

NetSDK_C# Auto Register

User's Manual



Foreword

General






This manual describes the basic process of Demo demonstration and development of NetSDK C# auto register.

Target Readers

Software engineer, product manager, project manager, and more that might use SDKs.

Safety Instructions

The following symbols may appear in this document and their meanings are as follows:

| 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. | October 2020 |

Privacy Protection Notice

As the device user or data controller, you might collect personal data of others such as face, fingerprints, car plate number, email address, phone number, GPS and so on. You need to be in compliance with the local privacy protection laws and regulations to protect the legitimate rights and interests of other people by implementing measures include but not limited to: providing clear and visible identification to inform data subject the existence of surveillance area and providing related contact.

About the Manual

- The manual is for reference only. If there is inconsistency between the manual and the actual product, the actual product shall prevail.
- We are not liable for any loss caused by the operations that do not comply with the manual.
- The manual would be updated according to the latest laws and regulations of related jurisdictions. For detailed information, refer to the paper manual, CD-ROM, QR code or our official website. If there is inconsistency between paper manual and the electronic version, the electronic version shall prevail.
- All the designs and software are subject to change without prior written notice. The product updates might cause some differences between the actual product and the manual. Please contact the customer service for the latest program and supplementary documentation.
- There still might be deviation in technical data, functions and operations description, or errors in print. If there is any doubt or dispute, we reserve the right of final explanation.
- Upgrade the reader software or try other mainstream reader software if the manual (in PDF format) cannot be opened.
- All trademarks, registered trademarks and the company names in the manual are the properties of their respective owners.
- Please visit our website, contact the supplier or customer service if there is any problem occurring when using the device.
- If there is any uncertainty or controversy, we reserve the right of final explanation.

Glossary

This chapter provides the definitions to some of the terms that appear in the manual to help you understand the function of each module.

| Term | Definition |
|---------------|--|
| Main Stream | A type of video stream that usually has better resolution and clarity and provides a better experience if the network resource is not restricted. |
| Sub Stream | A type of video stream that usually has lower resolution and clarity than the main stream but demands less network resources. The user can choose the stream type according to the particular scenes. |
| Resolution | Resolution is consisted of display resolution and image resolution. Display resolution refers to the quantity of pixels in unit area, and the image resolution refers to information quantity (the quantity of pixels per inch) stored in the image. |
| Video Channel | An abstract concept of the communication and video stream transmission between NetSDK and devices. For example, if a number of cameras (SD, IPC) are mounted on a storage device (NVR), the storage device manages the cameras as video channels which are numbered from 0. If NetSDK connects to the camera directly, the video channel is usually numbered as 0. |

Table of Contents

| | |
|--|------------|
| Foreword | I |
| Glossary | III |
| 1 Introduction | 5 |
| 1.1 General | 5 |
| 1.2 Basic Process | 5 |
| 2 NetSDK C# Auto Register Demo | 6 |
| 2.1 Getting and Changing Demo Config | 6 |
| 2.2 Demo Listening and Device Operations | 7 |
| 3 Interface Calling for NetSDK C# Auto Register | 10 |
| 3.1 Auto Register Config of Device | 10 |
| 3.1.1 Introduction..... | 10 |
| 3.1.2 Interface Overview | 10 |
| 3.1.3 Process Description | 10 |
| 3.1.4 Sample Code | 11 |
| 3.2 Starting and Stopping NetSDK Listening Service | 12 |
| 3.2.1 Introduction..... | 12 |
| 3.2.2 Interface Overview | 13 |
| 3.2.3 Process Description | 13 |
| 3.2.4 Sample Code | 14 |
| Appendix 1 Cybersecurity Recommendations | 17 |

1 Introduction

1.1 General

NetSDK auto register is used to solve network restrictions, for example, the public network server cannot search for devices in the Intranet, which shall actively connect to the public network server. It also helps the monitoring server efficiently configure the device to quickly implement or restore the application scenarios.

