NetSDK_Python

Programming Manual

V2.0.0

Foreword

Purpose

Welcome to use NetSDK (hereinafter referred to be "SDK") programming manual (hereinafter referred to be "the manual").

The manual describes the main function modules, interfaces and calling relationships, and provides example codes.

The example codes provided in the manual are only for demonstrating the procedure and not assured to copy for use.

Readers

- SDK software development engineers
- Project managers
- Product managers

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
V2.0.0	Added intelligent event enumeration. Added the interface for the optimization plan in the login section.	February 2025
V1.0.4	Added sections 2.11, 2.12, 2.13, 2.14, 3.11, 3.12, 3.13, 4.11, and Appendix 4.	July 2024
V1.0.3	Add Appendix 2 Intelligent Events.	December 2023
V1.0.2	Updated some descriptions.	February 2023
V1.0.1	Change the packing method of package library to whl package.	October 2020
V1.0.0	First release.	May 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.

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.
Motion Detection Alarm	When detecting a moving object on the image, an motion detection alarm will be uploaded.

Table of Contents

1 Overview

1.1 General

The following are the main functions:

Device login, live view, record playback, record download, remote snapshot, alarm upload, device search, intelligent event upload and snapshot, device restart, device timing and more.

Table 1-1 Files of NetSDK library

Library Type	Library File Name	Library File Description
Function library	dhnetsdk.dll	Library file
	avnetsdk.dll	Library file
Configuration library	dhconfigsdk.dll	Library file
Play (coding and decoding auxiliary library	dhplay.dll	Play library
	fisheye.dll	Fishereye correction
Dependent library of "avnetsdk.dll"	Infra.dll	Base library
	json.dll	Json library
	NetFramework.dll	Network base library
	Stream.dll	Media transmission structure package library
	StreamSvr.dll	Stream service
Auxiliary library of " dhnetsdk "	IvsDrawer.dll	Image display library
	StreamConvertor.dll	Transcoding database

Table 1-2 Files of package project

File Name	File Description
NetSDK.py	Call NetSDK library to package the interfaces as Python interfaces which can be used by users.
SDK_Callback.py	Store the callbacks used by the NetSDK library.
SDK_Enum.py	Store the enumerations used by the NetSDK library.
SDK_Struct.py	Store the structures used by the NetSDK library.

$\square$

- The function library and configuration library are necessary libraries.
- The function library is the main body of SDK, which is used for interaction between client and products, remotely controls device, queries device data, configures device data information, and gets and handles the streams.
- NetSDK library is the base of the Python package project. In project, file NetSDK.py file defines the reference path of the NetSDK library, and you need to put the NetSDK library under the corresponding path when using it. Users can customize the reference path.
- All the externally used interfaces are defined in the NetClient class. Before using, you need to define an object of the NetClient class, and then call the interfaces in the class by the object.

1.2 Applicability

- Recommended memory: No less than 512 M
- Python version: 3.7 version and later
- Operating system:
  - ◇ Windows: Windows 10/Windows 8.1/Windows 7/2000 and Windows Server 2008/2003.
  - ◇ Linux: General Linux systems such as Red Hat/SUSE.

1.3 Demo Running

- Download and unzip the Python version of NetSDK development kit, then find the .whl file in the dist folder. The corresponding name might vary slightly with the system, such as "NetSDK-2.0.0.1-py3-none-win_amd64.whl" or "NetSDK-2.0.0.1-py3-none-linux_i686.whl".
- This file is the python installation package of the NetSDK package library. After installing this file, Demo can directly "import NetSDK" and use its content for easier development.

1.3.1 Installing whl File

Step 1   Install python3.7, and add the installation directory to the system environment variables.

Step 2   Start instruction terminal to run the following command to install pyqt5 and pyqt5-tools.

```
pip install pyqt5
pip install pyqt5-tools
```

Step 3   Open the command terminal in the directory where the whl file is saved, and then run the following command to install the plug-in.

```
pip install NetSDK-2.0.0.1-py3-none-win_amd64.whl
```

Notes

- In Windows, the installation file is installed in the "NetSDK" folder in the "\Lib\site-packages" directory of the Python installation directory. In Linux, the installation file is installed in the "NetSDK" folder in the "site-packages" directory of the Python installation directory. If you need to refer to or change the content, refer to the files in the directory. Plug-ins installed by users are stored in the "site-packages" directory. The above mentioned PyQt is also in this directory.
- If you need to uninstall the plug-in, use the command "pip uninstall NetSDK".
- If both python2 and python3 exist in the system, replace "pip" in the command with "pip3".
- After installing whl, you can import NetSDK to develop relevant functions of SDK. Programs developed by customers do not rely on PyQt.
- If the Internet does not work, the installation cannot be successful through running the above command. Go to pypi module of python official website (https://pypi.org/) to download the following plug-ins, install correct versions of plug-ins according to versions of the system and python. The installation sequence is: python_dotenv, click, PyQt5-sip, PyQt5, pyqt5-tools, PyQt5Designer.
- When installing plug-ins offline, open the command terminal in the plug-in directory, and then

run the command pip install xxx. When demonstrating locally, the commands used are as follows.(Names of Linux plug-ins might be different, there are no difference from Windows.

```
pip install python_dotenv-0.10.1-py2.py3-none-any.whl

pip install Click-7.0-py2.py3-none-any.whl

pip install PyQt5_sip-4.19.13-cp37-none-win_amd64.whl

pip install PyQt5-5.11.3-5.11.2-cp35.cp36.cp37.cp38-none-win_amd64.whl

pip install pyqt5_tools-5.11.3.1.4-cp37-none-win_amd64.whl

pip install PyQt5Designer-5.10.1-cp37-none-win_amd64.whl
```

1.3.2 Running Demo

After the whl packager is installed, you can directly run Demo.

Take live view Demo as an example:

Open the "RealPlayDemo" folder, enable the command terminal, and run the command "python RealPlayDemo.py" to start Demo.

In Windows, if the py file is opened by python, you can also directly double-click the RealPlayDemo.py file to start the program.

Notes

- If both python2 and python3 exist in the system, replace the command "python RealPlayDemo.py" with "python3 RealPlayDemo.py".
- In Windows, double-click the file to run Demo, and an additional console window will pop up at the back. If you want to hide the console window when running the program, you can change the suffix of RealPlayDemo.py to ".pyw" ("RealPlayDemo.pyw"), then double-click to run it.
- When using PyCharm for development, you only need to open each Demo directory in the Demo folder, instead of the whole directory.

1.4 Project Configuration

1.4.1 Pycharm Configuration

Configure Interpreter, and then run the Demo project by pycharm.

Step 1   Open pycharm.

Step 2   Select **File > Settings**.

Figure 1-1 Select settings
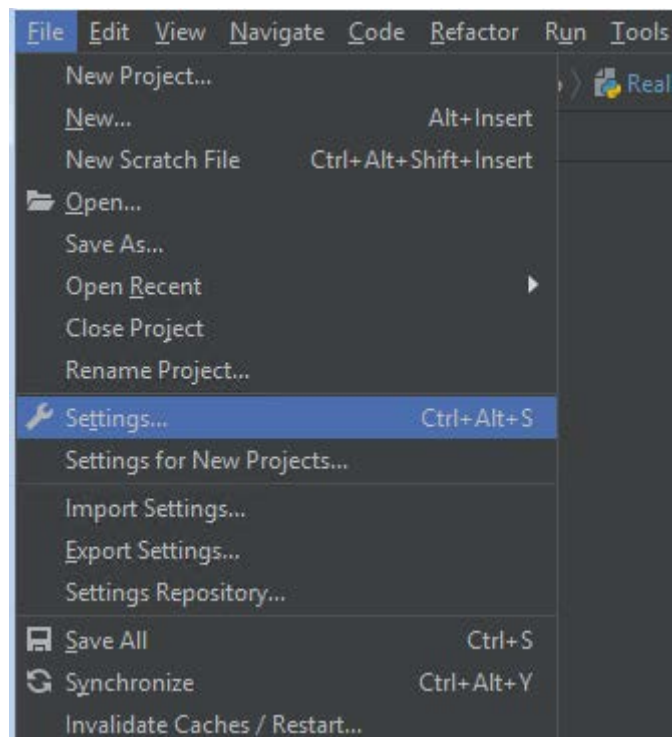
Step 3   Configure Interpreter.

Information about PyQt5 related software is displayed.

Figure 1-2 Configure interpreter
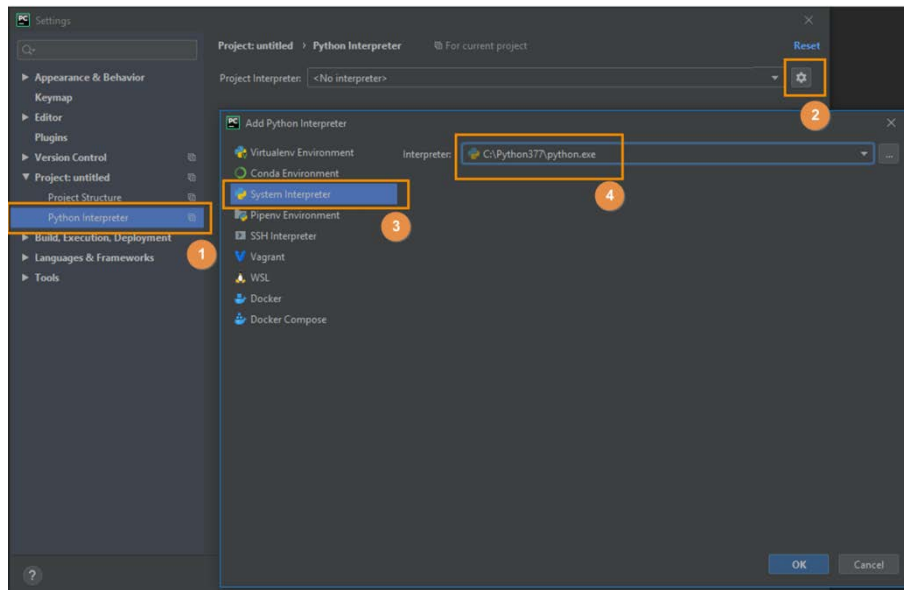
Step 1   Configure Demo.

1)   Select **Run > Debug**.

Figure 1-3 Configure Demo (1)
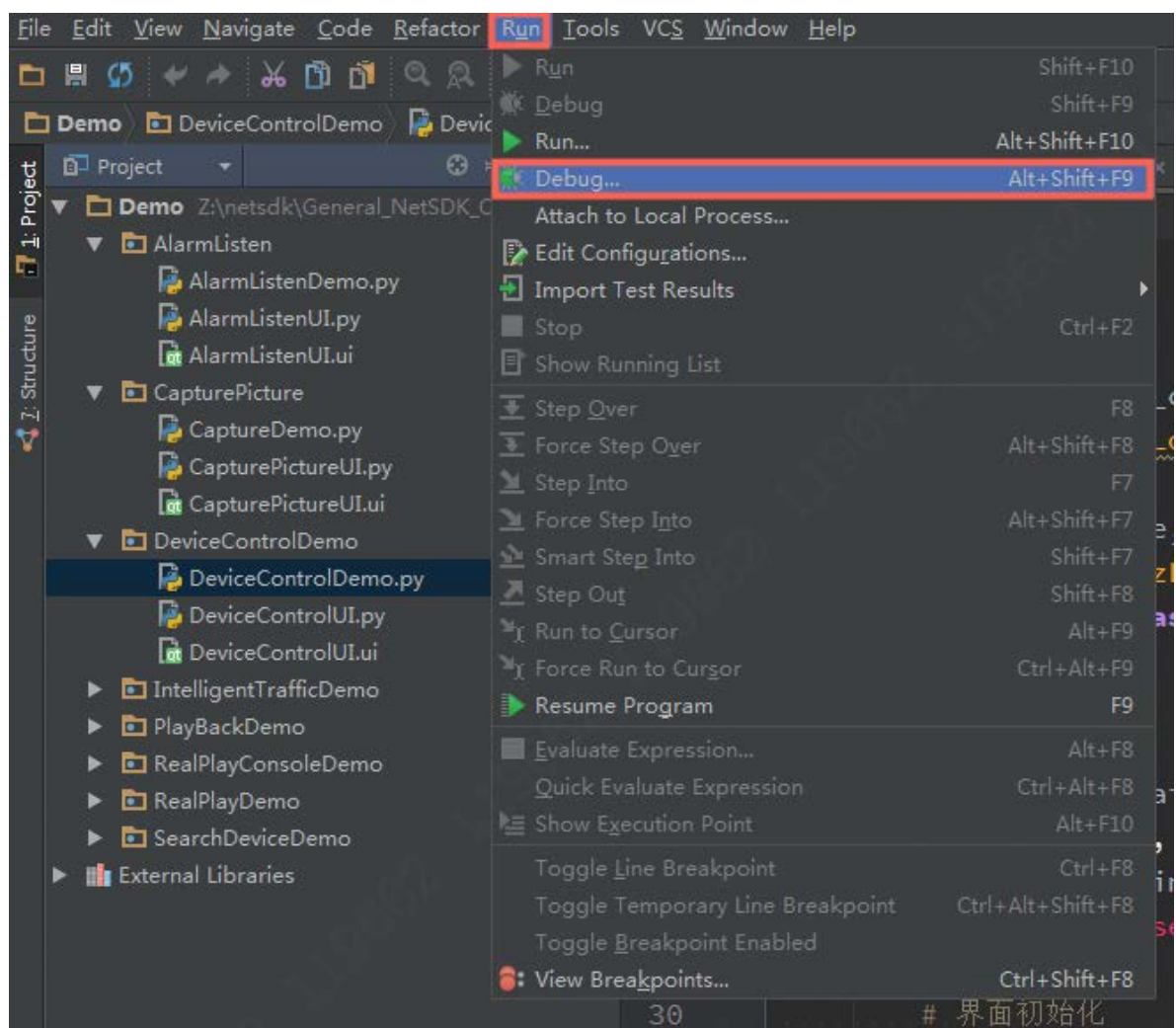
2)   Select **Edit Configurations**.

Figure 1-4 Configure Demo (2)
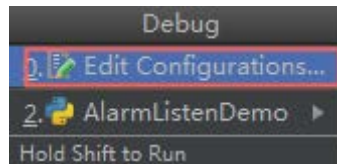
3) Select **+ > Python**.

Figure 1-5 Run Demo (3)
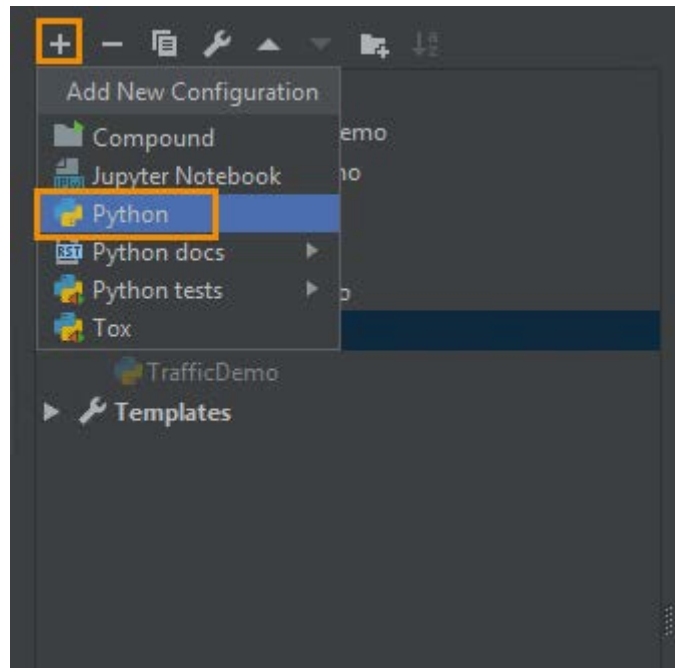
4) Set Demo configuration name and path of Demo.py.
   ◇ Name: Set Demo configuration name.
   ◇ Script path: Select path of Demo.py. Here takes PlayBackDemo.py as an example.

5) Click **Debug** to run Demo.

Figure 1-6 Run Demo configuration
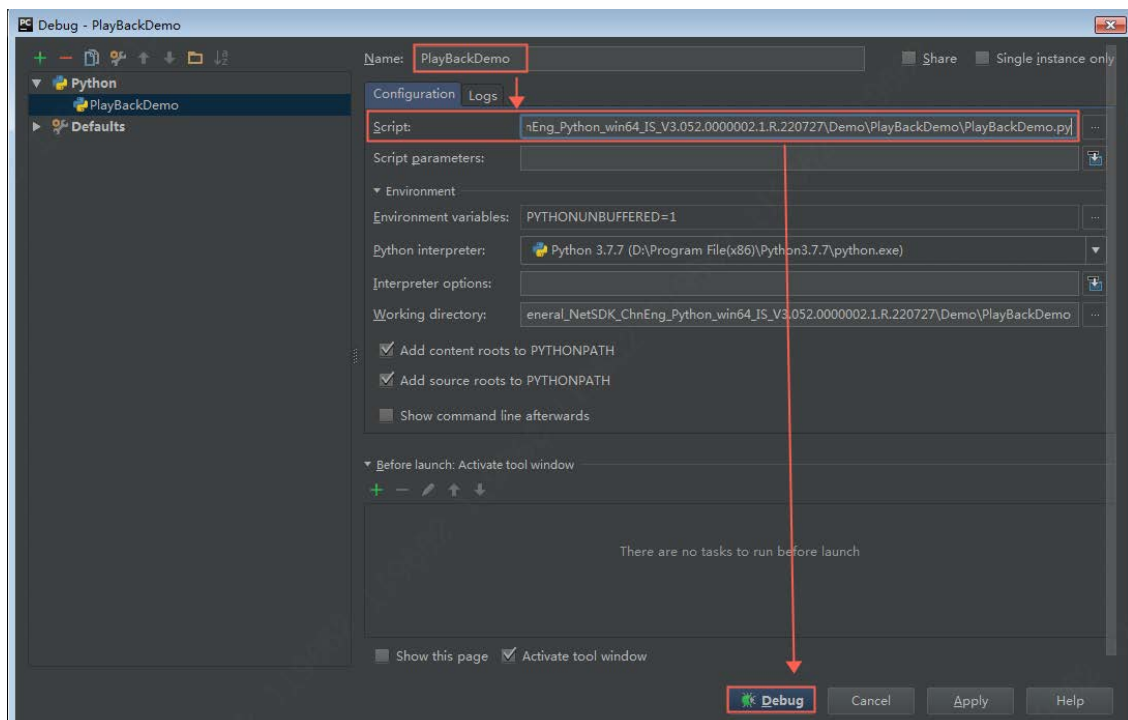
1.4.2 Adding Tool to Pycharm

Add pyqt5designer and pyuic5 to pycharm.

