_IN_SCADA_ATTACH_INFO stIn = new NET_IN_SCADA_ATTACH_INFO();
    stIn.cbCallBack = SCADAAttachInfoCallBack.getInstance();
    stIn.emPointType =
EM_NET_SCADA_POINT_TYPE.EM_NET_SCADA_POINT_TYPE_ALL;

    // 出参
    NET_OUT_SCADA_ATTACH_INFO          stOut          =
new
NET_OUT_SCADA_ATTACH_INFO();

    m_hAttachHandle = netsdkApi.CLIENT_SCADAAttachInfo(m_hLoginHandle, stIn,
stOut, 3000);
    if (0 != m_hAttachHandle.longValue()) {
        System.out.println("订阅监测点位实时数据信息成功!");
    } else {
        System.err.println("订阅监测点位实时数据信息失败" + ToolKits.getErrorCode());
    }
}

/*
 * 取消监测点位实时数据
 */
public void case_scada_real_detach() {
    if (m_hAttachHandle.longValue() != 0) {
        netsdkApi.CLIENT_SCADADetachInfo(m_hAttachHandle);
        m_hAttachHandle.setValue(0);

    }
}
```

2.7 订阅普通报警

2.7.1 简介

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

2.7.2 接口总览

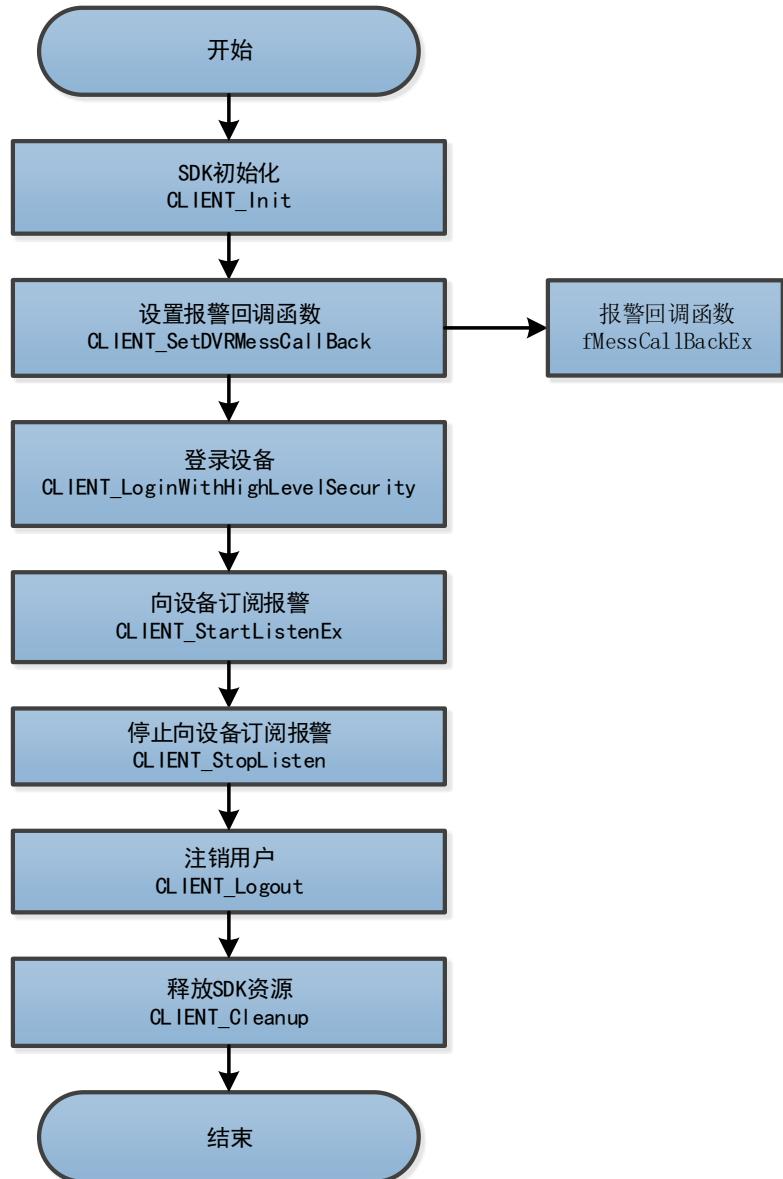
表2-8 报警上报的接口信息

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

2.7.3 流程说明

报警上报流程如图 2-7 所示。

图2-7 报警上报流程



流程说明

- 步骤1 调用 `CLIENT_Init` 函数，完成 NetSDK 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 NetSDK 功能使用完后，调用 `CLIENT_Cleanup` 函数释放 NetSDK 资源。

2.7.4 示例代码

```
/**
```

```
* 订阅报警信息
*/
public boolean case_StartListenEx() {
    if (bListening) {
        return true;
    }

    // 订阅报警 (报警回调函数已经在 initSdk 中设置)
    bListening = netsdkApi.CLIENT_StartListenEx(m_hLoginHandle);
    if (!bListening) {
        System.err.println(" 订 阅 报 警 失 败 !   LastError = 0x%x\n" +
netsdkApi.CLIENT_GetLastError());
    } else {
        System.out.println("订阅报警成功.");
    }
    return bListening;
}

