NetSDK_C# Programming Manual (Field Surveillance Unit)

User's Manual

V1.0.0

Foreword

Safety Instructions

The following categorized signal words with defined meaning might appear in the manual.

Signal Words	Meaning
⚠ **DANGER**	Indicates a high potential hazard which, if not avoided, will result in death or serious injury.
⚠ **WARNING**	Indicates a medium or low potential hazard which, if not avoided, could result in slight or moderate injury.
⚠ **CAUTION**	Indicates a potential risk which, if not avoided, could result in property damage, data loss, lower performance, or unpredictable result.
TIPS	Provides methods to help you solve a problem or save you time.
NOTE	Provides additional information as the emphasis and supplement to the text.

Revision History

Version	Revision Content	Release Time
V1.0.0	First release.	April 2021

Glossary

This chapter provides the definitions to some of the terms appearing in the Manual to help you understand the function of each module.

Term	Description
Device ID	This ID uniquely identifies an external device for the monitoring and collection of various data. It is described as DeviceID in the interface structure.
Monitoring Point	Functional abstraction that monitors a certain type of values on the external device, identified by a unique ID. It is described as ID in the interface structure.
Remote Signaling	Point alarm information.
Telemetering	Point monitoring data upload.

Table of Contents

1 Introduction

1.1 Overview

This document provides reference information of the packaging engineering NetSDKCS for the C# NetSDK library, including main functions, interface functions, and callback.

The main functions include: NetSDK initialization, device login, getting external device list, getting point information, subscription to monitoring point alarm, subscription to real-time information of monitoring point, and subscription to device status alarm.

- For files included in the C# NetSDK library, see Table 1-1.

Table 1-1 Files in the NetSDK library

Library Type	Library File Name	Description
Function Library	dhnetsdk.dll	Library file
	avnetsdk.dll	Library file
Configuration Library	dhconfigsdk.dll	Library file
Playing (Encoding/Decoding) Auxiliary Library	dhplay.dll	Play library
	fisheye.dll	Fisheye dewarping library
dhnetsdk Auxiliary Library	IvsDrawer.dll	Image display library
	StreamConvertor.dll	Transcoding library

- For files included in the C# packaging engineering, see Table 1-2.

Table 1-2 Files in the NetSDKCS engineering

File Name	File Description
NetSDK.cs	For packaging C# interfaces to be called by customers
NetSDKStruct.cs	For storing the used structure enumeration.
OriginalSDK.cs	For introducing C interfaces in the NetSDK library into the C# engineering.

- The function library and configuration library of NetSDK are necessary libraries.
- The function library is the main body of NetSDK, which is used for communication interaction between client and products, remote control, search, configuration, acquisition and processing of stream data.
- NetSDK library is the foundation of NetSDKCS engineering. The reference path of NetSDK library is defined in the OriginalSDK.cs file. In actual use, put the NetSDK library in the corresponding path. Users can customize the reference path.
- Customers can directly reference this packaging engineering in their own engineering, or put the files in the packaging engineering in their own engineering for use, or refer to this packaging engineering for packaging.
- This document introduces the use of C# engineering that packages the C library interface. For more information, see the development document in the C NetSDK library.

1.2 System Requirements

- Recommended memory: No less than 512 M.
- Systems supported by NetSDK: Windows 10/Windows 8.1/Windows 7/Windows Vista/Windows Server 2008/2003.

1.3 Field Surveillance Unit

Field surveillance unit is an excellent digital surveillance product designed for power and environment surveillance. The unit uses the embedded stable LINUX operating system.

- It supports protection zone alarm input and output, and the access of 4 mA–20 mA current sensor and RS-485 bus-based sensor.
- Based on video surveillance and the general H.264 video compression technology and G.711 audio compression technology, an all-around surveillance solution with advanced control technologies and powerful network data transmission capabilities is provided to integrate alarm management, power and environment acquisition and control, video surveillance, voice talk and audio advertisement, network switching, and optical fiber ring network.
- With an embedded design, the unit has high security and reliability.
- The unit can work independently and locally or be connected to the network to form a powerful security monitoring network.
- Along with professional network video monitoring platform (network) software, the unit has powerful networking and remote monitoring capabilities.

1.4 Scenes

The unit can be applied to the security in various fields and departments such as energy, natural gas, mining, telecommunication, power, agriculture, transportation, intelligent communities, factories, warehouses, resources, and water conservancy facilities.

2 Main Functions

2.1 Initializing NetSDK

2.1.1 Introduction

Initialization is the first step for NetSDK to conduct all the function modules. It does not have the surveillance function but can set some parameters that affect the SDK overall functions.

- Initialization occupies some memory.
- Only the first initialization is valid within one process.
- After initialization, call the NetSDK cleaning up interface to release resource.

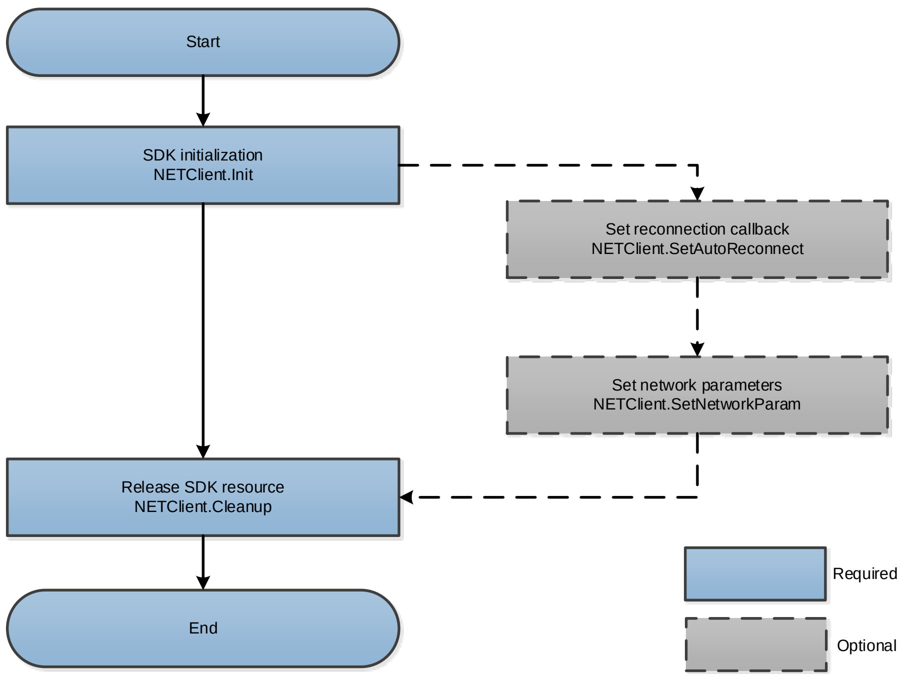
2.1.2 Interface Overview

Table 2-1 Description of NetSDK initialization interfaces

Interface	Description
NETClient.Init	NetSDK initialization interface
NETClient.Cleanup	NetSDK cleaning up interface
NETClient.SetAutoReconnect	Setting of reconnection callback interface
NETClient.SetNetworkParam	Setting of login network environment interface

2.1.3 Process Description

Figure 2-1 Process of NetSDK initialization



Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.

Step 2 (Optional) Call **NETClient.SetAutoReconnect** to set reconnection callback to allow the auto reconnecting after disconnection within NetSDK.

Step 3 (Optional) Call **NETClient.SetNetworkParam** to set network login parameters, which include the timeout period for device login and the number of attempts.