- After adding pyqt5designer to pycharm, select the corresponding ui file and open qt designer. Use the tool to design UI.
- After adding pyuic5 to pycharm, select the corresponding .ui file and create .py file. View the defined variables through the py file.

Step 1   Select File > Settings.

Figure 1-7 Select external tools
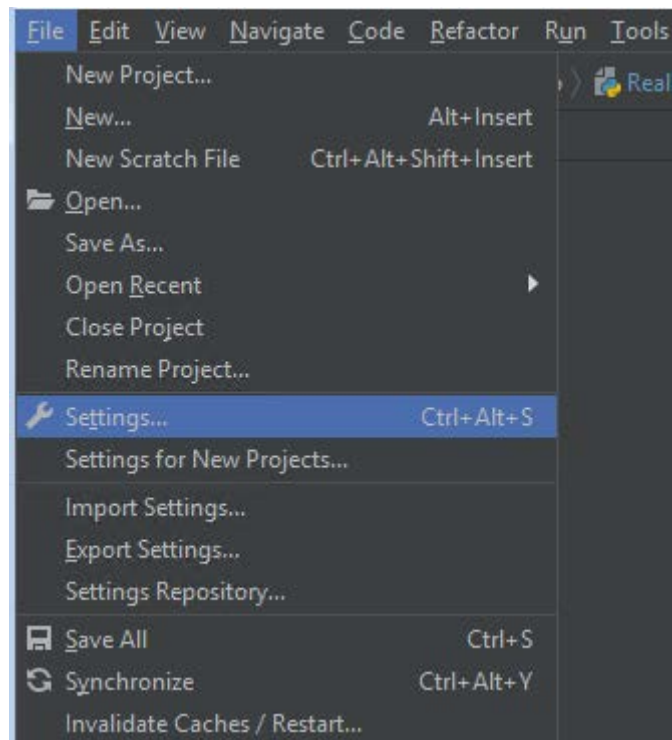
Step 2   Add pyqt5designer. Select **Tool > External Tools**, and click + to configure parameters. Click **OK**.

Figure 1-8 Add pyqt5designer
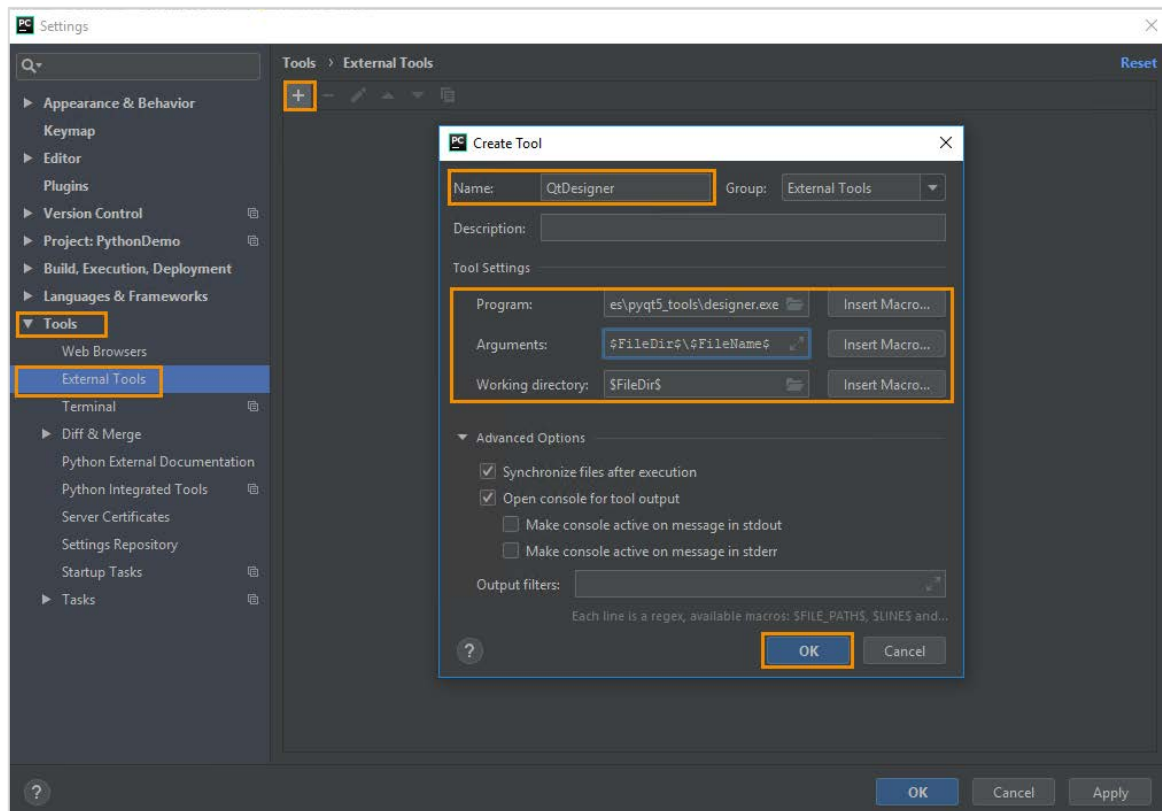
Table 1-3 Parameters of pyqt5designer

Paramater	Description
Name	Tool name which can be customized by users, such as QtDesigner.
Program	Enter the path of pyqt5designer.exe which is in the file folder of Scripts.
Arguments	$FileDir$\$FileName$
Working directory	$FileDir$

<u>Step 3</u>   Add pyuic5. Click + to configure parameters, and then click **OK**.

Figure 1-9 Add pyuic5
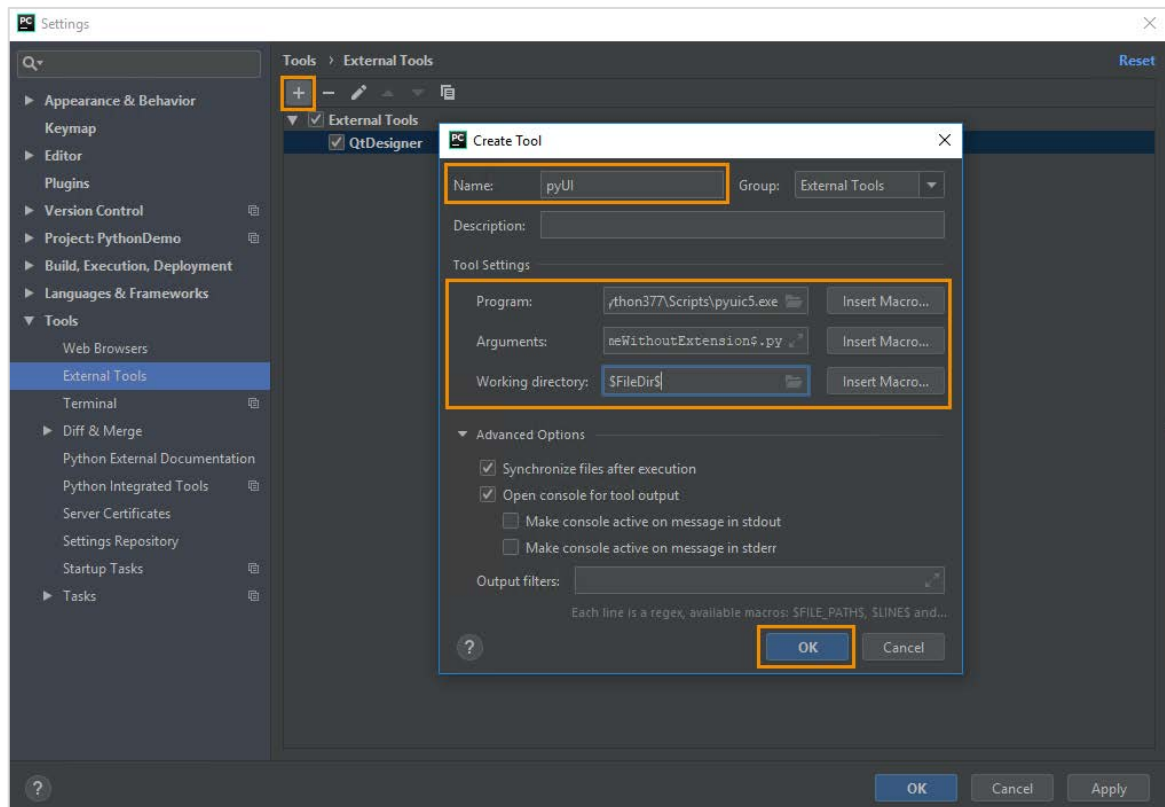
Table 1-4 Parameters of pyuic5

Paramater	Description
Name	Tool name which can be customized by users, such as PyUI.
Program	Enter the path of pyuic5.exe which is in the file folder of Scripts.
Arguments	$FileName$ -o $FileNameWithoutExtension$.py
Working directory	$FileDir$

Step 4   Use design interface of QtDesigner.

Select the corresponding .ui file, and right-click **External Tools > QtDesigner** to open QtDesigner.

Figure 1-10 Open QtDesigner
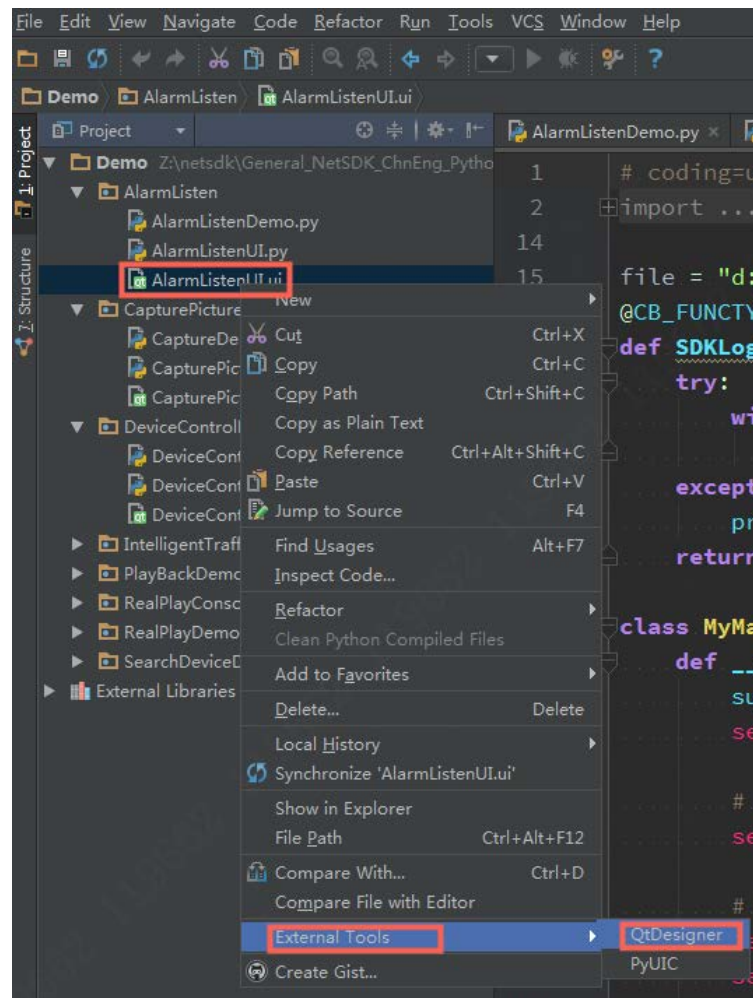
Figure 1-11 Design interface
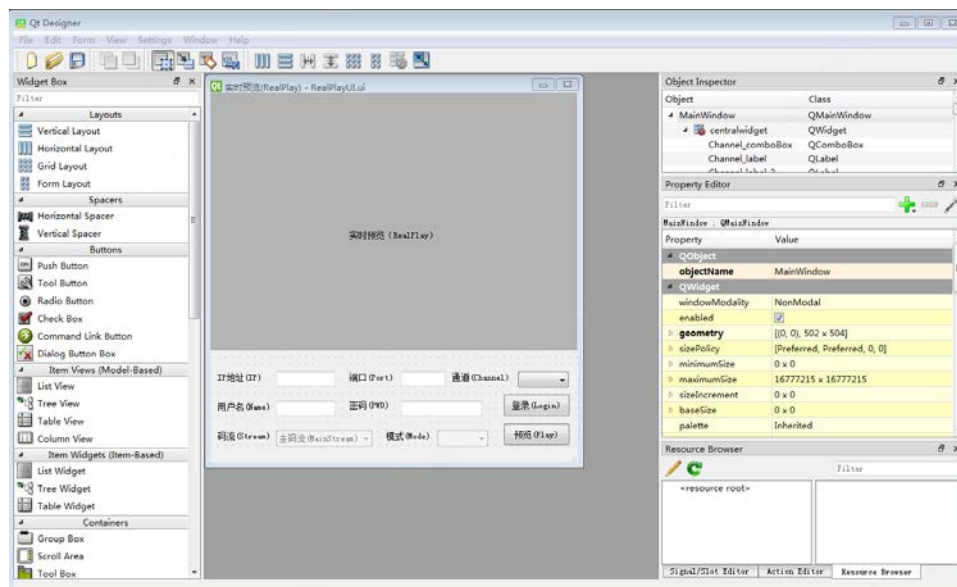
Step 5 Transform file from .ui format to.py format.

Click the corresponding file in .ui format, right click to open menu, and select **External Tools** > **pyuic5** to transform file format.

Figure 1-12 Transform file format from .ui to .py
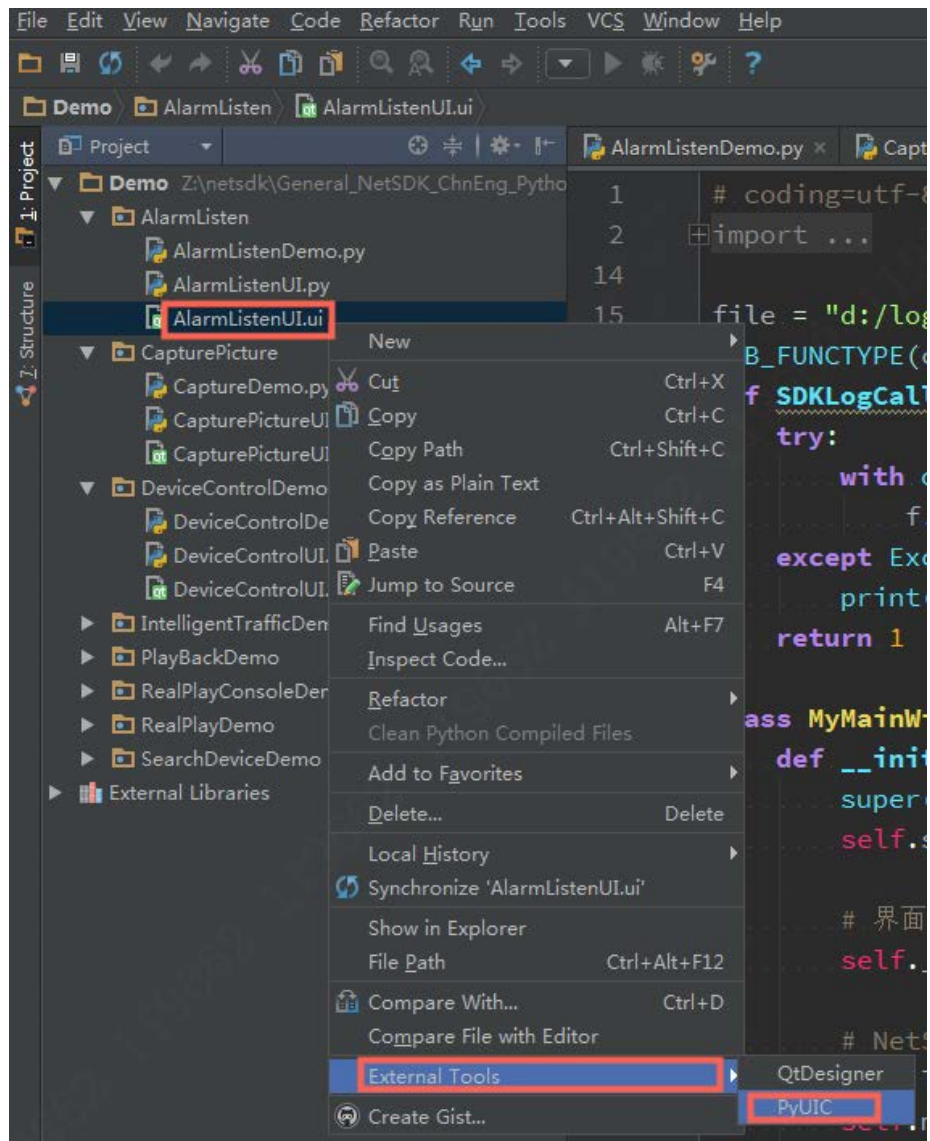
2 Function Modules

2.1 SDK Initialization

2.1.1 Introduction

Initialization is the first step of SDK 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 using this function, call cleanup interface to release SDK resource.
- The interfaces between **InitEx** and **Cleanup** are one-to-one corresponding. It is recommended to call it only once when writing codes.