/**
 * 取消订阅报警信息
*/
public void case_stopListen() {
    if (bListening) {
        System.out.println("取消订阅报警信息.");
        netsdkApi.CLIENT_StopListen(m_hLoginHandle);
        bListening = false;
    }
}
```

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	
方法	<code>public boolean CLIENT_Init(Callback cbDisConnect, Pointer dwUser);</code>	
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	<ul style="list-style-type: none">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	
方法	<code>public void CLIENT_Cleanup();</code>	
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	
方法	<code>public void CLIENT_SetAutoReconnect(Callback cbAutoConnect, Pointer dwUser);</code>	
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	
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	
方法	<code>public void CLIENT_SetNetworkParam(NET_PARAM pNetParam);</code>	
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 设备登录

3.2.1 用户登录设备 CLIENT_LoginWithHighLevelSecurity

表3-5 用户登录设备 CLIENT_LoginWithHighLevelSecurity

选项	说明	
描述	用户登录设备	
方法	<code>public LLong CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIGHELEVEL_SECURITY pstInParam, NET_OUT_LOGIN_WITH_HIGHELEVEL_SECURITY pstOutParam);</code>	
参数	[in]pstInParam	输入参数
	[out]pstOutParam	输出参数
返回值	成功返回登录句柄，失败返回 0	
说明	此方法封装在 NetSDKLib 接口中，通常通过以下方式调用： <code>m_hLoginHandle = netsdk.CLIENT_CLIENT_LoginWithHighLevelSecurity(pstInParam, pstOutParam);</code>	

3.2.2 用户登出设备 CLIENT_Logout

表3-6 用户登出设备 CLIENT_Logout

选项	说明	
描述	用户登出设备	
方法	<code>public boolean CLIENT_Logout(LLong ILoginID);</code>	
参数	[in]ILoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	成功返回 TRUE，失败返回 FALSE	
说明	此方法封装在 NetSDKLib 接口中，通常通过以下方式调用 <code>netsdk.CLIENT_Logout(m_hLoginHandle);</code>	

3.3 查询主机所接入的监测设备列表 CLIENT_QueryDevState

表3-7 查询接入的监测设备列表 CLIENT_QueryDevState

选项	说明	
描述	查询主机所接入的监测设备信息	
方法	<code>public boolean CLIENT_QueryDevState(LLong ILoginID, int nType, Pointer pBuf, int nBufLen, IntByReference pRetLen, int waittime);</code>	
参数	[in]ILoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]nType	查询信息类型, 这里对应 NET_DEVSTATE_SCADA_DEVICE_LIST
	[out]pBuf	用户申请的内存
	[in]nBufLen	申请内存大小
	[out]pRetLen	输出参数, 用于接收查询返回的数据的缓存, 这里 对应 NET_SCADA_DEVICE_LIST 类型。
	[in]waittime	获取超时时间
返回值	成功返回 TRUE, 失败返回 FALSE	
说明	查询主机所接入的监测设备信息	

3.4 获取设备点位信息

3.4.1 获取点位信息

表3-8 获取点位阈值 CLIENT_SCADAGetAttributeInfo

选项	说明	
描述	获取点位阈值	
方法	<code>public boolean CLIENT_SCADAGetAttributeInfo(LLong ILoginID, Pointer pInParam, Pointer pOutParam, int nWaitTime);</code>	
参数	[in]ILoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]pInParam	输入参数, 对应 NET_IN_SCADA_GET_ATTRIBUTE_INFO 类型
	[out]pOutParam	输出参数, 对应 NET_OUT_SCADA_GET_ATTRIBUTE_INFO 类型
	[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	无	

3.5 订阅监测点位报警

3.5.1 订阅监测点位报警 CLIENT_SCADAAlarmAttachInfo

表3-9 开启订阅 CLIENT_SCADAAlarmAttachInfo

选项	说明	
描述	开始监测点位报警订阅	
方法	<code>public LLong CLIENT_SCADAAlarmAttachInfo(LLong ILoginID, NET_IN_SCADA_ALARM_ATTACH_INFO pInParam, NET_OUT_SCADA_ALARM_ATTACH_INFO pOutParam, int nWaitTime);</code>	
参数	[in] ILoginID	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 <code>LLONG</code> 类型订阅句柄，失败返回 0	
说明	接口返回失败，请使用 <code>CLIENT_GetLastError</code> 获取错误码	

3.5.2 停止订阅 CLIENT_SCADAAlarmDetachInfo

表3-10 停止订阅 CLIENT_SCADAAlarmDetachInfo

选项	说明	
描述	停止监测点位报警订阅	
方法	<code>public boolean CLIENT_SCADAAlarmDetachInfo(LLong IAttachHandle);</code>	
参数	[in] IAttachHandle	监测点位报警句柄