NetSDK mainly has three auto register functions:

- Get and set the auto register config of the target device.
- Start and stop the listening server.
- The listening server quickly adds or deletes devices that are listening this server.

In addition, C# Demo can import and export device information, and implement basic functions such as pulling stream and voice talk for a single device.

1.2 Basic Process

Figure 1-1 Basic process of auto register

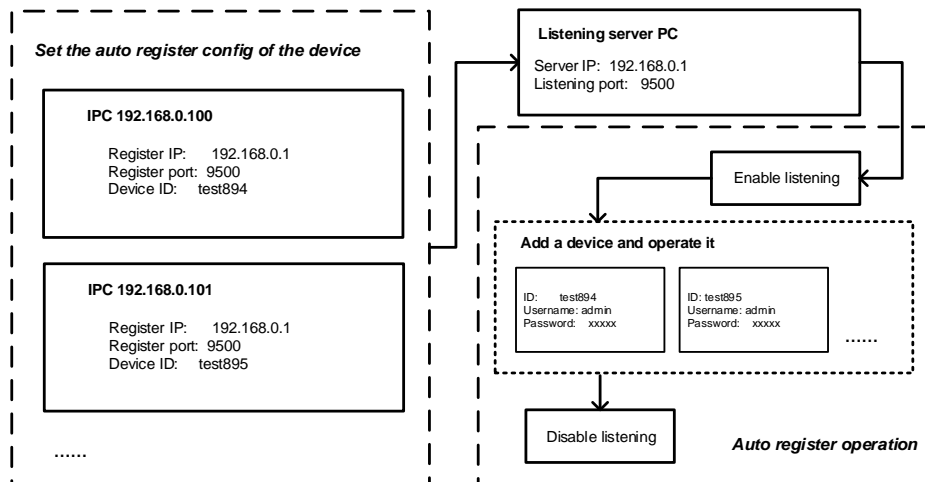


Table 1-1 Main Interface

| Interface | Description |
|--|---|
| Get and set the auto register config of the device | The device has two auto register config methods: Web config and SDK config. You can use SDK to get and set the auto register config of devices with fixed IP. For details, see Chapter Two. |
| Start and stop server listening. | SDK provides some basic functions for auto register listening, including starting and stopping listening. |

2 NetSDK C# Auto Register Demo

2.1 Getting and Changing Demo Config

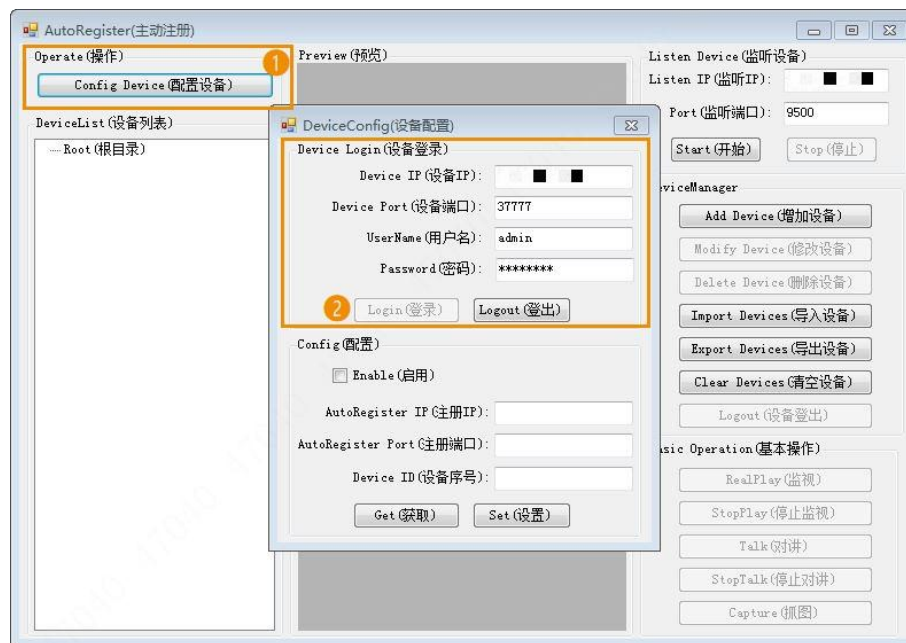
Prerequisite

This chapter is about config after server applied for login from device. Please make sure that the server has access to the device (if the network is separated, it is recommended that independent config tool be developed or select web config tool. Web config tool will not be introduced here, and please refer to device user's manual.)