2.1.2 Interface Overview

Table 2-1 Interfaces of initialization

Interface	Implication
_load_library	Load dynamic library.
InitEx	Initialize SDK.
SetAutoReconnect	(Optional) Set reconnection callback.
Cleanup	Release SDK sources.

2.1.3 Process
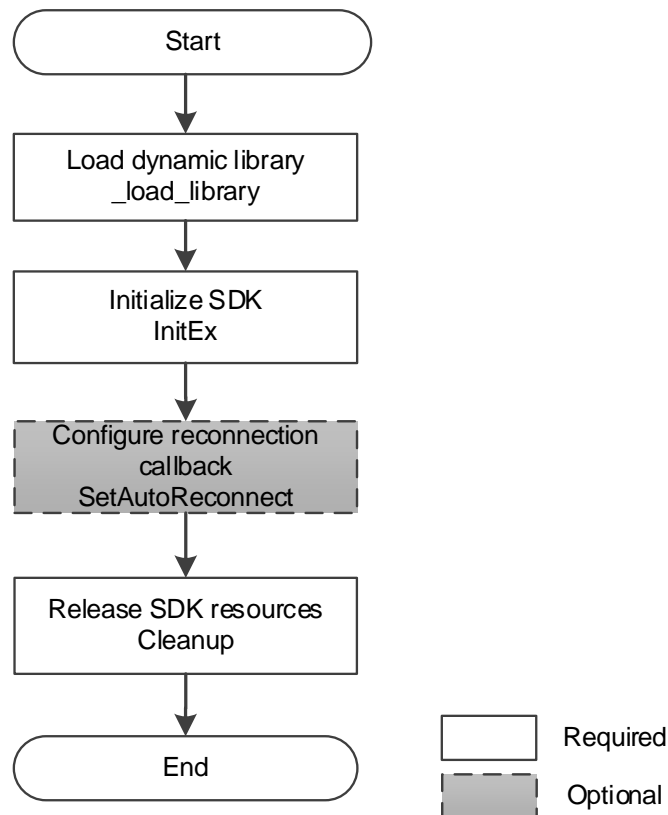
```
        ┌─────────────────────┐
        │        Start        │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐
        │ Load dynamic library│
        │    _load_library    │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐
        │   Initialize SDK    │
        │       InitEx        │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐
        │Configure reconnection│
        │      callback       │
        │   SetAutoReconnect  │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐
        │ Release SDK resources│
        │      Cleanup        │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐
        │         End         │
        └─────────────────────┘
```

☐ Required
▨ Optional

Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called with every function.

Process Description

Step 1   Call **_load_library** to load dynamic library.

Step 2   Call **InitEx** to initialize SDK and set disconnection callback.

Step 3   (Optional) Call **SetAutoReconnect** to set reconnection callback.

Step 4   Call **Cleanup** to release SDK resources. This function can be called after using NETSDK.

2.1.4 Sample Code

```
# state and initialize callback function
self.m_DisConnectCallBack = fDisConnect(self.DisConnectCallBack)
self.m_ReConnectCallBack = fHaveReConnect(self.ReConnectCallBack)
```

```
# get NetSDK object and initialize it
self.sdk = NetClient()
self.sdk.InitEx(self.m_DisConnectCallBack)
self.sdk.SetAutoReconnect(self.m_ReConnectCallBack)


# realize disconnection callback function
def DisConnectCallBack(self, lLoginID, pchDVRIP, nDVRPort, dwUser):
    self.setWindowTitle("live view (RealPlay)-disconnection (OffLine)")


# realize reconnection callback function
def ReConnectCallBack(self, lLoginID, pchDVRIP, nDVRPort, dwUser):
    self.setWindowTitle(' live view (RealPlay)-reconnection(OnLine)')


# release NetSDK resource
self.sdk.Cleanup()
```

2.1.5 Note

- InitEx only needed to be called before using NetSDK, which is at the beginning of running Demo. Cleanuponly need to be called after all functions related to NetSDK has been used to release NetSDK resources. These two interfaces do not need to be called each time the functions are used.
- _load_library is an internal callback of the NetClient which will be auto called when the NetClient class object is implemented. Here is just to remind users, if you need to change the location of NetSDK library, or to change the method and timing of calling NetSDK library, modify this function.
- Initialization: Call **InitEx** only once before using the SDK.
- Cleaning up: The interface **Cleanup** clears all the opened processes, such as login, live view, and alarm subscription.
- Reconnection: NetSDK can set the reconnection function for the situations such as network disconnection and power off. NetSDK will keep logging until succeeded. Only the live view, playback, smart event subscription and alarm subscription modules will be resumed after the connection is back.
- For callback details of example code, see "4 Callback Definition".

2.2 Device Search and Initialization

2.2.1 Introduction

Device search is mainly used to help user to get device info from network. Device search can work with login function. Device search interface can find relevant devices and login interface can login these devices.

Device search is classified into the following two types by whether crossing segment or not:

- Async same-segment device search: Search for device info within current segment.
- Sync cross-segment device search: According to user-set segment info, searching for device in corresponding segment.

2.2.2 Interface Overview

Table 2-2 Interface of device search and initialization

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
StartSearchDevicesEx	Asynchronously search for devices within the same networksegment.
StopSearchDevices	Stop asynchronously searching for devices within the same networksegment.
SearchDevicesByIPs	Stop asynchronously searching for devices in cross-segment.
InitDevAccount	Initialize device.
GetLastError	Get error codes of interfaces that fail to be called.

2.2.3 Process

2.2.3.1 Async Searching within Same Segment

Figure 2-2 Process of async device searching and initialization
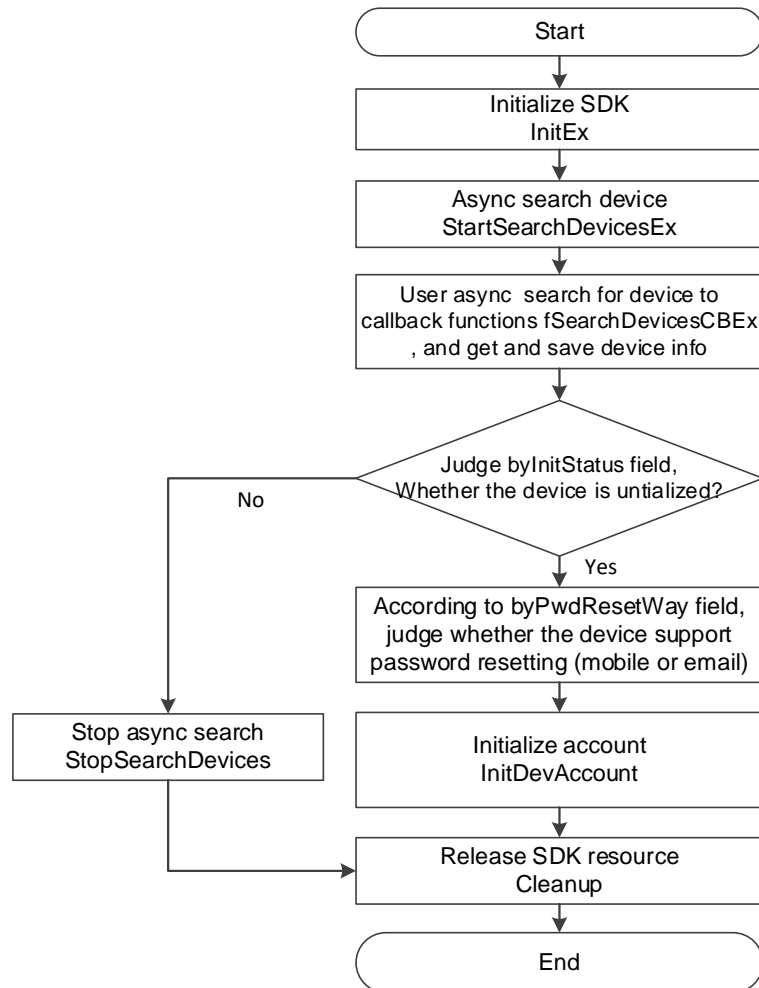
```
                              ┌──────────────────────┐
                              │        Start         │
                              └──────────┬───────────┘
                                         │
                              ┌──────────▼───────────┐
                              │    Initialize SDK    │
                              │        InitEx        │
                              └──────────┬───────────┘
                                         │
                              ┌──────────▼───────────┐
                              │  Async search device │
                              │  StartSearchDevicesEx│
                              └──────────┬───────────┘
                                         │
                       ┌─────────────────▼──────────────────┐
                       │ User async  search for device to    │
                       │ callback functions fSearchDevicesCBEx│
                       │ , and get and save device info       │
                       └─────────────────┬──────────────────┘
                                         │
                           ◇─────────────▼─────────────◇
            No             │  Judge byInitStatus field,  │
     ◄──────────────────── │ Whether the device is       │
                           │ untialized?                 │
                           ◇─────────────┬─────────────◇
                                         │ Yes
                              ┌──────────▼───────────┐
                              │According to byPwdResetWay field,│
                              │judge whether the device support│
                              │password resetting (mobile or email)│
                              └──────────┬───────────┘
     ┌──────────────────┐    ┌──────────▼───────────┐
     │ Stop async search│    │   Initialize account │
     │ StopSearchDevices│    │     InitDevAccount   │
     └────────┬─────────┘    └──────────┬───────────┘
              │                         │
              │              ┌──────────▼───────────┐
              └─────────────►│  Release SDK resource│
                             │       Cleanup        │
                             └──────────┬───────────┘
                                        │
                             ┌──────────▼───────────┐
                             │         End          │
                             └──────────────────────┘
```

Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called for every function.

Process Description

Step 1   Call **InitEx** to initialize SDK.

Step 2   Call **StartSearchDevicesEx** to search for devices.

Step 3   Find the uninitialized devices by search callback **fSearchDevicesCBEx**. Check that the device is uninitialized according to byInitStatus filed. Check that the password can be reset by cellphone or email according to byPwdResetWay field which is also required in interface initialization.

Step 4   Call **InitDevAccount** to initialize device.

Step 5   Call **StopSearchDevices** to stop searching.

Step 6   Call **Cleanup** to release SDK resource.

2.2.3.2 Sync Searching in Cross-segment

Figure 2-3 Process of sync search and initialization
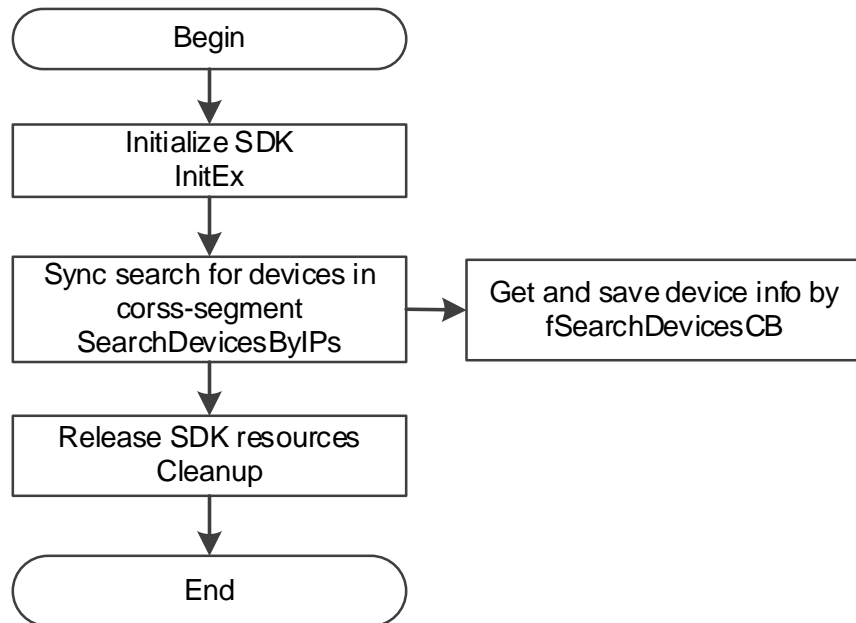
```
                    ┌──────────────────────┐
                    │        Begin         │
                    └──────────┬───────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │    Initialize SDK    │
                    │        InitEx        │
                    └──────────┬───────────┘
                               │
                               ▼
          ┌──────────────────────┐        ┌──────────────────────────┐
          │ Sync search for      │        │ Get and save device info │
          │ devices in           │───────▶│ by                       │
          │ corss-segment        │        │ fSearchDevicesCB         │
          │ SearchDevicesByIPs   │        └──────────────────────────┘
          └──────────┬───────────┘
                     │
                     ▼
          ┌──────────────────────┐
          │ Release SDK resources│
          │       Cleanup        │
          └──────────┬───────────┘
                     │
                     ▼
          ┌──────────────────────┐
          │         End          │
          └──────────────────────┘
```

Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called with every function.

Process Description

Step 1   Call **InitEx** to initialize SDK.

Step 2   Call **SearchDevicesByIPs** to search for devices. Get device info by **fSearchDevicesCB**.

Step 3   Call **Cleanup** to release SDK resource.

2.2.4 Sample Code

2.2.4.1 Async Searching within Same Segment and Device Initialization

Code Path

Demo\SearchDeviceDemo\ SearchDeviceDemo.py

Sample Code

```python
# multicast and broadcast search
    def start_search_device(self):
        # get local IP, search under taking multiple NIC
        # call searching interfaces for the number of NICs times
        IPList = self.getIPAddrs()
        nSuccess = 0
        for i in range(IPList.__len__()):
            startsearch_in = NET_IN_STARTSERACH_DEVICE()
            startsearch_in.dwSize = sizeof(NET_IN_STARTSERACH_DEVICE)
            startsearch_in.emSendType                                          =
EM_SEND_SEARCH_TYPE.MULTICAST_AND_BROADCAST
            startsearch_in.cbSearchDevices = search_device_callback
            startsearch_in.szLocalIp = IPList[i].encode()
            startsearch_out = NET_OUT_STARTSERACH_DEVICE()
            startsearch_out.dwSize = sizeof(NET_OUT_STARTSERACH_DEVICE)
            lSearchHandle = self.sdk.StartSearchDevicesEx(startsearch_in, startsearch_out)
            if lSearchHandle != 0:
                nSuccess += 1
                self.lSearchHandle_list.append(lSearchHandle)
        if(IPList.__len__() > 0):
            del IPList
        if(nSuccess > 0):
            return True
        else:
            return False


    # stop searching. Use with start_search_device
    def stop_search_device(self):
        for i in range(self.lSearchHandle_list.__len__()):
            result = self.sdk.StopSearchDevices(self.lSearchHandle_list[i])
```

```python
        nUpdateNum = 0
        self.lSearchHandle_list.clear()
        self.device_info_list.clear()
        self.device_mac_list.clear()
        self.tableWidget.clear()
        self.row = 0
        self.column = 0
        device_queue.queue.clear()
        if(not device_queue.empty()):
            device_queue.task_done()
        self.tableWidget.setHorizontalHeaderLabels(['(No.)', '(Status)', 'IP(IP Version)', '(IP Address)',
(Port)', ' (Subnet Mask)', ' (Gateway)', ' (Mac Address)', '(Device Type)', '(Detail Type)', 'Http(Http)'])
        return


    def Init_Btn(self):
        # get selected ip and initialization info
        currentRow = self.tableWidget.currentRow()
        if((len(self.device_info_list) ==0)or((self.device_info_list[currentRow][0]&3) != 1)):
            QMessageBox.about(self, '(prompt)', " (Please select not initialized device)")
        else:
            result = self.init_device_accout(self.device_info_list[currentRow])
            if result == True:
                QMessageBox.about(self, ' (prompt)', "(Initialize Success)")
                item = QTableWidgetItem("(Initialize)")
                self.device_info_list[currentRow][0] = 2
                self.tableWidget.setItem(currentRow, 1, item)
                self.tableWidget.update()
                self.tableWidget.viewport().update()


    # initialize account
    def init_device_accout(self, device_info:list):
        child = QDialog()
        child_ui = Ui_InitDevAccount()
        child_ui.setupUi(child)
        if (1 == (device_info[3] & 1)):
            # mobile phone
            child_ui.way_lineEdit.setText('(Phone)')
        elif (1 == (device_info[3] >> 1 & 1)):
            # email
```

```python
            child_ui.way_lineEdit.setText('Mail)')
        value = child.exec()
        if (value == 0):
            return False
        init_Account_In = NET_IN_INIT_DEVICE_ACCOUNT()
        init_Account_In.dwSize = sizeof(init_Account_In)
        init_Account_In.szMac = device_info[2]
        username = child_ui.username_lineEdit.text()
        password = child_ui.password_lineEdit.text()
        confirm_password = child_ui.confirm_password_lineEdit.text()
        if(password != confirm_password):
            QMessageBox.about(self, '(prompt)', "(Confirm  password  is  wrong ，  please  input
again)")
            return
        init_Account_In.szUserName = username.encode()
        init_Account_In.szPwd = password.encode()
        init_Account_In.szCellPhone = child_ui.reset_way_lineEdit.text().encode()
        if (1 == (device_info[3] & 1)):
            # mobile phone
            init_Account_In.szCellPhone = child_ui.reset_way_lineEdit.text().encode()
        elif(1 == (device_info[3] >> 1 & 1)):
            # email
            init_Account_In.szMail = child_ui.reset_way_lineEdit.text().encode()
        init_Account_In.byPwdResetWay = device_info[3]
        init_Account_Out = NET_OUT_INIT_DEVICE_ACCOUNT()
        init_Account_Out.dwSize = sizeof(init_Account_Out)

        result = self.sdk.InitDevAccount(init_Account_In, init_Account_Out, 5000, device_info[4])
        if result:
            return True
        else:
            QMessageBox.about(self, '(prompt)', 'error:' + str(self.sdk.GetLastError()))

            return False
```