Step 4 After using all functions of NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

Notes

- You need to call the interfaces **NETClient.Init** and **NETClient.Cleanup** in pairs. NetSDK supports single-thread multiple calling in pairs, but it is suggested to call the pair for only one time globally.
- Initialization: Calling **NETClient.Init** multiple times is only for internal count without repeating requesting resources.
- Cleaning up: The interface **NETClient.Cleanup** clears all the opened processes, such as login, real-time monitoring, and alarm subscription.
- Reconnection: NetSDK can set the reconnection function for the situations such as network disconnection and power off. NetSDK will keep logging in until succeeded. Only the real-time monitoring, playback, subscription to intelligent event, and alarm subscription can be restored after reconnection.

2.1.4 Example Code

```
//Declare static callback delegation (an error of releasing before callback might occur for ordinary
delegation)
private static fDisConnectCallBack m_DisConnectCallBack;        // Disconnection callback
private static fHaveReConnectCallBack m_ReConnectCallBack;    // Reconnection callback

// Commission
m_DisConnectCallBack = new fDisConnectCallBack(DisConnectCallBack);
m_ReConnectCallBack = new fHaveReConnectCallBack(ReConnectCallBack);

//Initialize NetSDK. Disconnection callback is implemented during initialization.
bool result = NETClient.Init(m_DisConnectCallBack, IntPtr.Zero, null);
if (!result)
{
MessageBox.Show(NETClient.GetLastError());//Display error message.
return;
}

//Set reconnection callback.
NETClient.SetAutoReconnect(m_ReConnectCallBack, IntPtr.Zero);

// Set network parameters.
```

```
NET_PARAM param = new NET_PARAM()
{
nWaittime = 10000,// Waiting timeout period (ms)
nConnectTime = 5000,// Connection timeout period (ms)
};
NETClient.SetNetworkParam(param);

// Clean up initialization resource.
NETClient.Cleanup();
```

2.2 Device Login and Logout

2.2.1 Introduction

Device login, also called user authentication, is the precondition of all the other function modules.

You can obtain a unique login ID upon logging in to the device and should pass in the login ID before using other NetSDK interfaces. The login ID becomes invalid once logged out.

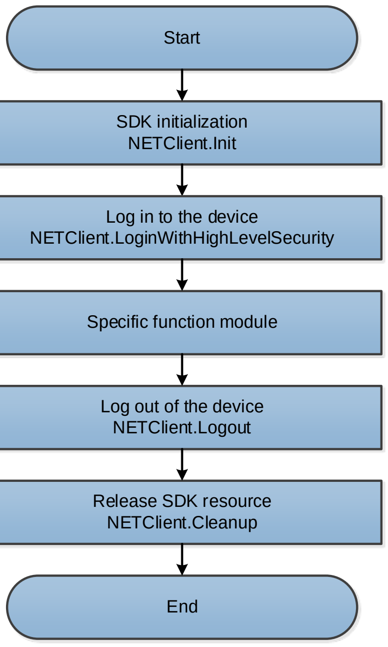
2.2.2 Interface Overview

Table 2-2 Description of device login interfaces

Interface	**Description**
NETClient.LoginWithHighLevelSecurity	Login interface
NETClient.Logout	Logout interface

2.2.3 Process Description

Figure 2-2 Process of login



Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.
Step 2 Call **NETClient.LoginWithHighLevelSecurity** to log in to the device.
Step 3 After successful login, you can realize the required function module.
Step 4 After using the function, call **NETClient.Logout** to log out of the device.
Step 5 After using NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

Notes

- Login handle: When the login is successful, the returned value of the interface is not 0 (even the handle is smaller than 0, the login is also successful). One device can log in multiple times with different handle at each login. If there is not special function module, it is suggested to log in only one time. The login handle can be repeatedly used on other function modules.
- Handle repetition: The login handle might be the same as an existing handle, which is normal. For example, if you log in to Device A and get **loginIDA**, then cancel **loginIDA**. When you log in again, you might get **LoginIDA** again. However, throughout the life cycle of a handle, the same handle will not appear.
- Logout: The interface will release the opened functions in the login session internally, but it is not suggested to rely on the cleaning up function of the logout interface. For example, if you opened the monitoring function, you should call the interface that stops the monitoring function when it is no longer required.
- Use login and logout in pairs: The login consumes some memory and socket information and releases sources once logged out.
- Login failure: Call **NETClient.GetLastError** to get failure information.
- After the device is disconnected, the login ID of the device is invalid, and will become valid after the device is reconnected.

2.2.4 Example Code

```
// Log in to the device.
NET_DEVICEINFO_Ex m_DeviceInfo = new NET_DEVICEINFO_Ex();
IntPtr m_LoginID = NETClient.LoginWithHighLevelSecurity(ip, port, name, password,
EM_LOGIN_SPAC_CAP_TYPE.TCP, IntPtr.Zero, ref m_DeviceInfo);
if (IntPtr.Zero == m_LoginID)
{
MessageBox.Show(this, NETClient.GetLastError());
return;
}

// Log out of the device.
if (IntPtr.Zero != m_LoginID)
{
bool result = NETClient.Logout(m_LoginID);
if (!result)
    {
    MessageBox.Show(this, NETClient.GetLastError());
    return;
}
m_LoginID = IntPtr.Zero;
}
```

2.3 Getting External Device List

2.3.1 Introduction

Get the ID of devices that are connected to the unit.

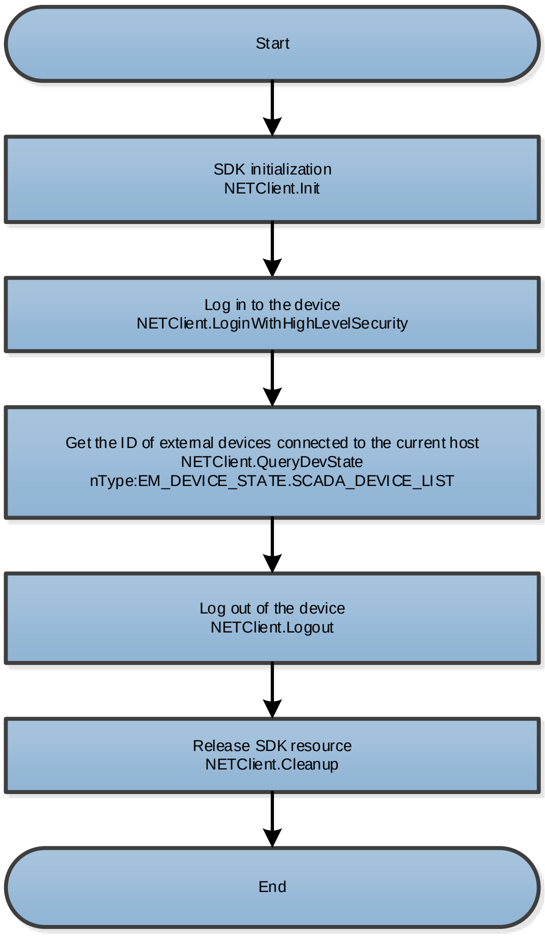
2.3.2 Interface Overview

Table 2-3 Description of the interface for getting external device list

Interface	Description
NETClient.QueryDevState	Get the ID of external devices that are connected to the current host.

2.3.3 Process Description

Figure 2-3 Process of getting external device list



Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.

Step 2 Call **NETClient.LoginWithHighLevelSecurity** to log in to the device.

Step 3 Call **NETClient.QueryDevState** to get the external device list. The parameter nType value is EM_DEVICE_STATE.SCADA_DEVICE_LIST, and the corresponding structure is NET_SCADA_DEVICE_LIST.