Procedure

Step 1 Log in to target device. Click **Config Device** in the upper left corner to open the **DeviceConfig** window, and then enter the login information to log in to the device to be configured.

Figure 2-1 Device login



Step 2 Click **Get** to get the original auto register config of the device.



It is recommended to do the change and release of new config after getting the previous config so as to prevent the possible required-field missing.

Step 3 (Optional) If the obtained config is not the platform that the device needs to register to, please change the config to the target platform.



Device ID shall be unique. During C# Demo server listening, all the obtained device information is saved in a list, and the information of a device is extracted by using this ID as Key for login and other follow-up operations.

Step 4 Select the **Enable** check box, and then click **Set** to release the new config to the device.



When releasing config, you must select the **Enable** check box; otherwise only the auto register information will be updated but the config will not take effect.

Figure 2-2 Getting and changing demo config



2.2 Demo Listening and Device Operations

Prerequisite

Please make sure the IP and port of front-end devices match with the IP and port of the listen server.

Procedure

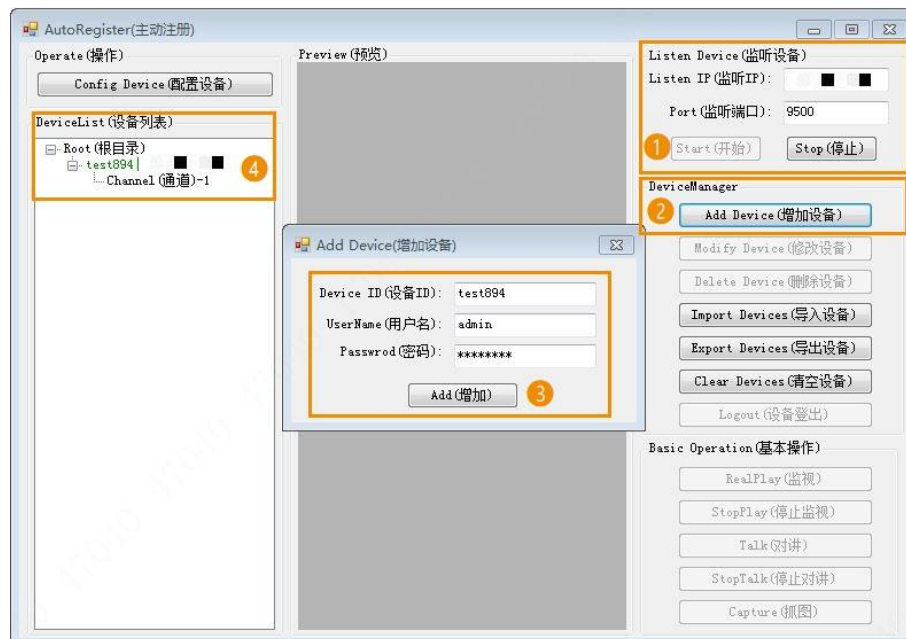
Step 1 Enable listening. Enter listen server's IP and port, and then click **Start**. The SDK callback will send information such as ID, IP and port of the devices that are registered to the server.



Because C# Demo is generally running on the PC, the listening IP is the IP address of the PC where Demo is running, and the port shall not conflict with other programs.

- Step 2** Click **Add Device**, and then enter ID, user name, and password in the pop-up window. Then click Add to log in to device. The device successfully logged in shows green listed on the left. The failed ones show grey.

Figure 2-3 Enable listening and add devices



- Step 3** C# Demo provides additional functions to help fast operation and demonstration.