2.2.4.2 Sync Searching in Cross-segment

Code Path

Demo\SearchDeviceDemo\ SearchDeviceDemo.py

Sample Code

```
# unicast search
    def start_search_device_byIP(self, start_IP, end_IP): #pay attention to validity of each IP address
        startsearchbyIp_in = DEVICE_IP_SEARCH_INFO()
        startsearchbyIp_in.dwSize = sizeof(DEVICE_IP_SEARCH_INFO)
        start = struct.unpack("!I", socket.inet_aton(start_IP))[0]    # network sequence transformed
to byte-order
        end = struct.unpack("!I", socket.inet_aton(end_IP))[0]
        if (end - start > 255):
            QMessageBox.about(self, '(prompt)', "256(Number of IP addresses exceeds the upper
limit 256.)")
            return False


        startsearchbyIp_in.nIpNum = end - start + 1


        for i in range(startsearchbyIp_in.nIpNum):
            ip = DEVICE_IP_SEARCH_INFO_IP()
            ip.IP = socket.inet_ntoa(struct.pack("!I", start + i)).encode()
            startsearchbyIp_in.szIP[i] = ip


        wait_time = int(wnd.Searchtime_lineEdit.text())
        # get local IP, search under multiple NICs
        # Call searching interface according the number of NICs
        IPList = self.getIPAddrs()
        nSuccessNum = 0
        for i in range(IPList.__len__()):
            result = self.sdk.SearchDevicesByIPs(startsearchbyIp_in, search_devie_byIp_callback, 0,
IPList[i].encode(), wait_time)
            if result:
                nSuccessNum =+ 1
        if (IPList.__len__() > 0):
            del IPList
        if(nSuccessNum > 0):
            return True
        else:
            return False
```

2.3 Device Login

2.3.1 Introduction

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

You will obtain a unique login ID upon log in to the device and should introduce login ID before using other SDK interfaces. The login ID becomes invalid once logged out.

2.3.2 Interface Overview

Table 2-3 Interfaces of device login

Interface	Implication
InitEx	Initialize SDK.
SetAutoReconnect	Set reconnection callback.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
Logout	Log out.
SetOptimizeMode	Set the optimization plan. It is used to optimize login time for NVR.

2.3.3 Process

Figure 2-4 Process of login
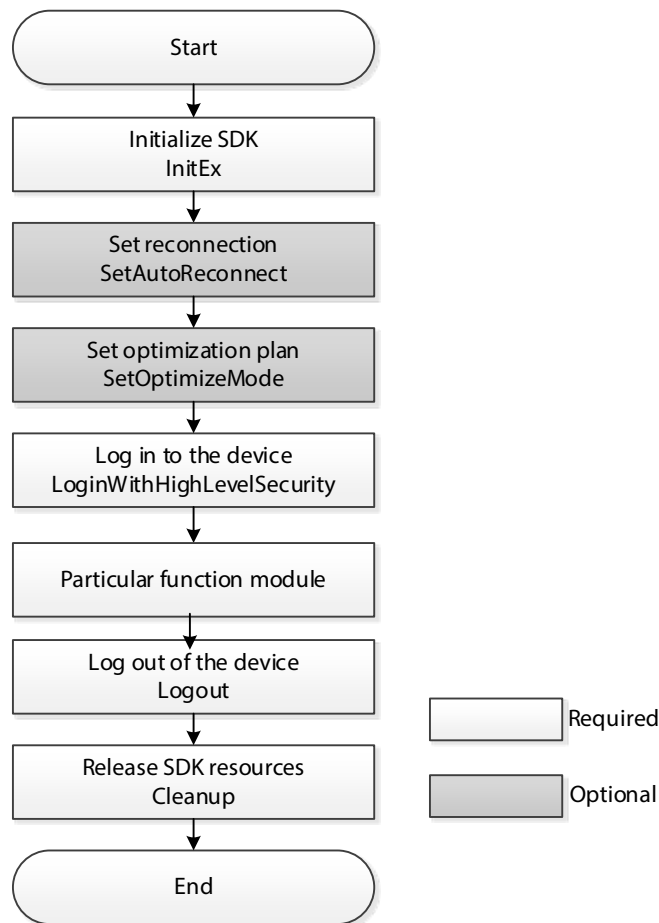
Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called with every function.

Process Description

Step 1   Call **InitEx** to initialize SDK.

Step 2   Call **SetAutoReconnect** to set reconnection callback.

Step 3   Call **SetOptimizeMode** to set the optimization plan.

Step 4   Call **LoginWithHighLevelSecurity** to log in to the device.

Step 5   Implement the required function modules.

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

Step 7   Call **Cleanup** to release SDK resources.

2.3.4 Sample Code

```
# log in to the device to get login handle and device info. If failed, error info will be displayed
//This operation is optional. Optimize obtaining hard disk information.
opt = ctypes.c_int(EM_OPTTYPE_MOBILE_TYPE.OPTTYPE_MOBILE_DISK_INFO)
self.sdk.SetOptimizeMode(EM_OPTIMIZE_TYPE.OPT_TYPE_MOBILE_OPTION, addressof(opt))


stuInParam = NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY()
stuInParam.dwSize = sizeof(NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY)
stuInParam.szIP = ip.encode()
stuInParam.nPort = port
stuInParam.szUserName = username.encode()
stuInParam.szPassword = password.encode()
stuInParam.emSpecCap = EM_LOGIN_SPAC_CAP_TYPE.TCP
stuInParam.pCapParam = None


stuOutParam = NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY()
stuOutParam.dwSize = sizeof(NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY)


self.loginID, device_info, error_msg = self.sdk.LoginWithHighLevelSecurity(stuInParam, stuOutParam)
if self.loginID != 0:
        for i in range(int(device_info.nChanNum)):
            self.Channel_comboBox.addItem(str(i))    # display channels of the device
    else:
        QMessageBox.critical(self, '(prompt)', error_msg, QMessageBox.Ok, QMessageBox.No)    #
display error info of the login interface


# log out
result = self.sdk.Logout(self.loginID)
    if result:
        self.loginID = 0
```

2.3.5 Note

- Login handle: When the login is successful, the returned value is not 0 (even the handle is smaller than 0, the login is also successful). One device can login multiple times with different handles at each login. If there is no special function module, it is suggested to login only once. The login handle can be repeatedly used on other function modules.
- Duplicate handles: It is normal that the login handle is the same as the existed handle. For example, log in to device A and get handle loginIDA. However, if you log out of loginIDA and

then log in, you may get LoginIDA again. But the duplicate handles do not occur throughout the lifetime of the handle.

- Logout: The interface will release the opened functions internally, but it is not suggested to rely on the cleaning up function of lougout. For example, if you opened the live view function, you should call the interface that stops the live viewg function when it is no longer required.
- Use login and logout in pairs: The login consumes some memory and socket information and release sources once logout.
- Login failure: It is suggested to check the failure through return parameter error_msg. for more details, see the error code list in **LoginWithHighLevelSecurity**.
- After reconnection, the original login ID will be invalid. After the device is auto reconnected, the login ID will take effect again.

2.4 Live View

2.4.1 Introduction

Live view obtains the real-time stream from the storage device or front-end device, which is an important part of the surveillance system.

SDK can get the main stream and sub stream from the device once it logged.
- Supports calling the window handle for SDK to directly decode and play the stream (Windows system only).
- Supports calling the real-time stream to you to perform independent treatment.
- Supports saving the real-time record to the specific file though saving the callback stream or calling the SDK interface.

2.4.2 Interface Overview

Table 2-4 Interfaces of live view

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
Logout	Log out.
RealPlayEx	Start live view extension interface.
StopRealPlayEx	Stop live view extension interface.
GetLastError	Get error codes of interfaces that fail to be called.
GetLastErrorMessage	Get error info of interfaces that fail to be called.

2.4.3 Process

Figure 2-5 Process of live view
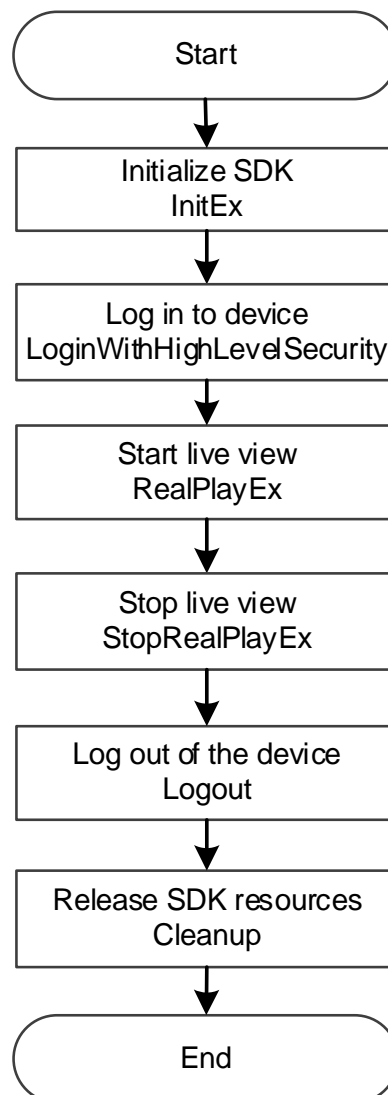
```
          ┌─────────────────────┐
          │        Start        │
          └─────────────────────┘
                     │
                     ▼
          ┌─────────────────────┐
          │    Initialize SDK   │
          │        InitEx       │
          └─────────────────────┘
                     │
                     ▼
          ┌─────────────────────────────┐
          │      Log in to device       │
          │  LoginWithHighLevelSecurity  │
          └─────────────────────────────┘
                     │
                     ▼
          ┌─────────────────────┐
          │    Start live view  │
          │      RealPlayEx     │
          └─────────────────────┘
                     │
                     ▼
          ┌─────────────────────┐
          │    Stop live view   │
          │    StopRealPlayEx   │
          └─────────────────────┘
                     │
                     ▼
          ┌─────────────────────┐
          │  Log out of the device │
          │        Logout       │
          └─────────────────────┘
                     │
                     ▼
          ┌─────────────────────┐
          │ Release SDK resources │
          │        Cleanup      │
          └─────────────────────┘
                     │
                     ▼
          ┌─────────────────────┐
          │         End         │
          └─────────────────────┘
```

Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called with every function.

Process Description

Step 1   Call **InitEx** to initialize SDK.
Step 2   Call **LoginWithHighLevelSecurity** to log in to the device.
Step 3   Call **RealPlayEx** to start live view.
Step 4   Call **StopRealPlayEx** to stop live view.

Step 5　Call **Logout** to log out of the device.

Step 6　Call **Cleanup** to release SDK resources.

2.4.4 Sample Code

```
#Start live view
channel = self.Channel_comboBox.currentIndex()     # channel No.
if self.StreamTyp_comboBox.currentIndex() == 0:
    stream_type = SDK_RealPlayType.Realplay # main streaam
else:
    stream_type = SDK_RealPlayType.Realplay_1 # sun stream
self.playID = self.sdk.RealPlayEx(self.loginID, channel, self.PlayWnd.winId(), stream_type)
if self.playID != 0:
    self.play_btn.setText("(Stop)")
    self.StreamTyp_comboBox.setEnabled(False)
else:
    QMessageBox.critical(self, ' (prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)

# Stop live view
result = self.sdk.StopRealPlayEx(self.playID)
if result:
    self.playID = 0
    self.PlayWnd.repaint()
```

2.4.5 Notes for Process

- **GetLastError** is the interface used to get the error codes when failed to call NetSDK interfaces. **GetLastErrorMessage** is the interface to get error information.
- It is recommended to call **GetLastErrorMessage** to get error information to identify the cause of the error.

2.5 Record Playback

2.5.1 Introduction

Record playback function plays the videos of a particular period in some channels to find the target videos for check.

The playback includes the following functions: Start playback, pause Playback, resume playback, and stop playback.

2.5.2 Interface Overview

Table 2-5 Interfaces of record playback

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
Logout	Log out.
PlayBackByTimeEx2	Extension interface of playback by time.
StopPlayBack	Stop playback.
PausePlayBack	Stop or resume playback.
SetDeviceMode	Set device mode.
QueryRecordFile	Query for all the record files within a period.

2.5.3 Process

After SDK initialization, you need to input channel number, start time, stop time, and valid window handle to realize the playback of the required record.

Figure 2-6 Process of record playback
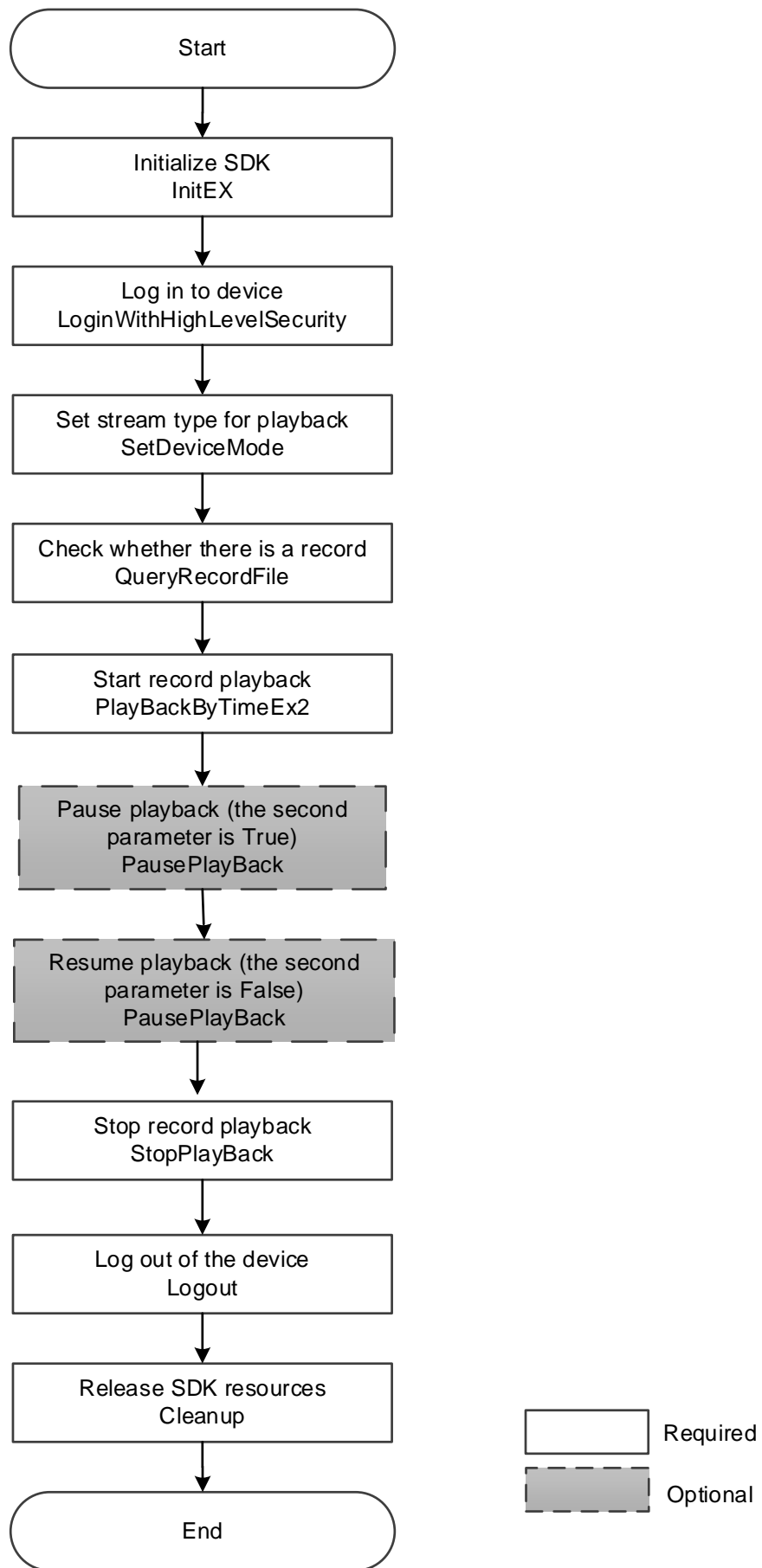
```
                    ┌──────────────────────┐
                    │        Start         │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │    Initialize SDK    │
                    │        InitEX        │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │   Log in to device   │
                    │ LoginWithHighLevelSecurity │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │ Set stream type for playback │
                    │     SetDeviceMode    │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │ Check whether there is a record │
                    │    QueryRecordFile   │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │  Start record playback │
                    │   PlayBackByTimeEx2   │
                    └──────────────────────┘
                               │
                               ▼
                    ┌ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
                    │ Pause playback (the second │
                    │  parameter is True)  │
                    │    PausePlayBack     │
                    └ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
                               │
                               ▼
                    ┌ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
                    │ Resume playback (the second │
                    │  parameter is False) │
                    │    PausePlayBack     │
                    └ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │  Stop record playback │
                    │    StopPlayBack      │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │  Log out of the device │
                    │        Logout        │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │ Release SDK resources │
                    │       Cleanup        │
                    └──────────────────────┘
                               │
                               ▼
                    ┌──────────────────────┐
                    │         End          │
                    └──────────────────────┘
```

Legend:
- ☐ Required
- ▨ Optional

Process Description

Step 1 Call **InitEx** to initialize SDK.