Step 4 After using the function, call **NETClient.Logout** to log out of the device.

Step 5 After using NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

2.3.4 Example Code

```
List<NET_SCADA_DEVICE_ID_INFO> deviceInfo_list = new List<NET_SCADA_DEVICE_ID_INFO>();

NET_SCADA_DEVICE_LIST device_list = new NET_SCADA_DEVICE_LIST();
device_list.dwSize = (uint)Marshal.SizeOf(typeof(NET_SCADA_DEVICE_LIST));
device_list.nMax = 64;
device_list.pstuDeviceIDInfo = IntPtr.Zero;
device_list.pstuDeviceIDInfo =
Marshal.AllocHGlobal(Marshal.SizeOf(typeof(NET_SCADA_DEVICE_ID_INFO)) * device_list.nMax);

NET_SCADA_DEVICE_ID_INFO[] array_deviceinfo = new NET_SCADA_DEVICE_ID_INFO[64];
for (int i = 0; i < 64; i++)
{
    array_deviceinfo[i] = new NET_SCADA_DEVICE_ID_INFO();
    Marshal.StructureToPtr(array_deviceinfo[i], IntPtr.Add(device_list.pstuDeviceIDInfo, i *
Marshal.SizeOf(typeof(NET_SCADA_DEVICE_ID_INFO))), true);
}

object objInfo = device_list;
bool ret = NETClient.QueryDevState(m_LoginID, EM_DEVICE_STATE.SCADA_DEVICE_LIST, ref objInfo,
typeof(NET_SCADA_DEVICE_LIST), 10000);
if (!ret)
{
    MessageBox.Show(NETClient.GetLastError());
    return;
}
else
{
    device_list = (NET_SCADA_DEVICE_LIST)objInfo;

    for (int i = 0; i < device_list.nRet; i++)
    {
        var deviceInfo_item =
(NET_SCADA_DEVICE_ID_INFO)Marshal.PtrToStructure(IntPtr.Add(device_list.pstuDeviceIDInfo, i *
Marshal.SizeOf(typeof(NET_SCADA_DEVICE_ID_INFO))), typeof(NET_SCADA_DEVICE_ID_INFO));
        deviceInfo_list.Add(deviceInfo_item);
    }
}
```

2.4 Getting Point Information

2.4.1 Introduction

Get the corresponding monitoring point information according to the ID of each device on the power and environment host.

2.4.2 Interface Overview

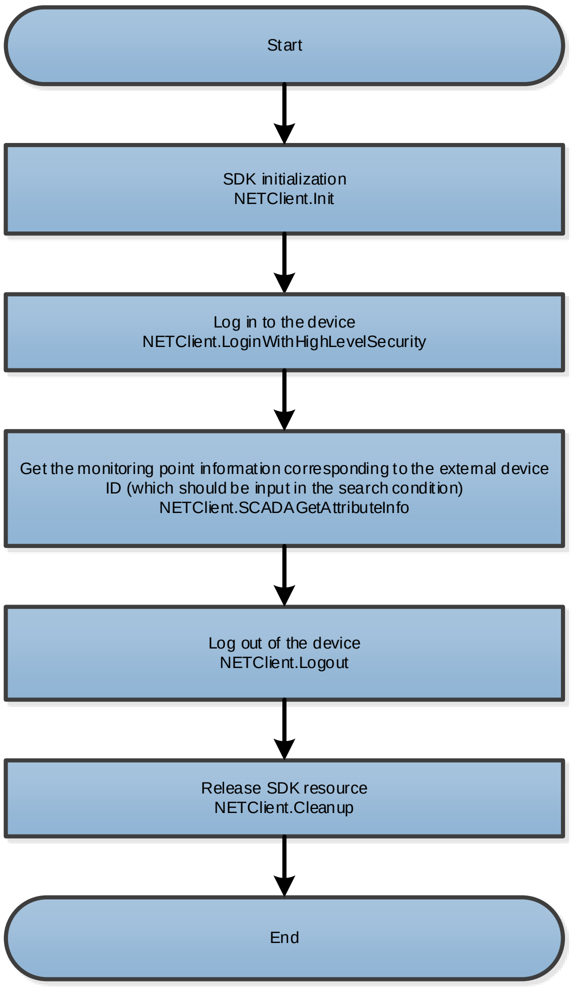
Table 2-4 Description of the interface for getting point information

Interface	Description

Interface	Description
NETClient.SCADAGetAttributeInfo	Get device point information.

2.4.3 Process Description

Figure 2-4 Process of getting point information


Start
SDK initialization
NETClient.Init
Log in to the device
NETClient.LoginWithHighLevelSecurity
Get the monitoring point information corresponding to the external device ID (which should be input in the search condition)
NETClient.SCADAGetAttributeInfo
Log out of the device
NETClient.Logout
Release SDK resource
NETClient.Cleanup
End

Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.

Step 2 After initialization, call **NETClient.LoginWithHighLevelSecurity** to log in to the device.

Step 3 Call **NETClient.SCADAGetAttributeInfo** to get the monitoring point information corresponding to the external device ID, which is obtained by inputting **NETClient.QueryDevState** in **szDeviceID** of the search condition **stuCondition**.

Step 4 After using the function, call **NETClient.Logout** to log out of the device.

Step 5 After using NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

2.4.4 Example Code

```
NET_IN_SCADA_GET_ATTRIBUTE_INFO stuAttributeInfoIn = new
NET_IN_SCADA_GET_ATTRIBUTE_INFO();
stuAttributeInfoIn.dwSize = (uint)Marshal.SizeOf(typeof(NET_IN_SCADA_GET_ATTRIBUTE_INFO));
stuAttributeInfoIn.stuCondition = new NET_GET_CONDITION_INFO();
stuAttributeInfoIn.stuCondition.szDeviceID = "******";

NET_OUT_SCADA_GET_ATTRIBUTE_INFO stuAttributeInfoOut = new
NET_OUT_SCADA_GET_ATTRIBUTE_INFO();
stuAttributeInfoOut.dwSize =
(uint)Marshal.SizeOf(typeof(NET_OUT_SCADA_GET_ATTRIBUTE_INFO));
stuAttributeInfoOut.nMaxAttributeInfoNum = 20;
stuAttributeInfoOut.pstuAttributeInfo = IntPtr.Zero;
stuAttributeInfoOut.pstuAttributeInfo =
Marshal.AllocHGlobal((int)(Marshal.SizeOf(typeof(NET_ATTRIBUTE_INFO)) *
stuAttributeInfoOut.nMaxAttributeInfoNum));

NET_ATTRIBUTE_INFO[] array_attributeInfo = new
NET_ATTRIBUTE_INFO[stuAttributeInfoOut.nMaxAttributeInfoNum];
for (int index = 0; index < stuAttributeInfoOut.nMaxAttributeInfoNum; index++)
{
        IntPtr pIndexBuf = IntPtr.Add(stuAttributeInfoOut.pstuAttributeInfo, index *
Marshal.SizeOf(typeof(NET_ATTRIBUTE_INFO)));
        Marshal.StructureToPtr(array_attributeInfo[index], pIndexBuf, true);
}

IntPtr pstInParam = IntPtr.Zero;
pstInParam = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(NET_IN_SCADA_GET_ATTRIBUTE_INFO)));
Marshal.StructureToPtr(stuAttributeInfoIn, pstInParam, true);

IntPtr pstOutParam = IntPtr.Zero;
pstOutParam =
Marshal.AllocHGlobal(Marshal.SizeOf(typeof(NET_OUT_SCADA_GET_ATTRIBUTE_INFO)));
Marshal.StructureToPtr(stuAttributeInfoOut, pstOutParam, true);

bool ret = NETClient.SCADAGetAttributeInfo(m_LoginID, pstInParam, pstOutParam, 5000);
if (!ret)
{
        MessageBox.Show(NETClient.GetLastError());
        return;
}
else
{
        var attribute_info =
(NET_OUT_SCADA_GET_ATTRIBUTE_INFO)Marshal.PtrToStructure(pstOutParam,
typeof(NET_OUT_SCADA_GET_ATTRIBUTE_INFO));
        if (attribute_info.nRetAttributeInfoNum > 0)
        {
                for (int j = 0; j < attribute_info.nRetAttributeInfoNum; j++)
                {
```

```
                var attributeInfo_item =
(NET_ATTRIBUTE_INFO)Marshal.PtrToStructure(IntPtr.Add(attribute_info.pstuAttributeInfo,
Marshal.SizeOf(typeof(NET_ATTRIBUTE_INFO)) * j), typeof(NET_ATTRIBUTE_INFO));
            }
        }
}

Marshal.FreeHGlobal(pstInParam);
pstInParam = IntPtr.Zero;
Marshal.FreeHGlobal(pstOutParam);
pstOutParam = IntPtr.Zero;
```