- Quickly change config and delete device. Select a device on the left list, and then click **Modify Device** and **Delete Device**.
- Batch process the devices. For example, one click to export the auto register information, including device ID, user name, and password. You can select device on the list and then click **Export Device** to export devices in batches.
- Do other operations such as picture capture. Select device on the list, and then do operations as prompted in the **Basic Operation** area, such as live view, capture, and calling.

Figure 2-4 Additional functions provided by Demo (1)

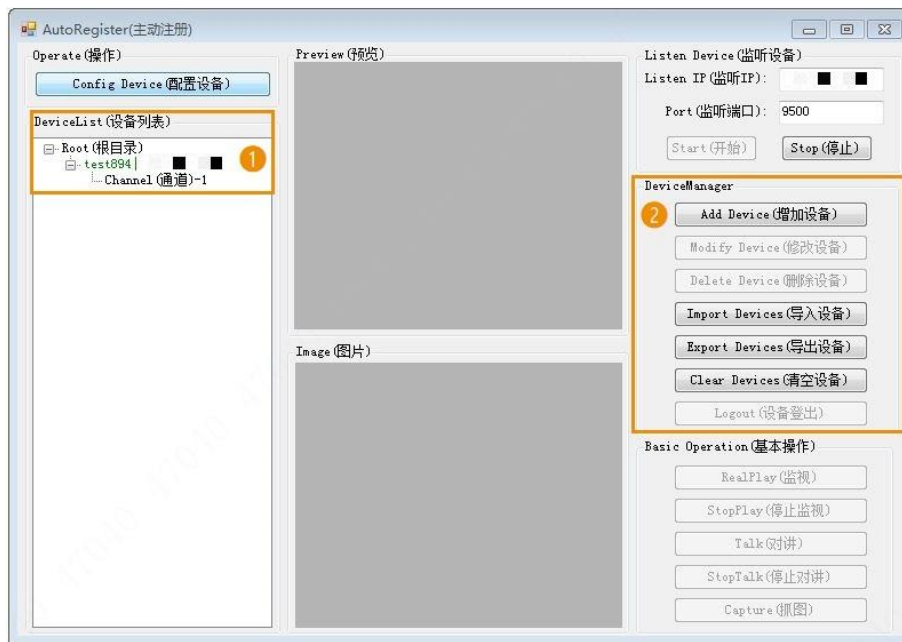


Figure 2-5 Additional functions provided by Demo (2)



Notes

Do not end listening before completing all device operations; otherwise the server cannot send commands to device any more.

3 Interface Calling for NetSDK C# Auto Register

3.1 Auto Register Config of Device

3.1.1 Introduction

The user can call SDK to config the auto register information of device, including enabling auto register, matching device IP and platform IP.