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

Step 3 Call **SetDeviceMode** to set the stream type.

Step 4 Call **QueryRecordFile** to check whether there is a record in the selected period.

Step 5 Call **PlayBackByTimeEx2** to start playback.

Step 6 (Optional) Call **PausePlayBack**. The playback will pause when the second parameter is True.

Step 7 (Optional) Call **PausePlayBack**. The playback will resume when the second parameter is False.

Step 8 Call **StopPlayBack** to stop playback.

Step 9 Call **Logout** to log out of the device.

Step 10 Call **Cleanup** to release SDK resources.

2.5.4 Sample Code

```
# configure stream type for playback. Main stream is configured here.

stream_type = c_int(0)

result = self.sdk.SetDeviceMode(self.loginID, int(EM_USEDEV_MODE.RECORD_STREAM_TYPE),
stream_type)

if not result:

    QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)


# query record file

result, fileCount, infos = self.sdk.QueryRecordFile(self.loginID, 0, int(EM_QUERY_RECORD_TYPE.ALL),
startTime, endTime, None, 5000, False)


# Enable video playback

inParam = NET_IN_PLAY_BACK_BY_TIME_INFO()

inParam.hWnd = self.PlayBackWnd.winId()

inParam.cbDownLoadPos = DownLoadPosCallBack

inParam.dwPosUser = 0

inParam.fDownLoadDataCallBack = DownLoadDataCallBack

inParam.dwDataUser = 0

inParam.nPlayDirection = 0

inParam.nWaittime = 5000

inParam.stStartTime.dwYear = start_time.dwYear

inParam.stStartTime.dwMonth = start_time.dwMonth

inParam.stStartTime.dwDay = start_time.dwDay

inParam.stStartTime.dwHour = start_time.dwHour

inParam.stStartTime.dwMinute = start_time.dwMinute
```

```python
inParam.stStartTime.dwSecond = start_time.dwSecond
inParam.stStopTime.dwYear = end_time.dwYear
inParam.stStopTime.dwMonth = end_time.dwMonth
inParam.stStopTime.dwDay = end_time.dwDay
inParam.stStopTime.dwHour = end_time.dwHour
inParam.stStopTime.dwMinute = end_time.dwMinute
inParam.stStopTime.dwSecond = end_time.dwSecond
outParam = NET_OUT_PLAY_BACK_BY_TIME_INFO()

nchannel = self.Channel_comboBox.currentIndex()
self.playbackID = self.sdk.PlayBackByTimeEx2(self.loginID, nchannel, inParam, outParam)
if self.playbackID != 0:
    self.PlayBack_pushbutton.setText("(Stop)")
    self.Pause_pushbutton.setEnabled(True)
    self.Channel_comboBox.setEnabled(False)
    self.StreamTyp_comboBox.setEnabled(False)
    self.Channel_comboBox.repaint()
    self.StreamTyp_comboBox.repaint()
    self.PlayBackWnd.repaint()
else:
    QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)

# Pause video playback
result = self.sdk.PausePlayBack(self.playbackID, True)

# resume video playback
result = self.sdk.PausePlayBack(self.playbackID, False)

# stop playback
result = self.sdk.StopPlayBack(self.playbackID)
if result:
    self.playbackID = 0
if not result:
    QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)
```

2.6 Record Download

2.6.1 Introduction

The record download function helps you obtain the records saved on the device through SDK and save into the local. It allows you to download from the selected channels and export to the local disk or external USB flash drive. The downloaded files are in the format of Dahua which requires Dahua player or integrated Dahua playsdk to play.

2.6.2 Interface Overview

Table 2-6 Interfaces of record download

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
Logout	Log out.
SetDeviceMode	Set device mode.
DownloadByTimeEx	Download by time.
StopDownload	Stop record download.

2.6.3 Process

You can import the start time and end time of download. SDK can download the specified record file and save it to the required place.

You can also provide a callback pointer to SDK which calls back the specified record file to you.

Figure 2-7 Process of download by time
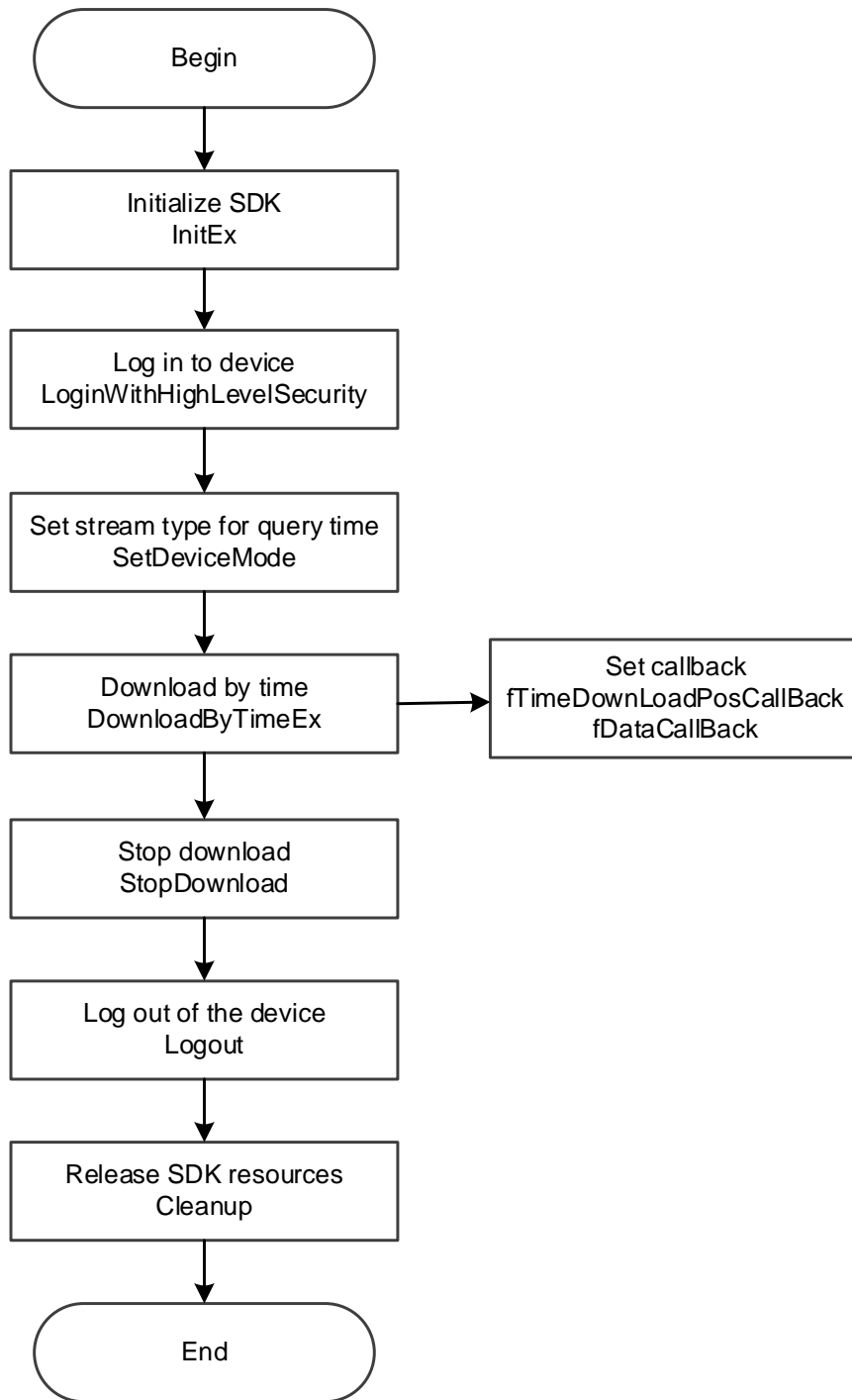
```
                    ┌─────────────────┐
                    │      Begin      │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │  Initialize SDK │
                    │      InitEx     │
                    └─────────────────┘
                             │
                             ▼
                    ┌──────────────────────────┐
                    │     Log in to device     │
                    │ LoginWithHighLevelSecurity│
                    └──────────────────────────┘
                             │
                             ▼
                    ┌──────────────────────────┐
                    │ Set stream type for query time │
                    │       SetDeviceMode      │
                    └──────────────────────────┘
                             │
                             ▼
            ┌──────────────────────┐        ┌──────────────────────────┐
            │   Download by time   │───────▶│       Set callback       │
            │   DownloadByTimeEx   │        │ fTimeDownLoadPosCallBack  │
            └──────────────────────┘        │      fDataCallBack       │
                             │              └──────────────────────────┘
                             ▼
                    ┌─────────────────┐
                    │  Stop download  │
                    │  StopDownload   │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │Log out of the device│
                    │     Logout      │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │Release SDK resources│
                    │     Cleanup     │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │       End       │
                    └─────────────────┘
```

Process Description

Step 1   Call **InitEx** to initialize SDK.

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

Step 3   Call **SetDeviceMode** to set the download stream type.

Step 4   Call **DownloadByTimeEx** to start downloading by time.

Step 5   Call **StopDownload** to stop download.

Step 6   (Optional) Call **fTimeDownLoadPosCallBack** to update the download progress.

Step 7   Call **Logout** to log out of the device.

Step 8　Call **Cleanup** to release SDK resources.

2.6.4 Example Code

```
# configure stream type for download. Main stream is configured here.
stream_type = c_int(0)
result = self.sdk.SetDeviceMode(self.loginID, int(EM_USEDEV_MODE.RECORD_STREAM_TYPE),
stream_type)
if not result:
    QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)


# enable video download
start_date = self.Start_dateTimeEdit.date()
start_time = self.Start_dateTimeEdit.time()
startDateTime = NET_TIME()
startDateTime.dwYear = start_date.year()
startDateTime.dwMonth = start_date.month()
startDateTime.dwDay = start_date.day()
startDateTime.dwHour = start_time.hour()
startDateTime.dwMinute = start_time.minute()
startDateTime.dwSecond = start_time.second()


end_date = self.End_dateTimeEdit.date()
end_time = self.End_dateTimeEdit.time()
enddateTime = NET_TIME()
enddateTime.dwYear = end_date.year()
enddateTime.dwMonth = end_date.month()
enddateTime.dwDay = end_date.day()
enddateTime.dwHour = end_time.hour()
enddateTime.dwMinute = end_time.minute()
enddateTime.dwSecond = end_time.second()


save_file_name = 'D:\savedata.dav'# folder path and name of files saved
nchannel = self.Channel_comboBox.currentIndex()
self.downloadID = self.sdk.DownloadByTimeEx(self.loginID, nchannel,
int(EM_QUERY_RECORD_TYPE.ALL), startDateTime, enddateTime, save_file_name,
TimeDownLoadPosCallBack, 0, DownLoadDataCallBack, 0)
if self.downloadID:
    self.Download_pushButton.setText("(Stop)")
else:
```

```python
        QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)


# Stop video download
result = self.sdk.StopDownload(self.downloadID)
if result:
    self.downloadID = 0


#callback function
@WINFUNCTYPE(None, c_longlong, c_ulong, POINTER(c_ubyte), c_ulong, c_longlong)
def DownLoadDataCallBack(lPlayHandle, dwDataType, pBuffer, dwBufSize, dwUser):
    pass


@WINFUNCTYPE(None, c_longlong, c_ulong, c_ulong, c_int, POINTER(NET_RECORDFILE_INFO),
c_ulong)
def TimeDownLoadPosCallBack(lPlayHandle, total_size, download_size, index, recordfileinfo,
dwUser):
    try:
        # display progress
        if download_size == 0xffffffff:
            self.downloadID = 0
            self.Download_progressBar.setValue(0)
            self.sdk.StopDownload(self.downloadID)
            self.Download_pushButton.setText("download)")
            self.Message_label.setText("Download End!")
        elif download_size == 0xfffffffe:
            self.downloadID = 0
            self.Download_progressBar.setValue(0)
            self.Download_pushButton.setText(" (download)")
            self.Message_label.setText("Download Failed!")
        else:
            if download_size >= total_size:
                self.Download_progressBar.setValue(100)
            else:
                percentage = int(download_size * 100 / total_size)
                self.Download_progressBar.setValue(percentage)
    except Exception as e:
        print(e)
    except Exception as e:
        print(e)
```

2.7 Device Control

2.7.1 Introduction

Get and set device time, and restart device remotely.

2.7.2 Interface Overview

Table 2-7 Interfaces of device control

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
Logout	Log out.
GetDevConfig	Query configuration info.
SetDevConfig	Set configuration info.
RebootDev	Restart device.

2.7.3 Process

Figure 2-8 Process of device control
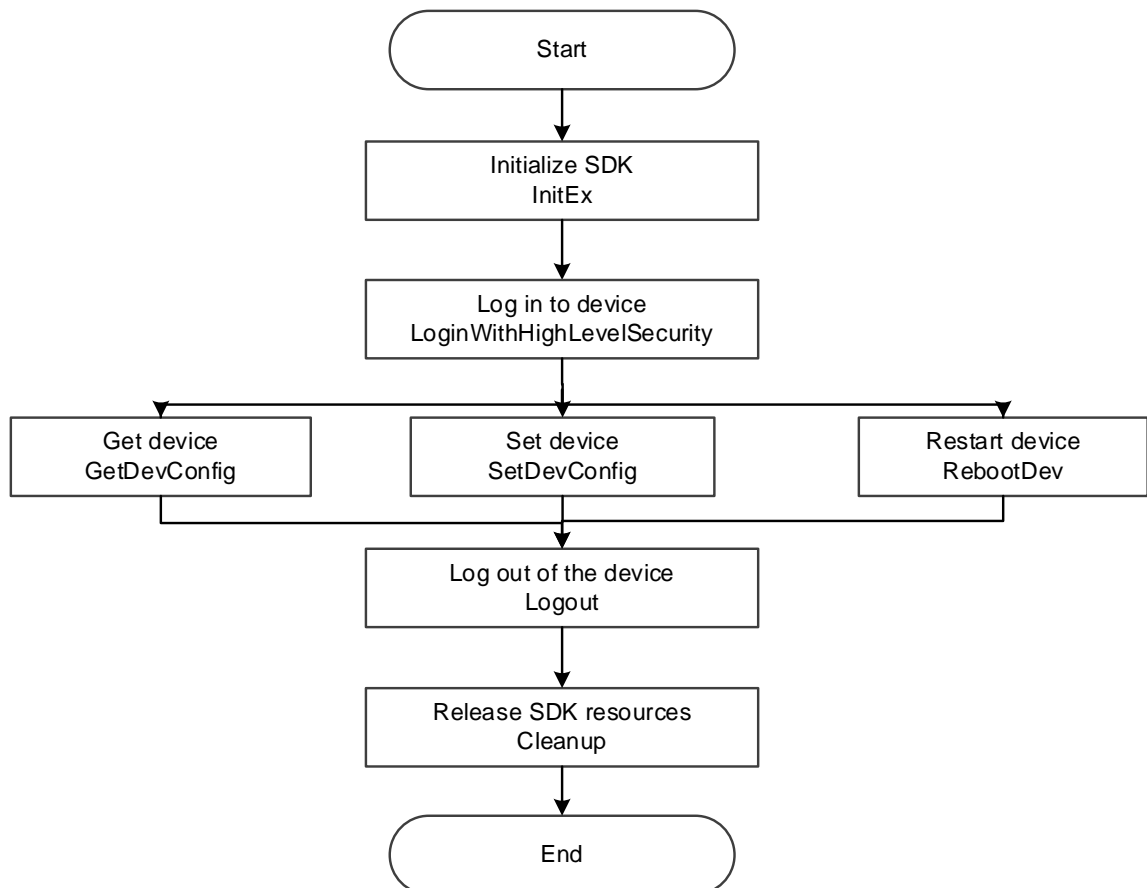
Process Description

2.7.4 Sample Code

```
# get device time
time = NET_TIME()
result = self.sdk.GetDevConfig(self.loginID, int(EM_DEV_CFG_TYPE.TIMECFG), -1, time,
sizeof(NET_TIME))
if not result:
    QMessageBox.critical(self, ' (prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.Ok,
QMessageBox.No)
else:
    get_time = QDateTime(time.dwYear, time.dwMonth, time.dwDay, time.dwHour, time.dwMinute,
time.dwSecond)
    self.Time_dateTimeEdit.setDateTime(get_time)

#configure device time
device_date = self.Time_dateTimeEdit.date()
device_time = self.Time_dateTimeEdit.time()
deviceDateTime = NET_TIME()
deviceDateTime.dwYear = device_date.year()
deviceDateTime.dwMonth = device_date.month()
deviceDateTime.dwDay = device_date.day()
deviceDateTime.dwHour = device_time.hour()
deviceDateTime.dwMinute = device_time.minute()
deviceDateTime.dwSecond = device_time.second()

result = self.sdk.SetDevConfig(self.loginID, int(EM_DEV_CFG_TYPE.TIMECFG), -1, deviceDateTime,
sizeof(NET_TIME))
if not result:
    QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.Ok,
QMessageBox.No)
```

```
# restart the device
result = self.sdk.RebootDev(self.loginID)
if not result:
QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.Ok,
QMessageBox.No)
```

2.8 Remote Snapshot

2.8.1 Introduction

Call NetSDK interface to send snapshot command. Device will capture images from live view and send them to NetSDK, and then NetSDK will return the image data to you.