2.5 Subscribing to Monitoring Point Alarm

2.5.1 Introduction

Monitor the alarm information of each monitoring point.

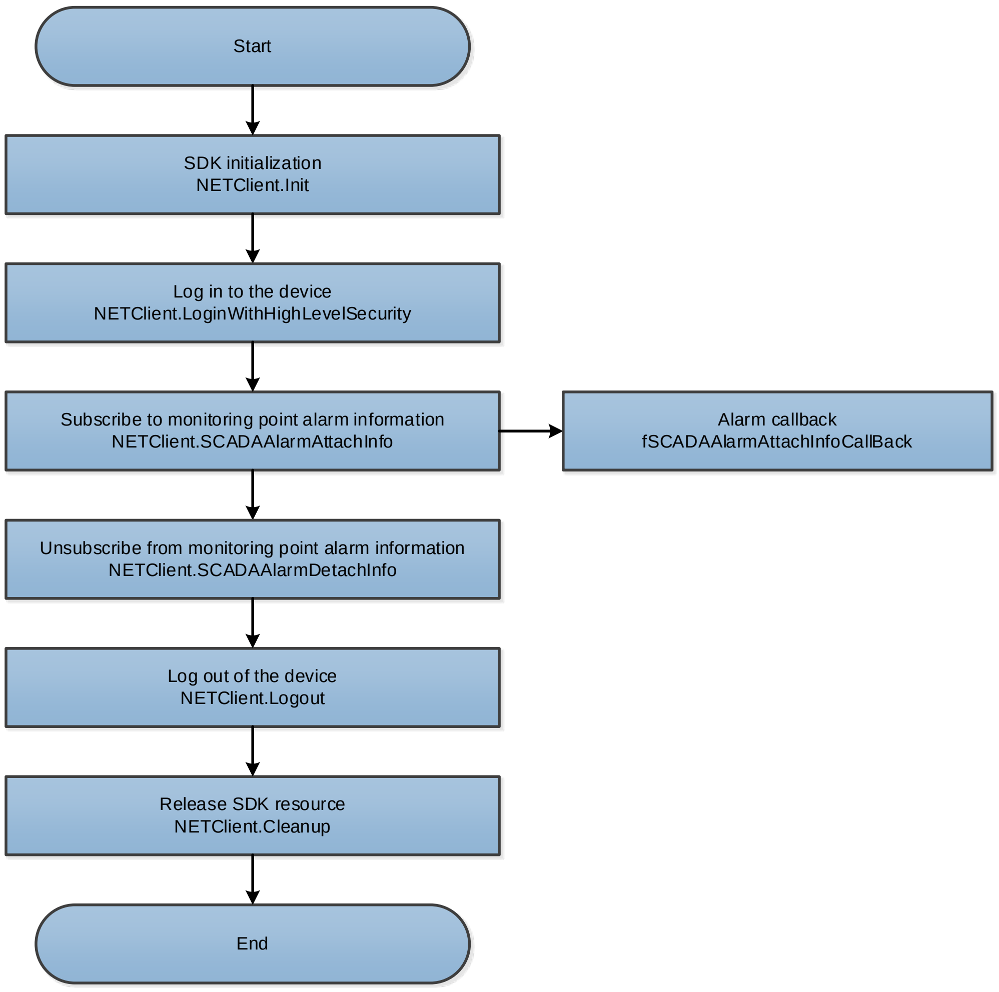
2.5.2 Interface Overview

Table 2-5 Description of the monitoring point alarm interface

Interface	**Description**
NETClient.SCADAAlarmAttachInfo	Subscribe to monitoring point alarm information.
NETClient.SCADAAlarmDetachInfo	Unsubscribe from monitoring point alarm information.

2.5.3 Process Description

Figure 2-5 Process of subscription to monitoring point alarm


Start
SDK initialization
NETClient.Init
Log in to the device
NETClient.LoginWithHighLevelSecurity
Subscribe to monitoring point alarm information
NETClient.SCADAAlarmAttachInfo
Alarm callback
fSCADAAlarmAttachInfoCallBack
Unsubscribe from monitoring point alarm information
NETClient.SCADAAlarmDetachInfo
Log out of the device
NETClient.Logout
Release SDK resource
NETClient.Cleanup
End

Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.

Step 2 After initialization, call **NETClient.LoginWithHighLevelSecurity** to log in to the device.

Step 3 Call **NETClient.SCADAAlarmAttachInfo** to subscribe to alarm from the device. The corresponding input and output structures are NET_IN_SCADA_ALARM_ATTACH_INFO and NET_OUT_SCADA_ALARM_ATTACH_INFO. After subscription, users are notified of the alarm events reported by the device through the callback function **cbCallBack** set in the NET_IN_SCADA_ALARM_ATTACH_INFO structure.

Step 4 After using the alarm reporting function, call **NETClient.SCADAAlarmDetachInfo** to stop subscribing to alarm from the device.

Step 5 After using the function, call **NETClient.Logout** to log out of the device.

Step 6 After using NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

2.5.4 Example Code

```
// Declare static callback delegation.
```

```
private static fSCADAAlarmAttachInfoCallBack m_SCADAAlarmAttachInfoCallBack = new
fSCADAAlarmAttachInfoCallBack(SCADAAlarmAttachInfoCallBack);

// Subscribe to alarms
NET_IN_SCADA_ALARM_ATTACH_INFO inInfo = new NET_IN_SCADA_ALARM_ATTACH_INFO();
inInfo.dwSize = (uint)Marshal.SizeOf(typeof(NET_IN_SCADA_ALARM_ATTACH_INFO));
inInfo.cbCallBack = m_SCADAAlarmAttachInfoCallBack;

NET_OUT_SCADA_ALARM_ATTACH_INFO outInfo = new NET_OUT_SCADA_ALARM_ATTACH_INFO();
outInfo.dwSize = (uint)Marshal.SizeOf(typeof(NET_OUT_SCADA_ALARM_ATTACH_INFO));

m_AttachAlarm = NETClient.SCADAAlarmAttachInfo(m_LoginID, inInfo, outInfo, 3000);
if (IntPtr.Zero == m_AttachAlarm)
{
    MessageBox.Show(this, NETClient.GetLastError());
    return;
}

// Unsubscribe from alarms
bool ret = NETClient.SCADAAlarmDetachInfo(m_AttachAlarm);
if (ret)
{
    MessageBox.Show(this, NETClient.GetLastError());
    return;
}
m_AttachAlarm = IntPtr.Zero;

// Handle alarm callback
private void SCADAAlarmAttachInfoCallBack(IntPtr lAttachHandle, IntPtr pInfo, int nBufLen, IntPtr
dwUser)
{
        NET_SCADA_NOTIFY_POINT_ALARM_INFO_LIST info =
(NET_SCADA_NOTIFY_POINT_ALARM_INFO_LIST)Marshal.PtrToStructure(pInfo,
typeof(NET_SCADA_NOTIFY_POINT_ALARM_INFO_LIST));
    for (int i = 0; i < info.nList; i++)
    {
        Console.WriteLine(info.stuList[i].szDevID);
        Console.WriteLine(info.stuList[i].szPointID);
    }
}
```

2.6 Subscribing to Real-time Information of Monitoring Point

2.6.1 Introduction

Monitor the general information upload of each monitoring point.