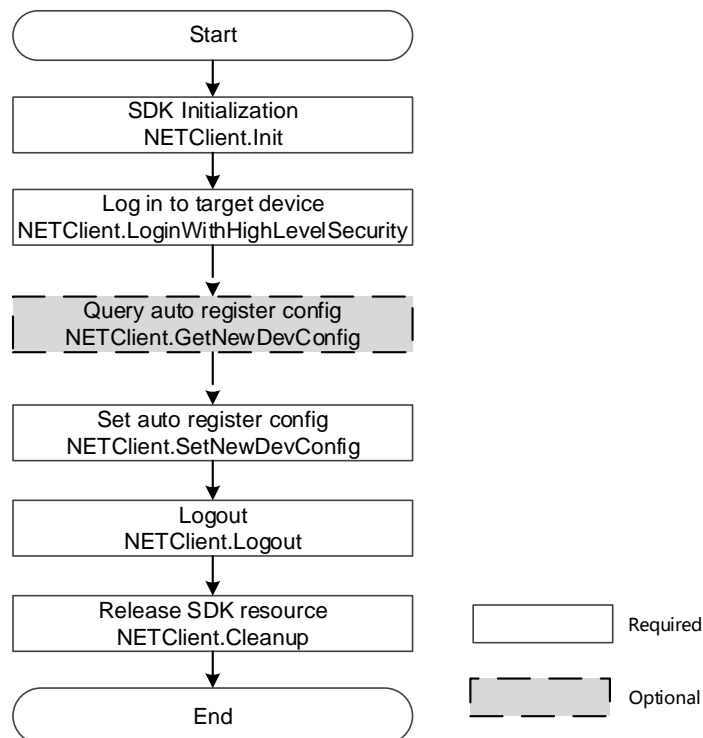
3.1.2 Interface Overview

Table 3-1 Interface overview

| Interface | Description |
|--------------------------------------|---------------------------------------|
| NETClient.Init | Initialize SDK |
| NETClient.LoginWithHighLevelSecurity | Log in to the device of high security |
| NETClient.GetNewDevConfig | Get auto config and parse data |
| NETClient.SetNewDevConfig | Package data and issue config |
| NETClient.Logout | Log out of the device |
| NETClient.Cleanup | Clean up the SDK resource |

3.1.3 Process Description

Figure 3-1 Basic process of getting and configuring auto register config of the device



Process Description

- Step 1 SDK calls **NETClient.Init** interface for initialization.
- Step 2 SDK calls **NETClient.LoginWithHighLevelSecurity** interface to log in to the target device.
- Step 3 (Optional) SDK calls **NETClient.GetNewDevConfig** interface to get and parse the config at one time.
- Step 4 SDK calls **NETClient.SetNewDevConfig** interface to package and send the config at one time.
- Step 5 SDK calls **NETClient.Logout** interface to log out of the device.
- Step 6 SDK calls **NETClient.Cleanup** interface to clean up the SDK resource.

Notes

- You need to call **NETClient.Init** and **NETClient.Cleanup** interfaces as a pair. Single-thread multiple calling in pairs is supported, but it is recommended to call the pair for only once globally. Do not initialize/clean up the resource repeatedly.
- Though you can directly write the parameters to issue config, it is recommended to modify the existing config and then issue it, so as to avoid missing key fields.
- Make sure that your device supports auto register function.
- Before getting and configuring SDK, complete the basic SDK operations first, including initialization and log in to target devices.

3.1.4 Sample Code

```
// For general codes, such as initialization, login, logout, cleanup, please refer to
《NetSDK_C# Programming Manual》
// —>Get auto register config
// m_LoginID is the login field obtained beforehand
CFG_DVRIP_INFO info = new CFG_DVRIP_INFO(); // Structured
parameter configured by auto register
String szCommand = CFG_CMD_DVRIP; // Configuration
enumeration of auto register
Int32 IChannel = -1 // Config channel, it is
set to -1
// C# Demo provides issue method of general configuration
object obj = info;
Type configType = typeof(CFG_DVRIP_INFO);
bool ret =
NETClient.GetNewDevConfig(m_LoginID, -1, szCommand, ref obj, configType ,
m_Waite);
```

```

    if (!ret)
    {
        System.Console.WriteLine(NETClient.GetLastError());
        return;
    }

    // For application of GetNewDevConfig, refer to NetSDK.cs in C# Demo
    NetSDKCS library.

    // obj is a unified structure of in and out parameter
    info = (CFG_DVRIP_INFO)obj;    // In this way you can get the config

    // —>Set config of auto register
    // We recommend to modify config based on the obtained config
    info.nRegistersNum = 1;
    info.stuRegisters[0].bEnable = true;
    info.stuRegisters[0].szDeviceID = "newDeviceID";           // Device ID
    info.stuRegisters[0].nServersNum = 1;
    info.stuRegisters[0].stuServers[0].szAddress = "192.168.0.1"; // Target
server IP
    info.stuRegisters[0].stuServers[0].nPort = 9500;           // Target server
port
    obj = info;
    ret = NETClient.SetNewDevConfig(m_LoginID, -1, szCommand, obj, configType,
m_Waite);
    if (!ret)
    {
        System.Console.WriteLine(NETClient.GetLastError());
        return;
    }

    // For application of SetNewDevConfig, refer to NetSDK.cs in C# Demo
    NetSDKCS library.