2.8.2 Interface Overview

Table 2-8 Interfaces of remote snapshot

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
SetSnapRevCallBack	Set remote snapshot callback.
SnapPictureEx	Snapshot extension interface.
Logout	Log out.
GetLastError	Get error codes of interfaces that failed to be called.

2.8.3 Process

Figure 2-9 Process of remote snapshot
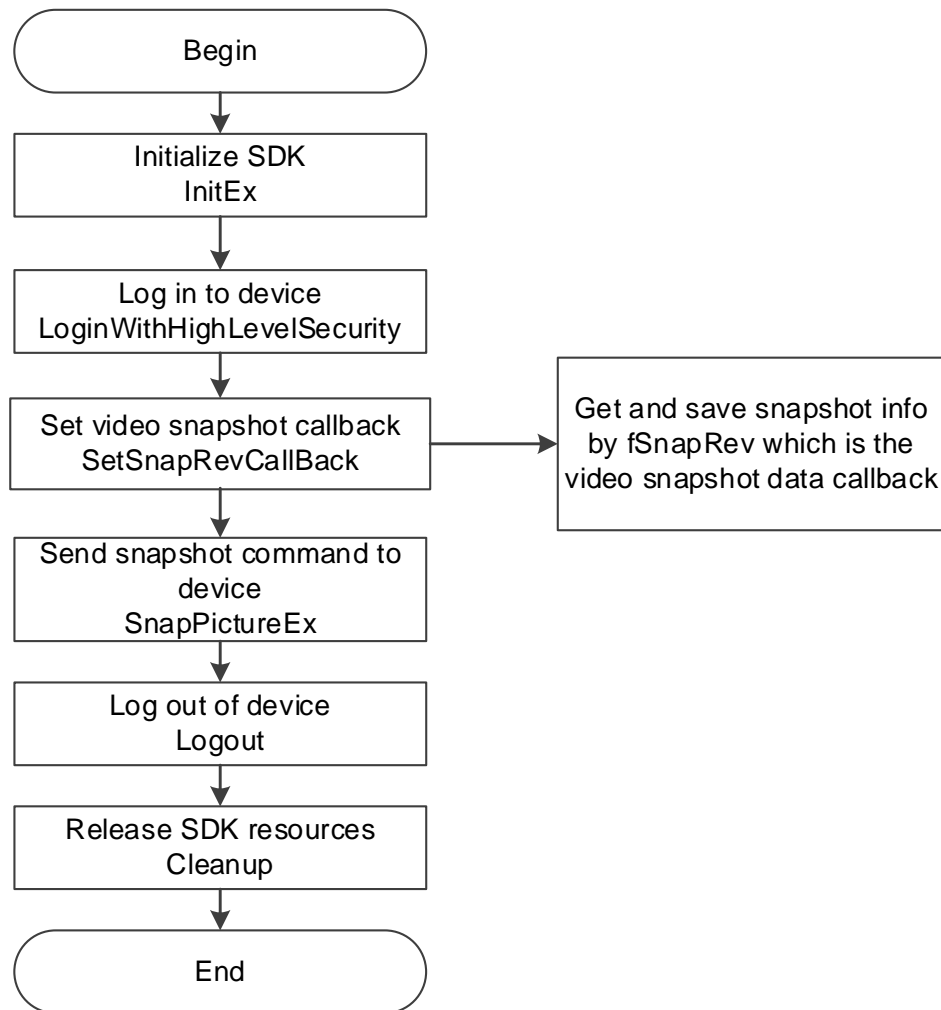
```
              ┌─────────────────┐
              │      Begin       │
              └─────────────────┘
                       │
                       ▼
              ┌─────────────────┐
              │  Initialize SDK  │
              │     InitEx       │
              └─────────────────┘
                       │
                       ▼
              ┌─────────────────┐
              │  Log in to device│
              │LoginWithHighLevelSecurity│
              └─────────────────┘
                       │
                       ▼
          ┌─────────────────────┐       ┌──────────────────────────┐
          │ Set video snapshot   │       │ Get and save snapshot info│
          │ callback             │──────▶│ by fSnapRev which is the  │
          │ SetSnapRevCallBack   │       │ video snapshot data callback│
          └─────────────────────┘       └──────────────────────────┘
                       │
                       ▼
          ┌─────────────────────┐
          │ Send snapshot command│
          │ to device            │
          │ SnapPictureEx        │
          └─────────────────────┘
                       │
                       ▼
          ┌─────────────────────┐
          │ Log out of device    │
          │ Logout               │
          └─────────────────────┘
                       │
                       ▼
          ┌─────────────────────┐
          │ Release SDK resources│
          │ Cleanup              │
          └─────────────────────┘
                       │
                       ▼
              ┌─────────────────┐
              │       End        │
              └─────────────────┘
```

Notes for Process

- Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish to release SDK resources. These two interfaces do not need to be called with every function.
- The time interval for snapshot should be more than 1 second. 3 seconds are recommended.

Process Description

Step 1   Call **InitEx** to initialize SDK.

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

Step 3   Call **SetSnapRevCallBack** to set snapshot callback. When NetSDK receives image data sent from device, NetSDK will call fSnapRev to send image info and image data to you.

Step 4   Call **SnapPictureEx** to send snapshot command. Wait for the returned image info in fSnapRev.

Step 5   Call **Logout** to log out of the device.

Step 6   Call **Cleanup** to release SDK resources.

2.8.4 Sample Code

Code Path

Demo\CapturePicture\CaptureDemo.py

Sample Code

```
def capture_btn_onclick(self):
    # configure snapshot callback
    dwUser = 0
    self.sdk.SetSnapRevCallBack(CaptureCallBack, dwUser)
    channel = self.Channel_comboBox.currentIndex()
    snap = SNAP_PARAMS()
    snap.Channel = channel
    snap.Quality = 1
    snap.mode = 0
    # snapshot
    self.sdk.SnapPictureEx(self.loginID, snap)
```

2.9 Alarm Upload

2.9.1 Introduction

Alarm upload, that is, the device sends an alarm to the platform to inform when the events to be set have occurred. The platform can receive information such as external alarms, video signal loss alarms, privacy masking alarms, and motion detection alarms,

Alarm upload can be realized by NetSDK active login device and subscription of the alarm function to the device, which will send the detected alarm event to NetSDK.

2.9.2 Interface Overview

Table 2-9 Interfaces of alarm upload

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
SetDVRMessCallBackEx1	Set alarm callback.
StartListenEx	Alarm susbscribtion extension interface.
StopListen	Stop alarm susbscribtion.
Logout	Log out.
GetLastError	Get error codes of interfaces that fail to be called.

2.9.3 Process

Figure 2-10 Process of alarm upload
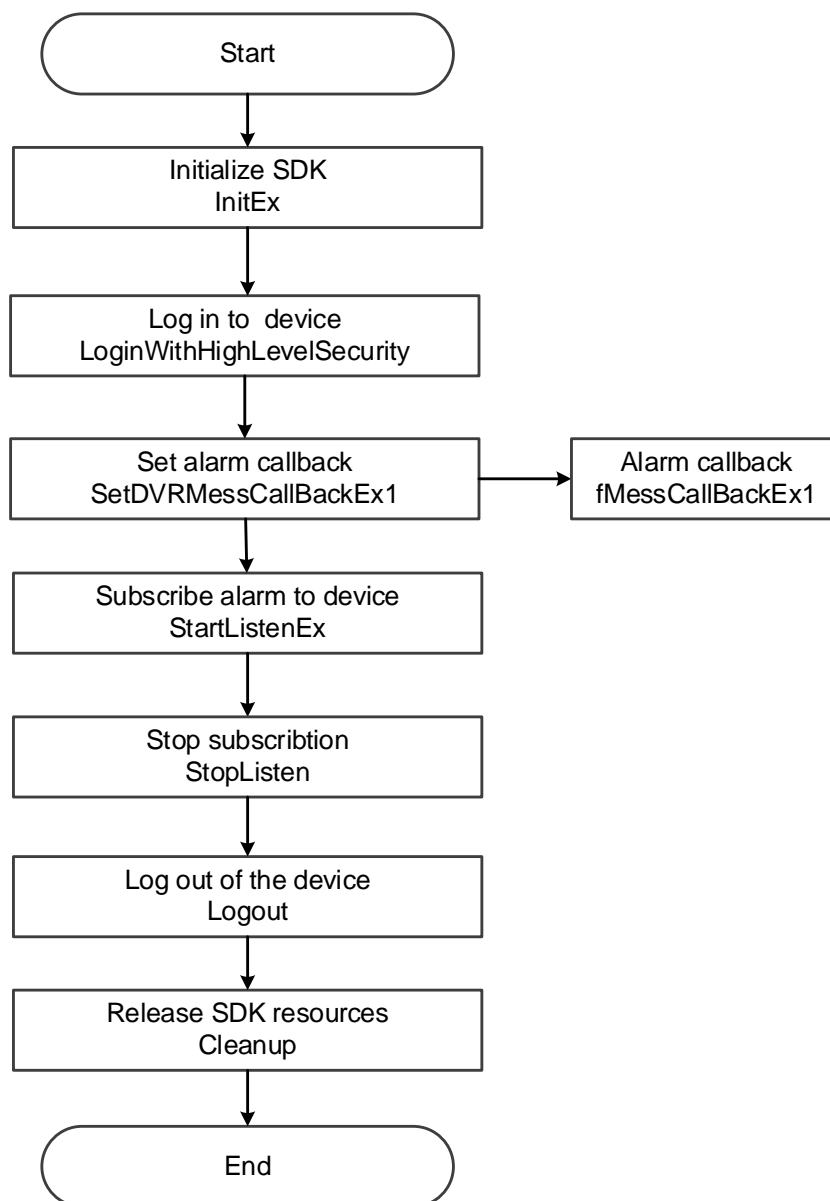
Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called with every function.

Process Description

Step 1   Call **InitEx** to initialize SDK.
Step 2   Call **LoginWithHighLevelSecurity** to log in to the device.
Step 3   Call **SetDVRMessCallBackEx1** to set alarm callback before alarm subscription.

Step 4   Call **StartListenEx** to subscribe to alarm from device. Then the uploaded event will be sent to you by **fMessCallBackEx1**.

Step 5   Call **StopListen** to stop subscribtion.

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

Step 7   Call **Cleanup** to release SDK resources.

2.9.4 Sample Code

Code path

Demo\AlarmListen\ AlarmListenDemo.py

Sample Code

```python
def __init__(self):
    super(StartListenWnd, self).__init__()
    self.setupUi(self)
    # interface initialization
    self.init_ui()

    # NetSDK variables and callbacks used
    self.loginID = C_LLONG()
    self.m_DisConnectCallBack = fDisConnect(self.DisConnectCallBack)
    self.m_ReConnectCallBack = fHaveReConnect(self.ReConnectCallBack)

    #get NetSDK object and initialize it
    self.sdk = NetClient()
    self.sdk.InitEx(self.m_DisConnectCallBack)
    self.sdk.SetAutoReconnect(self.m_ReConnectCallBack)
    #Configure alarm callback function
    self.sdk.SetDVRMessCallBackEx1(MessCallback,0)


def attach_btn_onclick(self):
    self.row = 0
    self.column = 0
    self.Alarmlisten_tableWidget.clear()
    self.Alarmlisten_tableWidget.setHorizontalHeaderLabels(['(No.)',(Time)', '(Channel)', '(Alarm Type)',
'(Status)'])
    result = self.sdk.StartListenEx(self.loginID)
    if result:
        QMessageBox.about(self, '(prompt)', "(Subscribe alarm success)")
        self.Stopalarmlisten_pushButton.setEnabled(True)
        self.Alarmlisten_pushButton.setEnabled(False)
    else:
        QMessageBox.about(self, '(prompt)', 'error:' + str(self.sdk.GetLastError()))
```

```
def detach_btn_onclick(self):
    if (self.loginID > 0):
        self.sdk.StopListen(self.loginID)
    self.Stopalarmlisten_pushButton.setEnabled(False)
    self.Alarmlisten_pushButton.setEnabled(True)
```

2.10 Intelligent Traffic Event Upload

2.10.1 Introduction

Intelligent traffic event upload is the function to analyze real-time stream from intelligent traffic devices. According to the pre-defined rules, SDK will check whether to upload events and carry images.

2.10.2 Interface Overview

Table 2-10 Interfaces of intelligent traffic event upload

Interface	Implication
InitEx	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
RealLoadPictureEx	Intelligent image alarm subscribtion interface.
StopLoadPic	Stop uploading intelligent analysis data-image.
Logout	Log out.
GetLastError	Get error codes of interfaces that fail to be called.

2.10.3 Process

Figure 2-11 Process of intelligent traffic event upload
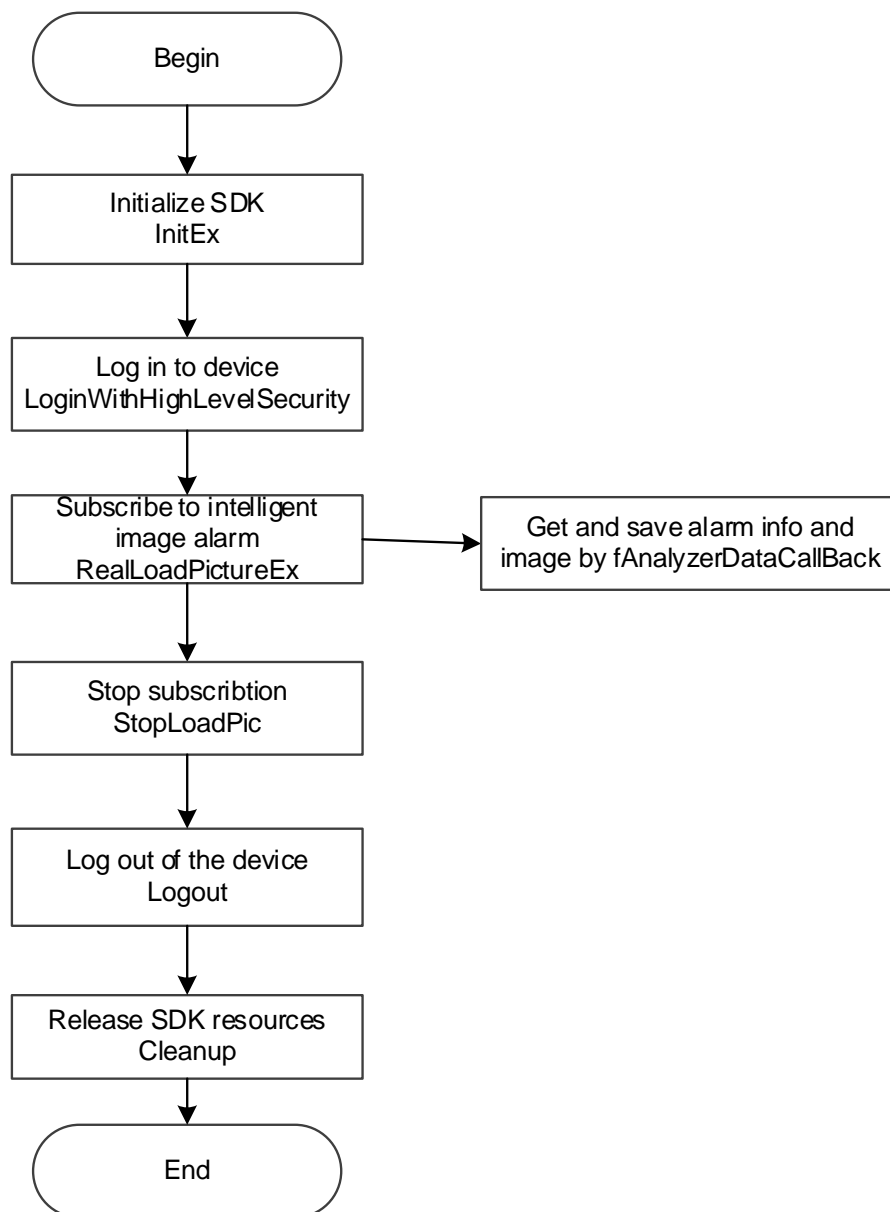
Notes for Process

Call **InitEx** only once before using the SDK during the entire Demo running process. And call **Cleanup** once when all SDK-related functions finish, to release SDK resources. These two interfaces do not need to be called with every function.

Process Description

Step 1　Call **InitEx** to initialize SDK.

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

Step 3   Call **RealLoadpictureEx** to subscribe to alarm from device, and the dwAlarmType should correspond to the enumeration values of EM_EVENT_IVS_TYPE. After the subscription, the uploaded event will be sent to you by callback which is be set in fAnalyzerDataCallBack. The main use of callback is to display and save events.

Step 4   Call **StopLoadPic** to stop subscription of intelligent traffic event.

Step 5   Call **Logout** to log out of the device.

Step 6   Call **Cleanup** to release SDK resources.