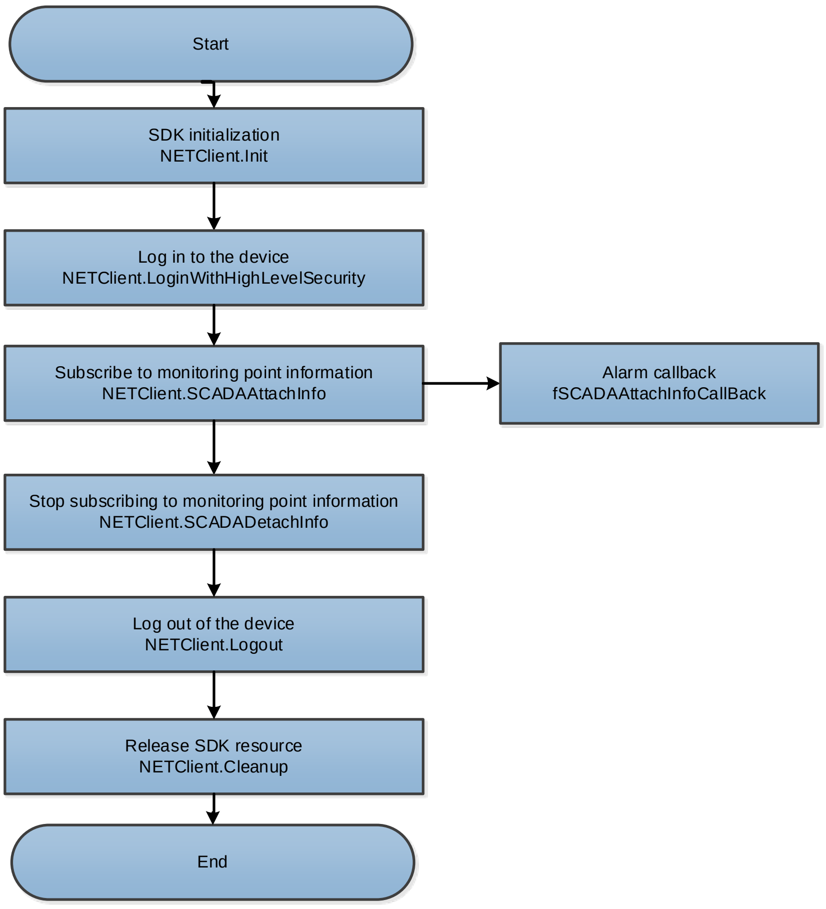
2.6.2 Interface Overview

Table 2-6 Monitoring point information

Interface	Description
NETClient.SCADAAttachInfo	Subscribe to real-time Information of monitoring point.
NETClient.SCADADetachInfo	Unsubscribe from monitoring point information.

2.6.3 Process Description

Figure 2-6 Process of subscription to real-time information of monitoring point


Start
SDK initialization
NETClient.Init
Log in to the device
NETClient.LoginWithHighLevelSecurity
Subscribe to monitoring point information
NETClient.SCADAAttachInfo
Alarm callback
fSCADAAttachInfoCallBack
Stop subscribing to monitoring point information
NETClient.SCADADetachInfo
Log out of the device
NETClient.Logout
Release SDK resource
NETClient.Cleanup
End

Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.
Step 2 After initialization, call **NETClient.LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **NETClient.SCADAAttachInfo** to subscribe to alarm from the device. The corresponding input and output structures are NET_IN_SCADA_ATTACH_INFO and NET_OUT_SCADA_ATTACH_INFO. After successful subscription, alarm events reported by the device are notified to you through the callback cbCallBack set in the NET_IN_SCADA_ATTACH_INFO structure.
Step 4 After using the alarm reporting function, call **NETClient.SCADADetachInfo** to stop subscribing to alarm from the device.
Step 5 After using the function, call **NETClient.Logout** to log out of the device.
Step 6 After using NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

2.6.4 Example Code

```
// Declare static callback delegation.
private static fSCADAAttachInfoCallBack m_SCADAAttachInfoCallBack = new
fSCADAAttachInfoCallBack(SCADAAttachInfoCallBack);

// Subscribe to real-time information
NET_IN_SCADA_ATTACH_INFO inInfo = new NET_IN_SCADA_ATTACH_INFO();
inInfo.dwSize = (uint)Marshal.SizeOf(typeof(NET_IN_SCADA_ATTACH_INFO));
inInfo.cbCallBack = m_SCADAAttachInfoCallBack;
inInfo.emPointType = EM_NET_SCADA_POINT_TYPE.ALL;

NET_OUT_SCADA_ATTACH_INFO outInfo = new NET_OUT_SCADA_ATTACH_INFO();
outInfo.dwSize = (uint)Marshal.SizeOf(typeof(NET_OUT_SCADA_ATTACH_INFO));

m_AttachInfo = NETClient.SCADAAttachInfo(m_LoginID, inInfo, outInfo, 3000);
if (IntPtr.Zero == m_AttachInfo)
{
    MessageBox.Show(this, NETClient.GetLastError());
    return;
}

// Unsubscribe from real-time information
bool ret = NETClient.SCADADetachInfo(m_AttachInfo);
if (ret)
{
    MessageBox.Show(this, NETClient.GetLastError());
    return;
}
m_AttachInfo = IntPtr.Zero;

// Handle alarm callback
```

```
private void SCADAAttachInfoCallBack(IntPtr lLoginID, IntPtr lAttachHandle, IntPtr pInfo, int nBufLen,
IntPtr dwUser)
{
                NET_SCADA_NOTIFY_POINT_INFO_LIST info =
(NET_SCADA_NOTIFY_POINT_INFO_LIST)Marshal.PtrToStructure(pInfo,
typeof(NET_SCADA_NOTIFY_POINT_INFO_LIST));

        for (int i = 0; i < info.nList; i++)
        {
                Console.WriteLine(info.stuList[i].szDevName);
                Console.WriteLine(info.stuList[i].szPointName);
                Console.WriteLine(info.stuList[i].emPointType.ToString());
        }
}
```

2.7 Subscribing to General Alarm

2.7.1 Introduction

Alarm reporting method: Use NetSDK to log in to the device and subscribe to the alarm function from the device. When the device detects an alarm event, it will send the event to NetSDK immediately. You can get the corresponding alarm information through the alarm callback.