```

3.2 Starting and Stopping NetSDK Listening Service

3.2.1 Introduction

SDK provides basic interfaces that are used by listening server to listen to devices registered to server, including starting and stopping listening. SDK also

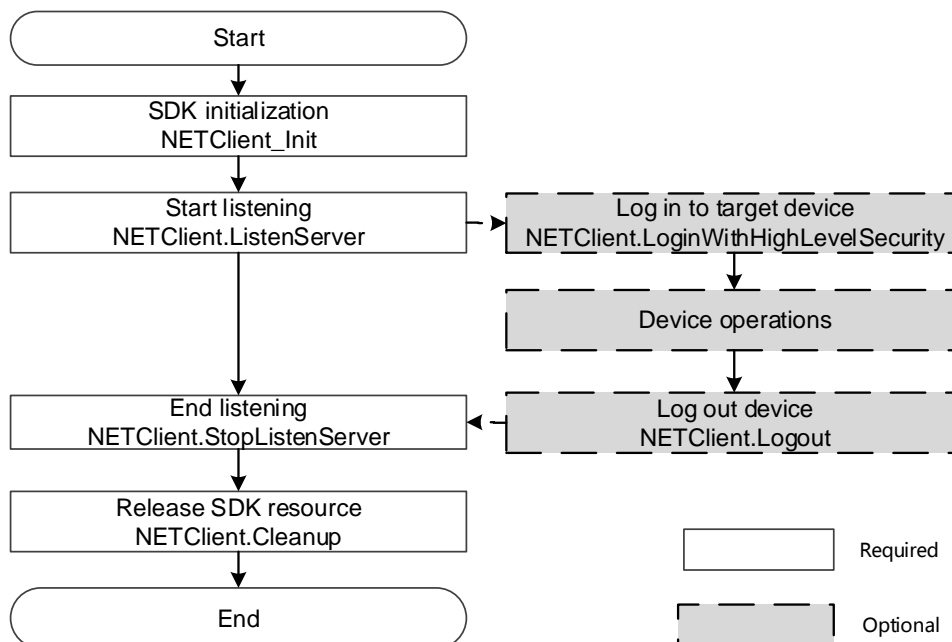
provides callback for the auto register service. The callback is used to listen to events and get the information and status of registered devices.

3.2.2 Interface Overview

| Interface | Description |
|----------------------------|-----------------------|
| NETClient.Init | Initialize SDK |
| NETClient.ListenServer | Start listening |
| NETClient.StopListenServer | Stop listening |
| NETClient.Cleanup | Clean up SDK resource |

3.2.3 Process Description

Figure 3-2 Starting and stopping listening



Process Description

- Step 1 Call the initialization interface **NETClient.Init**.
- Step 2 Call **NETClient.ListenServer** to start listening. **NETClient.ListenServer** interface needs to set **NETClient.fServiceCallBack** through which you can get the ID, IP, port, and other information of devices registered on this server and do other operations.



SDK provides multiple callback events, including device reconnection that helps to monitor the device status. For details, see **NetSDKCS.EM_LISTEN_TYPE**.

Step 3 Call **NETClient.StopListenServer** to end listening.

Step 4 Call **NETClient.Cleanup** to clean up the SDK resource.

Notes

- You need to call **NETClient.Init** and **NETClient.Cleanup** interface as a pair. Single-thread multiple calling in pairs is supported, but it is recommended to call the pair for only once globally. Do not initialize/clean up the resource repeatedly.
- Please make sure the target device IP and port match with server IP and port.
- Because the server can initiate communication to auto registered devices only in listening state, all operations that require the server to send commands to the device must be completed in listening state without stopping listening midway.
- It is recommended not to call other SDK interfaces in callbacks, and if needed, you need to start a new thread.