2.10.4 Sample Code

2.10.4.1 Intelligent Traffic Junction

Code Path

> \Demo\IntelligentTrafficDemo

Sample Code

```
# Intelligent traffic checkpoint event subscription
def attach_btn_onclick(self):
    self.Attach_tableWidget.setHorizontalHeaderLabels(['(Time)', '(Event)', '(Plate No.)', '(Plate Color)', '
(Vehicle Type)', '(Vehicle Color)'])
    channel = self.Channel_comboBox.currentIndex()
    bNeedPicFile = 1
    dwUser = 0
    self.attachID              =              self.sdk.RealLoadPictureEx(self.loginID,              channel,
EM_EVENT_IVS_TYPE.TRAFFICJUNCTION, bNeedPicFile, AnalyzerDataCallBack, dwUser, None)
    if not self.attachID:
        QMessageBox.about(self, '(prompt)', 'error:' + str(self.sdk.GetLastError()))
    else:
        self.Attach_pushButton.setEnabled(False)
        self.Detach_pushButton.setEnabled(True)
        QMessageBox.about(self, '(prompt)', " (Subscribe success)")


# cancel subscrpption
def detach_btn_onclick(self):
    if (self.attachID == 0):
        return
    self.sdk.StopLoadPic(self.attachID)
    self.attachID = 0
    self.Attach_pushButton.setEnabled(True)
    self.Detach_pushButton.setEnabled(False)
    self.Attach_tableWidget.clear()
    self.row = 0
    self.column = 0
```

```
        self.Attach_tableWidget.viewport().update()
        self.Attach_tableWidget.setHorizontalHeaderLabels([(Time)', '(Event)', '(Plate No.)', '(Plate Color)', '
(Vehicle Type)', '(Vehicle Color)'])
```

2.10.4.2 Target Recognition Event

Code Path

Demo\TargetRecognitionDemo\TargetRecognitionDemo.py

Sample Code

```
def listenevent_btn_onclick(self):
    if not self.realloadID:
        channel = self.Channel_comboBox.currentIndex()
        self.realloadID = self.sdk.RealLoadPictureEx(self.loginID, channel, EM_EVENT_IVS_TYPE.ALL,
True, self.m_AnalyzerDataCallBack)
        if self.realloadID != 0:
            self.ListenEvent_pushButton.setText(" (Detach Listen)")
        else:
            QMessageBox.critical(self,       '       (prompt)',       self.sdk.GetLastErrorMessage(),
QMessageBox.No)
    else:
        result = self.sdk.StopLoadPic(self.realloadID)
        if result:
            self.ListenEvent_pushButton.setText("(Listen Event)")
            self.realloadID = 0
        else:
            QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)


def  AnalyzerDataCallBack(self,  lAnalyzerHandle,  dwAlarmType,  pAlarmInfo,  pBuffer,  dwBufSize,
dwUser, nSequence, reserved):
    if lAnalyzerHandle == self.realloadID:
        if dwAlarmType == EM_EVENT_IVS_TYPE.FACERECOGNITION:
            alarm_info                              =                              cast(pAlarmInfo,
POINTER(DEV_EVENT_FACERECOGNITION_INFO)).contents
            self.show_recognition_info(alarm_info, pBuffer, dwBufSize)
```

2.10.4.3 Target Detection Event

Code Path

Demo\ TargetRecognitionDemo\ TargetRecognitionDemo.py

Sample Code

```
def listenevent_btn_onclick(self):
    if not self.realloadID:
        channel = self.Channel_comboBox.currentIndex()
        self.realloadID = self.sdk.RealLoadPictureEx(self.loginID, channel, EM_EVENT_IVS_TYPE.ALL,
True, self.m_AnalyzerDataCallBack)
        if self.realloadID != 0:
            self.ListenEvent_pushButton.setText(" (Detach Listen)")
        else:
            QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)
    else:
        result = self.sdk.StopLoadPic(self.realloadID)
        if result:
            self.ListenEvent_pushButton.setText("(Listen Event)")
            self.realloadID = 0
        else:
            QMessageBox.critical(self, '(prompt)', self.sdk.GetLastErrorMessage(), QMessageBox.No)


def AnalyzerDataCallBack(self, lAnalyzerHandle, dwAlarmType, pAlarmInfo, pBuffer, dwBufSize,
dwUser, nSequence, reserved):
    if lAnalyzerHandle == self.realloadID:
        if dwAlarmType == EM_EVENT_IVS_TYPE.FACEDETECT:
            alarm_info = cast(pAlarmInfo, POINTER(DEV_EVENT_FACEDETECT_INFO)).contents
            self.show_detect_info(alarm_info, pBuffer, dwBufSize)
```

2.11 Live View Transcoding

2.11.1 Introduction

Live view transcoding involves getting live videos from storage devices or front-end devices and transcoding the videos into the stream type that you need. The supported stream types include:
- GB program stream.
- Transport streams.
- MP4 format.
- H.264 and H.265.
- Program streams.
- RTP streams.

2.11.2 Interface Overview

Table 2-11 Interfaces of live view transcoding

Interface	Implication
Init	Initialize SDK.

Interface	Implication
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
RealPlayByDataType	Start live view transcoding interface.
StopRealPlayEx	Stop live view extension interface.
Logout	Log out.
GetLastError	Get error codes of interfaces that fail to be called.

2.11.3 Process

Figure 2-12 Process of live view transcoding
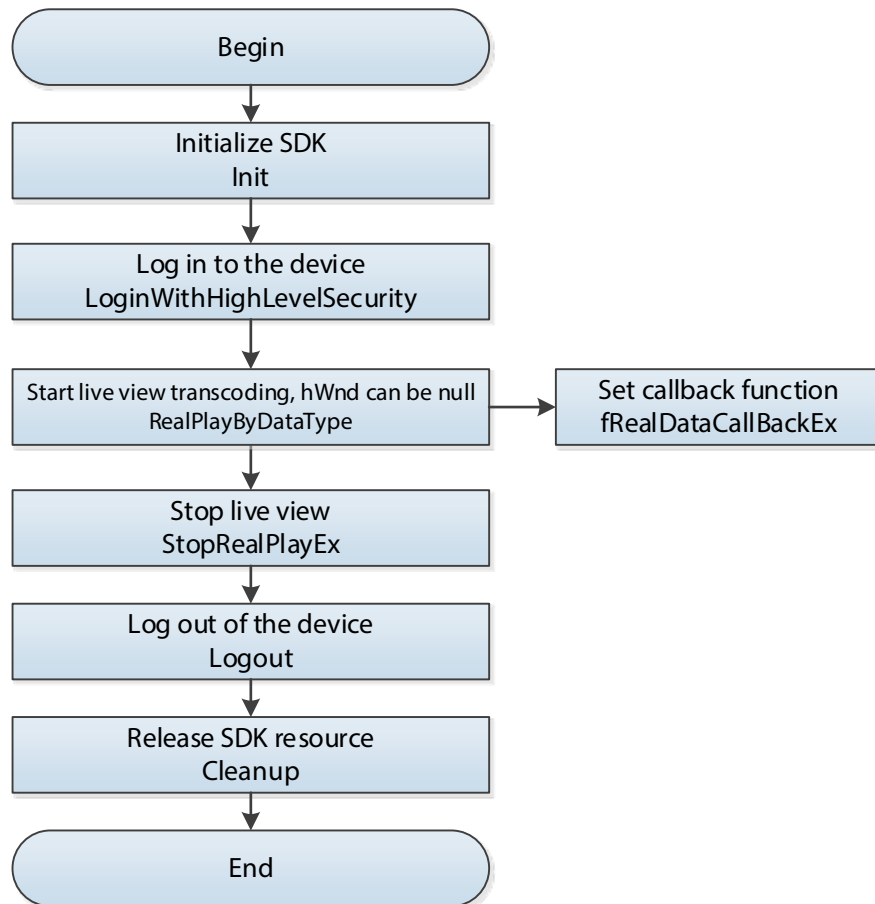
Process Description

Step 1 Initialize SDK.

Step 2 Call L**oginWithHighLevelSecurity** to log in to the device.

Step 3 Call **RealPlayByDataType** to start live view. The parameter hWnd can be set to null.

Step 4 Set the real-time data callback function fRealDataCallBackEx to save the transcoded data.

Step 5 After using the live view transcoding, call **StopRealPlayEx** to stop it.

Step 6 After using the service, call **Logout** to log out of the device.

Step 7 After using all SDK functions, call **Cleanup** to release SDK resources.

2.11.4 Sample Code

```python
# Start live view
def realplay(self):
        print("(Please input live view info)")
        self.channel = int(input('(channel):'))
        self.streamtype = int(input('(stream type(0:Main Stream; 1:Extra Stream)):'))
        if not self.playID:
            if self.streamtype == 0:
                stream_type = SDK_RealPlayType.Realplay
            else:
                stream_type = SDK_RealPlayType.Realplay_1

            inParam = NET_IN_REALPLAY_BY_DATA_TYPE()
            inParam.dwSize = sizeof(NET_IN_REALPLAY_BY_DATA_TYPE)
            inParam.nChannelID = self.channel
            inParam.hWnd = 0
            inParam.rType = stream_type
            inParam.cbRealData = self.m_RealDataCallBack
            inParam.emDataType = EM_REAL_DATA_TYPE.MP4
            inParam.dwUser = 0
            inParam.szSaveFileName = b'realplay.mp4'
            inParam.cbRealDataEx = self.m_RealDataCallBack
            inParam.emAudioType = EM_AUDIO_DATA_TYPE.DEFAULT

            outParam = NET_OUT_REALPLAY_BY_DATA_TYPE()
            outParam.dwSize = sizeof(NET_OUT_REALPLAY_BY_DATA_TYPE)

            self.playID = self.sdk.RealPlayByDataType(self.loginID, inParam, outParam, 5000)
            if self.playID != 0:
                print("(Live view succeed).")
                return True
            else:
                print("(Live view fail). " + self.sdk.GetLastErrorMessage())
                return False
        else:
            print("(Now playing, please close the play window first)")
```

```
# Stop live view
def stop_realplay(self):
        if self.playID:
            self.sdk.StopRealPlayEx(self.playID)
            self.playID = 0
        print("(Stop live view succeed).")
# Stream pulling callback function
    def RealDataCallBack(self, lRealHandle, dwDataType, pBuffer, dwBufSize, param, dwUser):
        pass
```

2.12 Record Playback Transcoding

2.12.1 Introduction

Record playback transcoding refers to remotely playing videos from a specific time period on the client, searching for the needed videos, and transcoding them into the stream type that you need. The supported stream types include:

- GB program stream.
- Transport streams.
- MP4 format.
- H.264 and H.265.
- Program streams.
- RTP streams.

2.12.2 Interface Overview

Table 2-12 Information publish description

Interface	Implication
Init	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
PlayBackByDataType	Start record playback transcoding interface.
StopPlayBack	Stop record playback interface.
Logout	Log out.
GetLastError	Get error codes of interfaces that fail to be called.

2.12.3 Process

Figure 2-13 Process of record playback transcoding
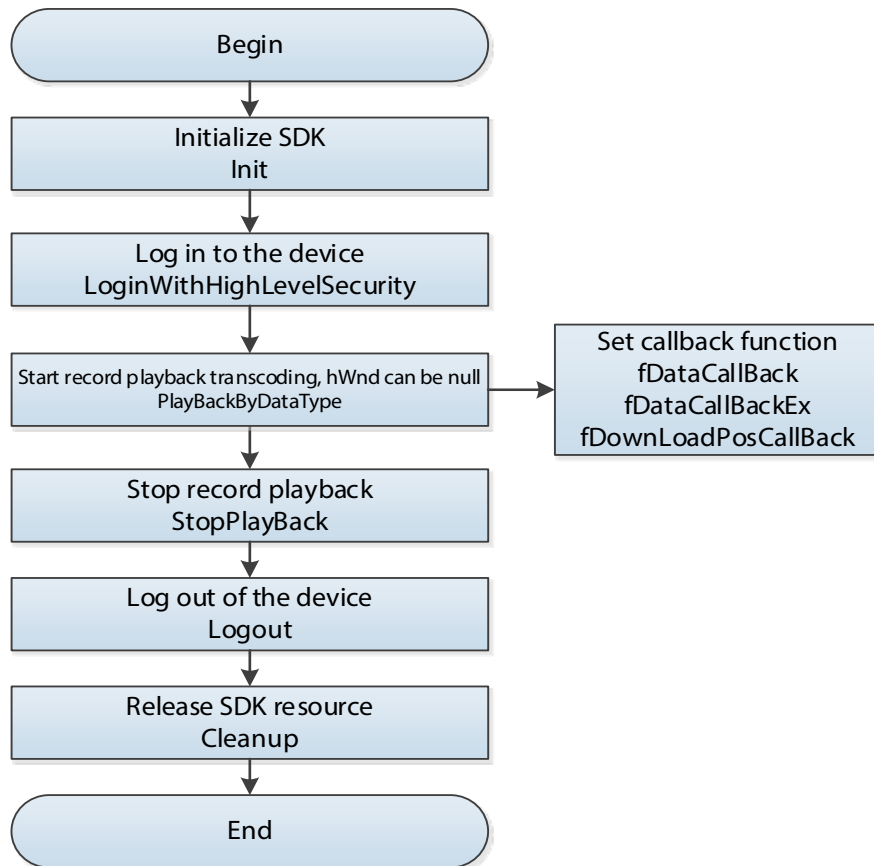
```
┌─────────────────────────────┐
│           Begin             │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│       Initialize SDK        │
│           Init              │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│     Log in to the device    │
│   LoginWithHighLevelSecurity │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐      ┌──────────────────────────┐
│ Start record playback       │─────▶│   Set callback function  │
│ transcoding, hWnd can be null│      │      fDataCallBack       │
│    PlayBackByDataType        │      │     fDataCallBackEx       │
└─────────────────────────────┘      │   fDownLoadPosCallBack    │
              │                       └──────────────────────────┘
              ▼
┌─────────────────────────────┐
│    Stop record playback     │
│        StopPlayBack         │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│    Log out of the device    │
│          Logout             │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│     Release SDK resource    │
│          Cleanup            │
└─────────────────────────────┘
              │
              ▼
┌─────────────────────────────┐
│            End              │
└─────────────────────────────┘
```

Process Description

Step 1 Initialize SDK.

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

Step 3 Call **PlayBackByDataType** to start downloading videos. The parameter hWnd can be set to null.

Step 4 Set the video playback data callback functions fDataCallBackEx and fDataCallBack, and the video playback process callback function fDownLoadPosCallBack to save the transcoded data.

Step 5 After using the record playback transcoding, call **StopPlayBack** to stop it.

Step 6 After using the service, call **Logout** to log out of the device.

Step 7 After using all SDK functions, call **Cleanup** to release SDK resources.

2.12.4 Sample Code

```python
# Start playback
def playback(self):
```

```python
        inParam = NET_IN_PLAYBACK_BY_DATA_TYPE()
        inParam.dwSize = sizeof(NET_IN_PLAYBACK_BY_DATA_TYPE)
        inParam.nChannelID = 3 # channel
        inParam.stStartTime = NET_TIME()
        inParam.stStartTime.dwYear = 2022
        inParam.stStartTime.dwMonth = 1
        inParam.stStartTime.dwDay = 11
        inParam.stStartTime.dwHour = 0
        inParam.stStartTime.dwMinute = 0
        inParam.stStartTime.dwSecond = 0
        inParam.stStopTime = NET_TIME()
        inParam.stStopTime.dwYear = 2022
        inParam.stStopTime.dwMonth = 1
        inParam.stStopTime.dwDay = 12
        inParam.stStopTime.dwHour = 0
        inParam.stStopTime.dwMinute = 0
        inParam.stStopTime.dwSecond = 0
        inParam.hWnd = 0
        inParam.cbDownLoadPos = self.m_DownLoadPosCallBack
        inParam.fDownLoadDataCallBack = self.m_DataCallBack
        inParam.emDataType = 4
        inParam.nPlayDirection = 0
        inParam.emAudioType = 0
        inParam.fDownLoadDataCallBackEx = self.m_DataCallBackEx
        outParam = NET_OUT_PLAYBACK_BY_DATA_TYPE()
        outParam.dwSize = sizeof(NET_OUT_PLAYBACK_BY_DATA_TYPE)
        self.playbackID = self.sdk.PlayBackByDataType(self.loginID, inParam, outParam, 5000)
        if self.playbackID != 0:
            print("(PlayBack succeed).")
            return True
        else:
            print("(PlayBack fail). " + self.sdk.GetLastErrorMessage())
            return False


# Callback function
def DownLoadPosCallBack(self, lPlayHandle, dwTotalSize, dwDownLoadSize, dwUser):
        pass


def DataCallBack(self, lRealHandle, dwDataType, pBuffer, dwBufSize, dwUser):
```

```
        return 0
def DataCallBackEx(self, lRealHandle, pDataCallBack, dwUser):
        pass
```

2.13 Record Download Transcoding

2.13.1 Introduction

The record download function helps you obtain the records saved on the storage device through SDK and save into the local. It allows you to download records of different stream types from the selected channels and export to the local disk or external USB flash drive. The supported stream types include:
- GB program stream.
- Transport streams.
- MP4 format.
- H.264 and H.265.
- Program streams.
- RTP streams.

2.13.2 Interface Overview

Table 2-13 Interfaces of record download transcoding

Interface	Implication
Init	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
DownloadByDataType	Start record download transcoding interface.
StopDownload	Stop record download interface.
Logout	Log out.
GetLastError	Get error codes of interfaces that fail to be called.