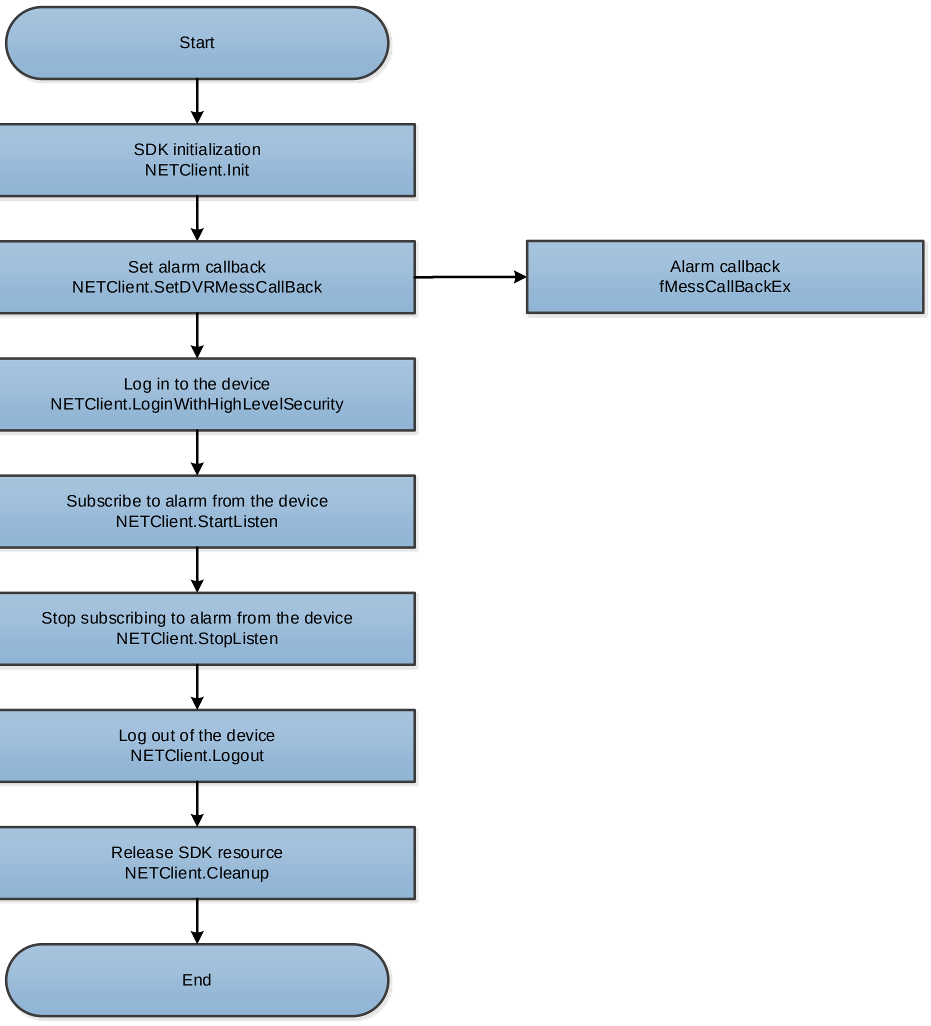
2.7.2 Interface Overview

Table 2-7 Description of alarm reporting interfaces

Interface	**Description**
NETClient.SetDVRMessCallBack	Set alarm callback interface.
NETClient.StartListen	Extension interface for subscribing to alarm.
NETClient.StopListen	Unsubscribe from alarm.

2.7.3 Process Description

Figure 2-7 Process of subscription to general alarm



Process Description

Step 1 Call **NETClient.Init** to initialize NetSDK.

Step 2 Call **NETClient.SetDVRMessCallBack** to set alarm callback. This interface should be called before alarm subscription.

Step 3 Call **NETClient.LoginWithHighLevelSecurity** to log in to the device.

Step 4 Call **NETClient.StartListen** to subscribe to alarm from the device. After successful subscription, alarm events reported by the device are notified to you through the callback set by **NETClient.SetDVRMessCallBack**.

Step 5 After using the alarm reporting function, call **NETClient.StopListen** to stop subscribing to alarm from the device.

Step 6 Call **NETClient.Logout** to log out of the device.

Step 7 After using NetSDK, call **NETClient.Cleanup** to release NetSDK resource.

2.7.4 Example Code

```
// Declare static callback delegation.
```

```
private static fMessCallBackEx m_AlarmCallBack;
m_AlarmCallBack = new fMessCallBackEx(AlarmCallBackEx);
// Configure alarm callback.
NETClient.SetDVRMessCallBack(m_AlarmCallBack, IntPtr.Zero);

// Process alarm callback.
private bool AlarmCallBackEx(int lCommand, IntPtr lLoginID, IntPtr pBuf, uint dwBufLen, IntPtr
pchDVRIP, int nDVRPort, bool bAlarmAckFlag, int nEventID, IntPtr dwUser)
{
        EM_ALARM_TYPE type = (EM_ALARM_TYPE)lCommand;
        switch (type)
        {
                case EM_ALARM_TYPE.ALARM_SCADA_DEV_ALARM:
                        NET_ALARM_SCADA_DEV_INFO info =
(NET_ALARM_SCADA_DEV_INFO)Marshal.PtrToStructure(pBuf,
typeof(NET_ALARM_SCADA_DEV_INFO));

                        Console.WriteLine(info.nChannel.ToString());
                        Console.WriteLine(info.stuTime.ToString());
                        Console.WriteLine(info.szDevName);
                        break;
                default:
                        break;
        }

        return true;
}
// Subscribe to alarm.
bool ret = NETClient.StartListen(m_LoginID);
if (!ret)
{
MessageBox.Show(this, NETClient.GetLastError());
return;
}
// Unsubscribe from alarm.
bool ret = NETClient.StopListen(m_LoginID);
if (!ret)
{
MessageBox.Show(this, NETClient.GetLastError());
return;
}
```

3 Interface Function

3.1 Initializing NetSDK

3.1.1 Init

Table 3-1 NetSDK Initialization Init

Item	Description	
Description	Initialize the NetSDK.	

Function	bool Init(
	    fDisConnectCallBack cbDisConnect,
	    IntPtr dwUser,
	    NETSDK_INIT_PARAM? stuInitParam
	);	

Parameter	[in]cbDisConnect	Disconnection callback.
	[in]dwUser	User parameters for disconnection callback.
	[in]stuInitParam	NetSDK initialization parameter.
Return Value	● Success: true. ● Failure: false.	
Description	● Prerequisite for calling other functions of the network NetSDK. ● When the callback is set as NULL, the device will not be called back to the user after disconnection. ● The dwUser parameter input by Init will be returned in the same field dwUser within the callback cbDisConnect. This helps to position. The same applies to other functions.	

3.1.2 Cleanup

Table 3-2 NetSDK Cleanup

Item	Description
Description	Clean up NetSDK.
Function	void Cleanup().
Parameter	None.
Return Value	None.
Description	NetSDK cleaning up interface is finally called before the end.

3.1.3 SetAutoReconnect

Table 3-3 Set reconnection callback SetAutoReconnect

Item	Description
Description	Set auto reconnection callback.

Item	Description	

Function	void SetAutoReconnect(
	        fHaveReConnectCallBack cbAutoConnect,
	        IntPtr dwUser
	);	

Parameter	[in]cbAutoConnect	Reconnection callback.
	[in]dwUser	User parameters for reconnection callback.
Return Value	None.	
Description	Set reconnection callback interface. If the callback is set as NULL, the device will not be reconnected automatically.	

3.1.4 SetNetworkParam

Table 3-4 Set network parameters SetNetworkParam

Item	Description	
Description	Set related parameters of network environment.	
Function	void SetNetworkParam(NET_PARAM? netParam);	
Parameter	[in]netParam	Network delay, number of reconnections, buffer size and other parameters.
Return Value	None.	
Description	You can adjust parameters according to the actual network environment.	