3.2.4 Sample Code

```
// For general codes, such as initialization and cleanup, please refer to
《NetSDK_C# Programming Manual》

//////////////////// Config listening callback
////////////////////

//////////////////// —>C# provides function entrustment

public delegate int fServiceCallBack( IntPtr IHandle,           // Listening handle
                                     IntPtr plp,               // Device IP pointer
                                     ushort wPort,             // Device port
                                     int ICommand,              // Callback event type
                                     IntPtr pParam,             // Device ID pointer
                                     uint dwParamLen,           // ID pointer data
                                     length
                                     IntPtr dwUserData);        // User data pointer

//////////////////// —>C# custom callback

private int ServiceCallBack(IntPtr IHandle, IntPtr plp, ushort wPort, int
ICommand,
                               IntPtr pParam, uint dwParamLen, IntPtr
dwUserData)
{
    EM_LISTEN_TYPE type = (EM_LISTEN_TYPE)ICommand;
    string ip = Marshal.PtrToStringAnsi(plp);
    string id = "";
```

```

        if (dwParamLen > 0)
        {
            id = Marshal.PtrToStringAnsi(pParam);
        }

        this.BeginInvoke(new Action<string, ushort, EM_LISTEN_TYPE,
string>(UpdateDevice), ip, wPort, type, id);
        return 0;
    }

    // EM_LISTEN_TYPE provides several event types that occur during listening by
    auto register
    private void UpdateDevice(string ip, ushort port, EM_LISTEN_TYPE type, string
    id)
    {
        switch (type)
        {
            case EM_LISTEN_TYPE.NET_DVR_DISCONNECT:
                .....    // Verify device disconnection and callback
            case EM_LISTEN_TYPE.NET_DEV_AUTOREGISTER_RETURN:
                .....    // Carry serial number and token during device registration
            case EM_LISTEN_TYPE.NET_DEV_NOTIFY_IP_RETURN:
                .....    // Device send IP, but it is not used for auto register
            case EM_LISTEN_TYPE.NET_DVR_SERIAL_RETURN
            {
                // Carry serial number during device registration
                TreeNode[] nodes = treeView_devicelist.Nodes[0].Nodes.Find(id, false);
                if (nodes.Count() > 0)
                {
                    DEVICE_INFO info = (DEVICE_INFO)nodes[0].Tag;
                    info.IP = ip;
                    info.Port = port;
                    lock (queueLock)
                    {
                        m_DeviceQueue.Enqueue(info);
                    }
                }
            }
            break;
        }
    }
}

```


// The C# Demo in above example stores the information of registered device in m_DeviceQueue.

```
//////////////////////////////////// Start/end listening
////////////////////////////////////

//////////////////////////////////// —>Start listening and register callback of listening service
private fServiceCallBack m_ServiceCallBack;
m_ServiceCallBack = new fServiceCallBack(ServiceCallBack);          //
Register callback
m_ListenID = NETClient.ListenServer(
                                ListenIP,          // Server listening IP
                                port,              // Server listening port
                                1000,              // Timeout period
                                m_ServiceCallBack, // Service callback
                                IntPtr.Zero);      // User data
if (IntPtr.Zero == m_ListenID)          // Get the listening handle and match it
with the one in callback
{
    System.Console.WriteLine (NETClient.GetLastError());
    return;
}
// For application of ListenServer, refer to NetSDK.cs in C# Demo NetSDKCS
library

//////////////////////////////////// —>End listening

bool ret = NETClient.StopListenServer(m_ListenID);    // Use the listening
handle that is obtained when you start listening
if (!ret)
{
    MessageBox.Show(NETClient.GetLastError());
    return;
}
// For application of StopListenServer, refer to refer to NetSDK.cs in C# Demo
NetSDKCS library
```

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 equipment 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 equipment (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 equipment 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 equipment network security:

1. Physical Protection

We suggest that you perform physical protection to equipment, especially storage devices. For example, place the equipment 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 equipment (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 equipment 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 equipment, 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 equipment 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 equipment, 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 equipment 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.