返回值	BOOL 类型 ● 成功：TRUE ● 失败：FALSE	
说明	接口返回失败，请使用 <code>CLIENT_GetLastError</code> 获取错误码	

3.6 订阅监测点实时信息

3.6.1 订阅监测点位实时信息 CLIENT_SCADAAttachInfo

表3-11 订阅监测点位实时信息 CLIENT_SCADAAttachInfo

选项	说明	
描述	订阅监测点位实时信息	
函数	<code>public LLong CLIENT_SCADAAttachInfo(LLong ILoginID, NET_IN_SCADA_ATTACH_INFO pInParam, NET_OUT_SCADA_ATTACH_INFO pOutParam, int nWaitTime);</code>	
参数	[in] ILoginID	LoginWithHighLevelSecurity 返回值
	[in] pInParam	订阅输入参数

选项	说明	
	[out]pOutParam	订阅输出参数
	[in]waittime	等待时间
返回值	失败返回 0, 成功返回非 0 的值	
说明	无	

3.6.2 取消订阅点位信息 CLIENT_SCADADetachInfo

表3-12 取消订阅点位信息 CLIENT_SCADADetachInfo

选项	说明	
描述	取消订阅点位信息	
函数	<code>public boolean CLIENT_SCADADetachInfo(LLong IAttachHandle);</code>	
参数	[in]IAttachHandle	CLIENT_SCADAAttachInfo 返回值
返回值	失败返回 false, 成功返回 true	
说明	无	

3.7 报警上报

3.7.1 设置报警回调函数 CLIENT_SetDVRMessCallBack

表3-13 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	<code>public void CLIENT_SetDVRMessCallBack(Callback cbMessage, Pointer dwUser);</code>	
参数	[in]cbMessage	<ul style="list-style-type: none"> ● 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	<ul style="list-style-type: none"> ● 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, NetSDK 默认不回调 ● 此回调函数 fMessCallBack 必须先调用报警消息订阅接口 StartListenEx 才生效 	

3.7.2 订阅报警 CLIENT_StartListenEx

表3-14 订阅报警 CLIENT_StartListenEx

选项	说明	
描述	订阅报警	
函数	<code>public boolean CLIENT_StartListenEx(LLong ILoginID);</code>	
参数	[in]ILoginID	LoginWithHighLevelSecurity 返回值
返回值	失败返回 false, 成功返回 true	
说明	订阅设备消息, 得到的消息从 SetDVRMessCallBack 的设置值回调出来	

3.7.3 停止订阅报警 CLIENT_StopListen

表3-15 停止订阅报警 CLIENT_StopListen

选项	说明
描述	停止订阅报警
函数	public boolean CLIENT_StopListen(LLong ILoginID);
参数	[in]ILoginID LoginWithHighLevelSecurity 返回值
返回值	失败返回 false, 成功返回 true
说明	无

第4章 回调函数

4.1 断线回调函数 fDisConnect

表4-1 断线回调函数 fDisConnect

选项	说明	
描述	断线回调函数	
接口及方法	<pre>public interface fDisConnect extends StdCallCallback { public void invoke(LLong lLoginID, String pchDVRIP, int nDVRPort, Pointer dwUser); }</pre>	
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	

4.2 断线重连回调函数 fHaveReConnect

表4-2 断线重连回调函数 fHaveReConnect

选项	说明	
描述	断线重连回调函数	
接口及方法	<pre>public interface fHaveReConnect extends StdCallCallback { public void invoke(LLong lLoginID, String pchDVRIP, int nDVRPort, Pointer dwUser); }</pre>	
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	

4.3 数据回调函数 fSCADAAlarmAttachInfoCallBack

表4-3 回调函数 fSCADAAlarmAttachInfoCallBack

选项	说明	
描述	遥测回调函数	
接口及方法	<pre>public interface fSCADAAlarmAttachInfoCallBack extends Callback { public void invoke(LLong lAttachHandle, NET_SCADA_NOTIFY_POINT_ALARM_INFO_LIST pInfo, int nBufLen, Pointer dwUser); }</pre>	
参数	[out] lAttachHandle	CLIENT_SCADAAttachInfo 的返回值
	[out] pInfo	报警数据信息

选项	说明	
	[out]pBuffer	预览数据块地址
	[out]dwUser	回调函数的用户参数
返回值	无	
说明	无	

4.4 报警回调函数 fMessCallBack

表4-4 报警回调函数 fMessCallBack

选项	说明	
描述	报警回调函数	
函数	<pre>public interface fMessCallBack extends Callback{ public boolean invoke(int ICommand, LLong ILoginID, Pointer pStuEvent, int dwBufLen, String strDeviceIP, NativeLong nDevicePort, Pointer dwUser); }</pre>	
参数	[out]ILoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[out]ICommand	监测采集设备报警事件(对应结构体 ALARM_SCADA_DEV_INFO)
	[out]pStuEvent	监测采集设备报警事件
	[out]dwBufLen	pBuf 的长度, 单位: 字节
	[out]strDeviceIP	设备 IP
	[out]nDevicePort	端口
	[out]dwUser	回调函数的用户参数
返回值	无	
说明	无	

附录1 法律声明

商标声明

-  : 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。