3.2 Logging in and out

3.2.1 LoginWithHighLevelSecurity

Table 3-5 Log in to device LoginWithHighLevelSecurity

Item	Description	
Description	The user logs in to the device.	

Function	IntPtr LoginWithHighLevelSecurity(
	        string pchDVRIP,
	        ushort wDVRPort,
	        string pchUserName,
	        string pchPassword,
	        EM_LOGIN_SPAC_CAP_TYPE emSpecCap,
	        IntPtr pCapParam,
	        ref NET_DEVICEINFO_Ex deviceInfo
	);	

Parameter	[in]pchDVRIP	Device IP.
	[in]wDVRPort	Device port.
	[in]pchUserName	Username.
	[in]pchPassword	Password.
	[in]emSpecCap	Login category.

Item	Description	
	[in]pCapParam	Login category parameter.
	[out]deviceInfo	Device information.
Return Value	● Success: Non-0. ● Failure: 0.	
Description	None.	

3.2.2 Logout

Table 3-6 Log out ofdDevice Logout

Item	Description	
Description	The user logs out of the device.	

Function	bool Logout(
	    IntPtr lLoginID
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
Return Value	Return false for failure and true for success.	
Description	None.	

3.3 Getting External Device List QueryDevState

Table 3-7 Get list of external devices QueryDevState

Item	Description	
Description	Get the list of external devices that are connected to the current host.	

Function	bool QueryDevState(
	    IntPtr lLoginID,
	    int nType,
	    ref object obj,
	    Type typeName,
	    int waittime
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
	[in]nType	Search for information type. When getting the external device list, nType is EM_DEVICE_STATE.SCADA_DEVICE_LIST.
	[out]obj	Cache for receiving data returned by the search, corresponding to the NET_SCADA_DEVICE_LIST structure.
	[in]typeName	Search for structure type.
	[in]waittime	Waiting time in search status.
Return Value	● Success: true. ● Failure: false.	
Description	None.	

3.4 Getting Point Information SCADAGetAttributeInfo

Table 3-8 Get Device Point Information SCADAGetAttributeInfo

Item	Description	
Description	Get device point information.	

Function	bool SCADAGetAttributeInfo(
	    IntPtr lLoginID,
	    IntPtr pstInParam,
	    IntPtr pstOutParam,
	    int nWaitTime
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
	[in]pstInParam	Input parameter.
	[out]pstOutParam	Output parameter.
	[in]nWaitTime	Waiting time in search status.
Return Value	● Success: Non-0. ● Failure: 0.	
Description	None.	

3.5 Subscribing to Monitoring Point Alarm

3.5.1 SCADAAlarmAttachInfo

Table 3-9 Subscribe to Monitoring Point Alarm Information SCADAAlarmAttachInfo

Item	Description	
Description	Subscribe to monitoring point alarm information.	

Function	IntPtr SCADAAlarmAttachInfo(
	    IntPtr lLoginID,
	    NET_IN_SCADA_ALARM_ATTACH_INFO pInParam,
	    NET_OUT_SCADA_ALARM_ATTACH_INFO pOutParam,
	    int nWaitTime = 3000
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
	[in]pInParam	Subscription input parameter.
	[out]pOutParam	Subscription output parameter.
	[in]waittime	Waiting time.
Return Value	● Success: Non-0. ● Failure: 0.	
Description	None	

3.5.2 SCADAAlarmDetachInfo

Table 3-10 Unsubscribe from Point Alarm Information SCADAAlarmDetachInfo

Item	Description	
Description	Unsubscribe from point alarm information.	
Function	bool SCADAAlarmDetachInfo(IntPtr lAttachHandle)	
Parameter	[in]lAttachHandle	Return value of NETClient.SCADAAlarmAttachInfo
Return Value	Return false for failure and true for success.	
Description	None.	

3.6 Subscribing to Real-time Information of Monitoring Point

3.6.1 SCADAAttachInfo

Table 3-11 Subscribe to Real-time Information of Monitoring Point SCADAAttachInfo

Item	Description	
Description	Subscribe to real-time information of monitoring point.	

Function	IntPtr SCADAAttachInfo(
	    IntPtr lLoginID,
	    NET_IN_SCADA_ATTACH_INFO pInParam,
	    NET_OUT_SCADA_ATTACH_INFO pOutParam,
	    int nWaitTime
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
	[in]pInParam	Subscription input parameter.
	[out]pOutParam	Subscription output parameter.
	[in]waittime	Waiting time.
Return Value	● Success: Non-0. ● Failure: 0.	
Description	None.	

3.6.2 SCADADetachInfo

Table 3-12 Unsubscribe from Point Information SCADADetachInfo

Item	Description	
Description	Unsubscribe from point information.	

Function	bool SCADADetachInfo(
	    IntPtr lAttachHandle
	);	

Parameter	[in]lAttachHandle	Return value of NETClient.SCADAAttachInfo.
Return Value	● Success: true. ● Failure: false.	

Item	Description
Description	None.

3.7 Reporting Alarm

3.7.1 CallbackSetDVRMessCallBack

Table 3-13 Set alarm callback SetDVRMessCallBack

Item	Description	
Description	Set alarm callback.	

Function	void SetDVRMessCallBack(
	        fMessCallBackEx cbMessage,
	        IntPtr dwUser
	);	

Parameter	[in]cbMessage	● Message callback, which can call back the device status, such as alarm status. ● When the value is set as 0, it means callback is forbidden.
	[in]dwUser	User-defined data.
Return Value	None.	
Description	● Set device message callback to get the current device status information; this function is independent of the calling sequence, and the NetSDK is not called back by default. ● The callback fMessCallBack must call the alarm message subscription interface StartListen first before it takes effect.	

3.7.2 StartListen

Table 3-14 Subscribe to alarm StartListen

Item	Description	
Description	Subscribe to alarm.	

Function	bool StartListen(
	        IntPtr lLoginID
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
Return Value	● Success: true. ● Failure: false.	
Description	Subscribe to device message, and the message received is called back from the set value of SetDVRMessCallBack.	

3.7.3 StopListen

Table 3-15 Unsubscribe from alarm StopListen

Item	Description	
Description	Stop subscribing to alarm.	

Function	bool StopListen(
	    IntPtr lLoginID
	);	

Parameter	[in]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity.
Return Value	● Success: true. ● Failure: false.	
Description	None.	

4 Callback

4.1 fDisConnectCallBack

Table 4-1 Disconnection callback fDisConnectCallBack

Item	Description	
Description	Disconnection callback	

Function	public delegate void fDisConnectCallBack(
	    IntPtr lLoginID,
	    IntPtr pchDVRIP,
	    int nDVRPort,
	    IntPtr dwUser
	);	

Parameter	[out]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity
	[out]pchDVRIP	Disconnected device IP
	[out]nDVRPort	Disconnected device port
	[out]dwUser	User parameters for callback
Return Value	None	
Description	None	

4.2 fHaveReConnectCallBack

Table 4-2 Reconnection callback fHaveReConnectCallBack

Item	Description	
Description	Reconnection callback	

Function	public delegate void fHaveReConnectCallBack(
	    IntPtr lLoginID,
	    IntPtr pchDVRIP,
	    int nDVRPort,
	    IntPtr dwUser
	);	

Parameter	[out]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity
	[out]pchDVRIP	Reconnected device IP
	[out]nDVRPort	Reconnected device port
	[out]dwUser	User parameters for callback
Return Value	None	
Description	None	

4.3 fSCADAAlarmAttachInfoCallBack

Table 4-3 Monitoring point alarm information callback fSCADAAlarmAttachInfoCallBack

Item	Description	
Description	Monitoring point alarm information callback	

Function	public delegate void fSCADAAlarmAttachInfoCallBack(
	        IntPtr lAttachHandle,
	        IntPtr pInfo,
	        int nBufLen,
	        IntPtr dwUser
	);	

Parameter	[out]lAttachHandle	Return value of NETClient. SCADAAlarmAttachInfo
	[out]pInfo	Alarm message data block address
	[out]nBufLen	Length of alarm message data block, in bytes
	[out]dwUser	User parameters for callback
Return Value	None	
Description	None	

4.4 fSCADAAttachInfoCallBack

Table 4-4 Monitoring point information callback fSCADAAttachInfoCallBack

Item	Description	
Description	Monitoring point information callback	

Function	public delegate void fSCADAAttachInfoCallBack(
	        IntPtr lLoginID,
	        IntPtr lAttachHandle,
	        IntPtr pInfo,
	        int nBufLen,
	        IntPtr dwUser
	);	

Parameter	[out]lLoginID	Return value of NETClient.LoginWithHighLevelSecurity
	[out]lAttachHandle	Return value of NETClient. SCADAAttachInfo
	[out]pInfo	Reported information data block address
	[out]nBufLen	Length of reported information data block, in bytes
	[out]dwUser	User parameters for callback
Return Value	None	
Description	None	

4.5 fMessCallBackEx

Table 4-5 Alarm callback fMessCallBackEx

Item	Description
Description	Alarm callback

Item	Description	

Function	public delegate bool fMessCallBackEx(
	        int lCommand,
	        IntPtr lLoginID,
	        IntPtr pBuf,
	        uint dwBufLen,
	        IntPtr pchDVRIP,
	        int nDVRPort,
	        bool bAlarmAckFlag,
	        int nEventID,
	        IntPtr dwUser
	);	

Parameter	[out]lCommand	Alarm type. For details, see Table 4-6
	[out]lLoginID	Return value of login interface
	[out]pBuf	Buffer that receives alarm data, which is filled with different data according to different listening interfaces called and lCommand values
	[out]dwBufLen	Length of pBuf, in bytes
	[out]pchDVRIP	Device ip
	[out]nDVRPort	Port
	[out]bAlarmAckFlag	● TRUE: This event can be confirmed. ● FALSE: This event cannot be confirmed.
	[out]nEventID	Return event ID
	[out]dwUser	User-defined data
Return Value	● TRUE: Callback is executed correctly ● FALSE: Execution error	
Description	Usually, call the set callback during application initialization, and process properly in the callback according to different device ID and command values	

Table 4-6 Alarm type description

Alarm Type	Name	pBuf
ALARM_SCADA_DEV_ALARM	Detection and acquisition device alarm event	NET_ALARM_SCADA_DEV_INFO structure

Appendix 1 Cybersecurity Recommendations

Cybersecurity is more than just a buzzword: it's something that pertains to every device that is connected to the internet. IP video surveillance is not immune to cyber risks, but taking basic steps toward protecting and strengthening networks and networked appliances will make them less susceptible to attacks. Below are some tips and recommendations on how to create a more secured security system.

Mandatory actions to be taken for basic device network security:

1. **Use Strong Passwords**

   Please refer to the following suggestions to set passwords:

   - The length should not be less than 8 characters;
   - Include at least two types of characters; character types include upper and lower case letters, numbers and symbols;
   - Do not contain the account name or the account name in reverse order;
   - Do not use continuous characters, such as 123, abc, etc.;
   - Do not use overlapped characters, such as 111, aaa, etc.;

2. **Update Firmware and Client Software in Time**

   - According to the standard procedure in Tech-industry, we recommend to keep your device (such as NVR, DVR, IP camera, etc.) firmware up-to-date to ensure the system is equipped with the latest security patches and fixes. When the device is connected to the public network, it is recommended to enable the "auto-check for updates" function to obtain timely information of firmware updates released by the manufacturer.
   - We suggest that you download and use the latest version of client software.

"Nice to have" recommendations to improve your device network security:

1. **Physical Protection**

   We suggest that you perform physical protection to device, especially storage devices. For example, place the device in a special computer room and cabinet, and implement well-done access control permission and key management to prevent unauthorized personnel from carrying out physical contacts such as damaging hardware, unauthorized connection of removable device (such as USB flash disk, serial port), etc.

2. **Change Passwords Regularly**

   We suggest that you change passwords regularly to reduce the risk of being guessed or cracked.

3. **Set and Update Passwords Reset Information Timely**

   The device supports password reset function. Please set up related information for password reset in time, including the end user's mailbox and password protection questions. If the information changes, please modify it in time. When setting password protection questions, it is suggested not to use those that can be easily guessed.

4. **Enable Account Lock**

   The account lock feature is enabled by default, and we recommend you to keep it on to guarantee the account security. If an attacker attempts to log in with the wrong password several times, the corresponding account and the source IP address will be locked.

5. **Change Default HTTP and Other Service Ports**

   We suggest you to change default HTTP and other service ports into any set of numbers between 1024~65535, reducing the risk of outsiders being able to guess which ports you are using.

6. **Enable HTTPS**

   We suggest you to enable HTTPS, so that you visit Web service through a secure communication channel.

7. **MAC Address Binding**

   We recommend you to bind the IP and MAC address of the gateway to the device, thus reducing the risk of ARP spoofing.

8. **Assign Accounts and Privileges Reasonably**

   According to business and management requirements, reasonably add users and assign a minimum set of permissions to them.

9. **Disable Unnecessary Services and Choose Secure Modes**

   If not needed, it is recommended to turn off some services such as SNMP, SMTP, UPnP, etc., to reduce risks.

   If necessary, it is highly recommended that you use safe modes, including but not limited to the following services:

   - SNMP: Choose SNMP v3, and set up strong encryption passwords and authentication passwords.
   - SMTP: Choose TLS to access mailbox server.
   - FTP: Choose SFTP, and set up strong passwords.
   - AP hotspot: Choose WPA2-PSK encryption mode, and set up strong passwords.

10. **Audio and Video Encrypted Transmission**

    If your audio and video data contents are very important or sensitive, we recommend that you use encrypted transmission function, to reduce the risk of audio and video data being stolen during transmission.

    Reminder: encrypted transmission will cause some loss in transmission efficiency.

11. **Secure Auditing**

    - Check online users: we suggest that you check online users regularly to see if the device is logged in without authorization.
    - Check device log: By viewing the logs, you can know the IP addresses that were used to log in to your devices and their key operations.

12. **Network Log**

    Due to the limited storage capacity of the device, the stored log is limited. If you need to save the log for a long time, it is recommended that you enable the network log function to ensure that the critical logs are synchronized to the network log server for tracing.

13. **Construct a Safe Network Environment**

    In order to better ensure the safety of device and reduce potential cyber risks, we recommend:

    - Disable the port mapping function of the router to avoid direct access to the intranet devices from external network.
    - The network should be partitioned and isolated according to the actual network needs. If there are no communication requirements between two sub networks, it is suggested to use VLAN, network GAP and other technologies to partition the network, so as to achieve the network isolation effect.
    - Establish the 802.1x access authentication system to reduce the risk of unauthorized access to private networks.
    - Enable IP/MAC address filtering function to limit the range of hosts allowed to access the device.