2.13.3 Process

Figure 2-14 Process of record download transcoding
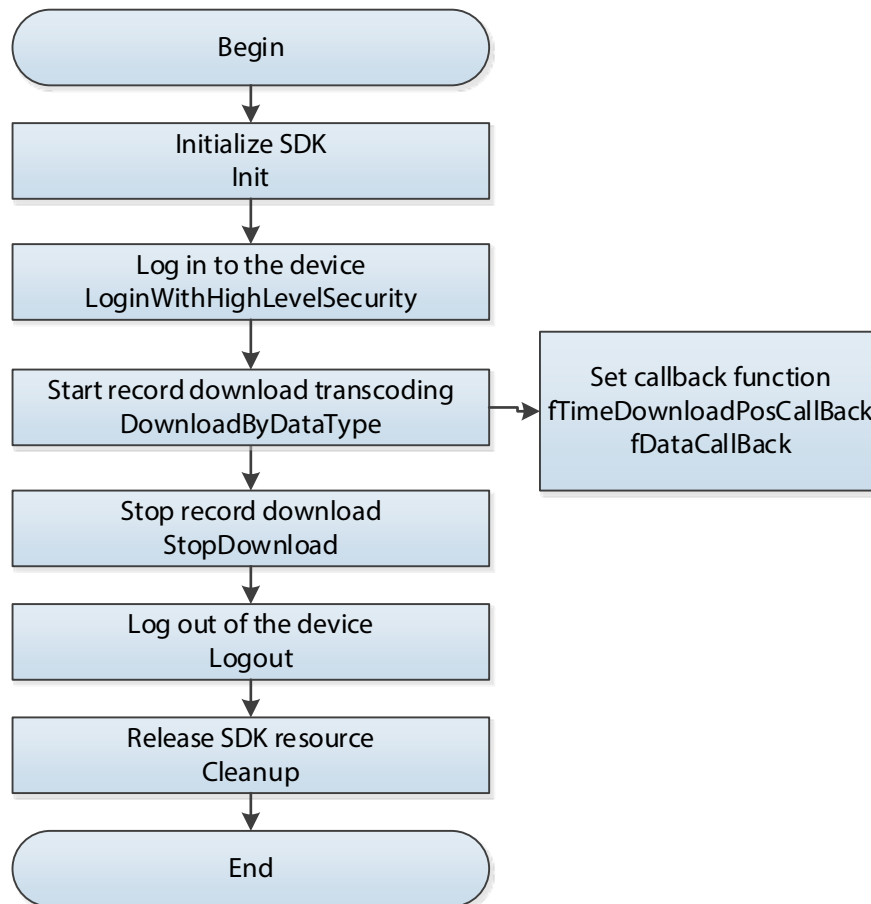
```
                    ┌─────────────────────────┐
                    │          Begin          │
                    └─────────────────────────┘
                                  │
                    ┌─────────────────────────┐
                    │     Initialize SDK      │
                    │          Init           │
                    └─────────────────────────┘
                                  │
                    ┌─────────────────────────┐
                    │    Log in to the device │
                    │ LoginWithHighLevelSecurity│
                    └─────────────────────────┘
                                  │
                    ┌─────────────────────────┐    ┌──────────────────────────┐
                    │ Start record download   │    │   Set callback function  │
                    │ transcoding             │───▶│ fTimeDownloadPosCallBack │
                    │ DownloadByDataType      │    │      fDataCallBack        │
                    └─────────────────────────┘    └──────────────────────────┘
                                  │
                    ┌─────────────────────────┐
                    │   Stop record download  │
                    │      StopDownload       │
                    └─────────────────────────┘
                                  │
                    ┌─────────────────────────┐
                    │    Log out of the device│
                    │         Logout          │
                    └─────────────────────────┘
                                  │
                    ┌─────────────────────────┐
                    │   Release SDK resource  │
                    │         Cleanup         │
                    └─────────────────────────┘
                                  │
                    ┌─────────────────────────┐
                    │           End           │
                    └─────────────────────────┘
```

Process Description

Step 1   Initialize SDK.

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

Step 3   Call **DownloadByDataType** to start record download transcoding. The parameter hWnd can be set to null.

Step 4   Set the video download process callback function fTimeDownloadPosCallBack, and the video download data callback function fDataCallBackEx to save the transcoded data.

Step 5   After using the record download transcoding, call **StopDownload** to stop it.

Step 6   After using the service, call **Logout** to log out of the device.

Step 7   After using all SDK functions, call **Cleanup** to release SDK resources.

2.13.4 Sample Code

```
# Download videos
def download(self):
```

```python
        inParam = NET_IN_DOWNLOAD_BY_DATA_TYPE()
        inParam.dwSize = sizeof(NET_IN_DOWNLOAD_BY_DATA_TYPE)
        inParam.nChannelID = 3    # channel
        inParam.emRecordType = 0
        inParam.szSavedFileName = b'download.mp4'
        inParam.stStartTime = NET_TIME()
        inParam.stStartTime.dwYear = 2022
        inParam.stStartTime.dwMonth = 1
        inParam.stStartTime.dwDay = 11
        inParam.stStartTime.dwHour = 0
        inParam.stStartTime.dwMinute = 0
        inParam.stStartTime.dwSecond = 0
        inParam.stStopTime = NET_TIME()
        inParam.stStopTime.dwYear = 2022
        inParam.stStopTime.dwMonth = 1
        inParam.stStopTime.dwDay = 12
        inParam.stStopTime.dwHour = 0
        inParam.stStopTime.dwMinute = 0
        inParam.stStopTime.dwSecond = 0
        inParam.cbDownLoadPos = self.m_TimeDownLoadPosCallBack
        inParam.fDownLoadDataCallBack = self.m_DataCallBack
        inParam.emDataType = 4
        inParam.emAudioType = 0
        outParam = NET_OUT_DOWNLOAD_BY_DATA_TYPE()
        outParam.dwSize = sizeof(NET_OUT_DOWNLOAD_BY_DATA_TYPE)
        self.downloadID = self.sdk.DownloadByDataType(self.loginID, inParam, outParam, 5000)
        if self.downloadID != 0:
            print("(download succeed).")
            return True
        else:
            print("(download fail). " + self.sdk.GetLastErrorMessage())
            return False


# Callback function
def DataCallBack(self, lRealHandle, dwDataType, pBuffer, dwBufSize, dwUser):
        return 0


def TimeDownLoadPosCallBack(self, lPlayHandle, dwTotalSize, dwDownLoadSize, index,
recordfileinfo, dwUser):
```

2.14 PTZ Control

2.14.1 Introduction

PTZ control is an important part of the surveillance system. In different scenarios, users have different needs for live view. For example, in a regular scenario, users want to track the target in the image. Through SDK, you can control PTZ devices and perform operations such as moving up, down, left, and right, focusing, zooming in and out, tours, and 3-dimensional positioning.
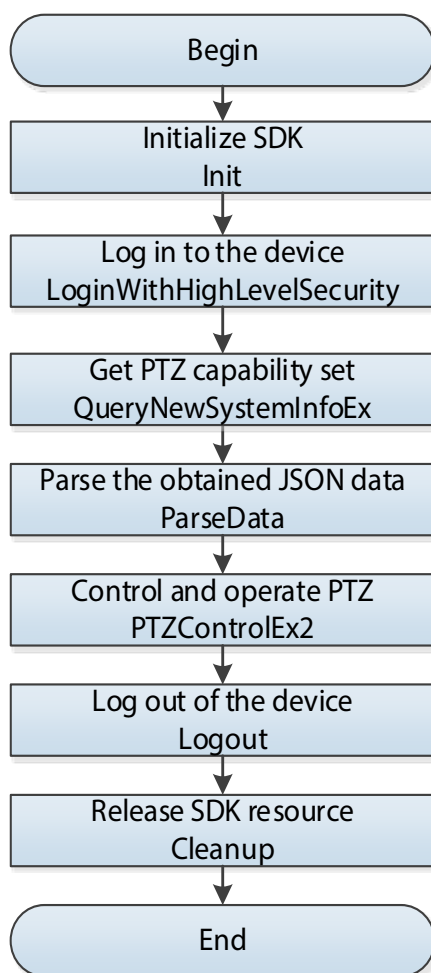
2.14.2 Interface Overview

Table 2-14 Interfaces of PTZ control

Interface	Implication
Init	Initialize SDK.
Cleanup	Clean up SDK.
LoginWithHighLevelSecurity	Log in with high level security.
ParseData	Parse the searched configuration information.
PTZControlEx2	Extension interface for PTZ control.
QueryNewSystemInfoEx	Query the capability of the new system.
Logout	Log out.
GetLastError	Get error codes of interfaces that fail to be called.

2.14.3 Process

Figure 2-15 Process of PTZ control

```
                            ┌─────────────────────┐
                            │        Begin        │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │   Initialize SDK    │
                            │        Init         │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │  Log in to the device │
                            │ LoginWithHighLevelSecurity │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │ Get PTZ capability set │
                            │  QueryNewSystemInfoEx │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │ Parse the obtained JSON data │
                            │      ParseData      │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │ Control and operate PTZ │
                            │    PTZControlEx2    │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │ Log out of the device │
                            │       Logout        │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │ Release SDK resource │
                            │      Cleanup        │
                            └─────────────────────┘
                                      │
                                      ▼
                            ┌─────────────────────┐
                            │         End         │
                            └─────────────────────┘
```

Process Description

Step 1   Initialize SDK.

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

Step 3   Call **QueryNewSystemInfoEx** and use the CFG_CAP_CMD_TYPE. PTZ command to get the capability set of the PTZ. Call **ParseData** and use the CFG_CAP_CMD_TYPE. PTZ command to parse the obtained capability set.

Step 4   According to your needs, call **PTZControlEx2** to operate the PTZ. Different PTZ commands might require different parameters. For certain operation commands, you might need to call the corresponding stop command, such as left or right movement. Refer to the sample code for more details.

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

Step 6   After using all SDK functions, call **Cleanup** to release SDK resources.

2.14.4 Sample Code

```
def ptz_control(self):
    channel = 3
    # Confirm the preset number first
```

```python
        # Preset position control
        param2 = 9    # preset position number
        print("Step 1: set preset position")
        result = self.sdk.PTZControlEx2(self.loginID, channel,
SDK_PTZ_ControlType.POINT_SET_CONTROL, 0, param2,
                                        0, False, None)
        if result:
            print("set preset position succeed.")
        else:
            print("set preset position fail. " + self.sdk.GetLastErrorMessage())
            return False
```

3 Interface Definition

3.1 SDK Initialization

3.1.1 InitEx

Table 3-1 Initialize SDK

Item	Description	
Name	Initialize SDK.	
Function	def InitEx(cls, call_back: fDisConnect = None, user_data: C_LDWORD = 0, init_param: NETSDK_INIT_PARAM = NETSDK_INIT_PARAM()) -> int	
Parameter	[in] call_back	Disconnection callback.
	[in] user_data	User parameter of disconnection callback.
	[in] init_param	Initialzie parameters.
Return value	● Success: 1. ● Failure: 0.	
Note	It is the precondition for calling other function modules. If the callback is set as None, the callback will not be sent to the user after the device is disconnected. The parameter user_data passed in by InitEx will be returned in the same field user_data of fDisConnect. User_data is not processed inside NetSDK, and is only used to carry user data into the callback.	

3.1.2 Cleanup

Table 3-2 Clean up SDK

Item	Description
Name	Clean up SDK.
Function	def Cleanup(cls)
Parameter	None.
Return value	None.
Note	Call the SDK cleanup interface before the process ends.

3.1.3 SetAutoReconnect

Table 3-3 Set reconnection callback

Item	Description
Name	Set auto reconnection callback.

Item	Description	
Function	def SetAutoReconnect(cls, call_back: fHaveReConnect, user_data: C_LDWORD = None)	
Parameter	[in] call_back	Reconnection callback.
	[in] user_data	User parameter of disconnection callback.
Return value	None.	
Note	Set the reconnection callback interface. If the callback is set as None, it will not connect automatically.	

3.2 Device Search and Device Initialization

3.2.1 StartSearchDevicesEx

Table 3-4 Async device search

Item	Description	
Name	Async device search.	
Function	def StartSearchDevicesEx(cls, pInBuf: NET_IN_STARTSERACH_DEVICE, pOutBuf: NET_OUT_STARTSERACH_DEVICE) -> C_LLONG	
Parameter	[in] pInBuf	Async device searching input structure.
	[out] pOutBuf	Async device searching output structure.
Return value	● Success: Search handle. ● Failure: 0. ● Call GetLastError to get error codes.	
Note	Only support searching for devices within the same network segment. The number of calls to the search interface is the same as the number of network cards. After the device searching is successful, bind the search handle to the IP. After the callback search result is returned, find the corresponding local IP through the search handle, and pass in the local IP when initializing the device account.	

3.2.2 SearchDevicesByIPs

Table 3-5 Search for device in cross-segment

Item	Description
Name	Search for device IP in cross-segemt.
Function	def SearchDevicesByIPs(cls, pIpSearchInfo: DEVICE_IP_SEARCH_INFO, cbSearchDevices: fSearchDevicesCB, dwUserData: C_LDWORD, szLocalIp: c_char_p = None,

Item	Description	
	dwWaitTime: C_DWORD = 5000) -> c_int:	
Parameter	[in] pIpSearchInfo	Search device info.
	[in] cbSearchDevices	Search device callback. When a device response packet returns, NetSDK parses the response packet into valid information and notifies users by the callback. For details, see the description of fSearchDevicesCB. Callback cannot be null.
	[in] dwUserData	User data. NetSDK returns the data to users by fSearchDevicesCB whichis the device search callback.
	[in] szLocalIp	Local IP. The default value is None. And no value enrtered is allowed.
	[in] dwWaitTime	Search time expected by users. Se the parameters as nedded. This interfacre is a synchronous interface, so it only returns when the waiting time of search is finished.
Return value	● Success: 1. ● Failure: 0.	
Note	This interfacre is a synchronous interface, so it only returns when the waiting time of search is finished. Enter the search time according to own network situations.	

3.2.3 StopSearchDevices

Table 3-6 Stop async searching

Item	Description	
Name	Stop searching for devices with the same network segment, such as IPC and NVS.	
Function	def StopSearchDevices(cls, lSearchHandle: C_LLONG) -> c_int	
Parameter	[in] lSearchHandle	Async search for device ID. Return value of async search interfaces, such as StartSearchDevicesEx.
Return value	● Success: 1. ● Failure: 0.	
Note	Use with StartSearchDevicesEx in pairs.	

3.2.4 InitDevAccount

Table 3-7 Initialize device

Item	Description
Name	Initialize device account.

Item	Description	
Function	def InitDevAccount(cls, pInitAccountIn: NET_IN_INIT_DEVICE_ACCOUNT, pInitAccountOut: NET_OUT_INIT_DEVICE_ACCOUNT, dwWaitTime: int = 5000, szLocalIp: c_char_p = None) -> c_int	
Parameter	[in] pInitAccountIn	Input structure of decive initialization.
	[out]pInitAccountOut	Output structure of decive initialization.
	[in] dwWaitTime	Waiting time. The unit is ms.
	[in] szLocalIp	Local IP. Should be the same with szLocalIp filed of pInBuf of StartSearchDevicesEx.
Return value	● Success: 1. ● Failure: 0.	
Note	If the PC has several network cards, you need to call StartSearchDevicesEx for several times. After the search is successful, the search handle is bound to the IP. When searching for callback information, find the corresponding local IP by the search handle. During initialization, szLocalIp should be the local IP.	

3.3 Device Login

3.3.1 LoginWithHighLevelSecurity

Table 3-8 Log in

Item	Description	
Name	Log in to the device.	
Function	def LoginWithHighLevelSecurity(cls, stuInParam: NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY, stuOutParam: NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY) -> tuple	
Parameter	[in] stuInParam	Input parameter structure.
	[out] stuOutParam	Output parameter structure.
	[out] device_info	Device info.
	[out] error_message	Error info of login interfece.
Return value	● Success: Non-0. ● Failure: 0.	
Note	None.	

3.3.2 Logout

Table 3-9 Log out

Item	Description
Name	Log out of the device.

Item	Description	
Function	def Logout(cls, login_id: int) -> int	
Parameter	[in]login_id	Return value of LoginWithHighLevelSecurity.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.4 Live View

3.4.1 RealPlayEx

Table 3-10 Start preview live ivew

Item	Description	
Name	Start live view.	
Function	def RealPlayEx(cls, login_id: int, channel: int, hwnd: int, play_type=SDK_RealPlayType.Realplay) -> C_LLONG	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] channel	Video channel No. is a round No., starting from 0.
	[in] hwnd	Window handle valid only under Windows system.
	[in] play_type	Live type.
Return value	● Success: Non-0. ● Failure: 0	
Note	Windows system: ● When hWnd is valid, the corresponding window displays picture. ● When hWnd is None, get the video data through setting a callback and send to user for handle.	

Table 3-11 Live type and meaning

Live type	Meaning
Realplay	Real-time live
Multiplay	Multi-picture live
Realplay_0	Live view—main stream, equivalent to Realplay
Realplay_1	Live view—sub stream 1
Realplay_2	Live view—sub stream 2
Realplay_3	Live view—sub stream 3
Multiplay_1	Multi-picture live—1 picture
Multiplay_4	Multi-picture live—4 pictures
Multiplay_8	Multi-picture live—8 pictures

Live type	Meaning
Multiplay_9	Multi-picture live—9 pictures
Multiplay_16	Multi-picture live—16 pictures
Multiplay_6	Multi-picture live—6 pictures
Multiplay_12	Multi-picture live—12 pictures
Multiplay_25	Multi-picture live—25 pictures
Multiplay_36	Multi-picture live—36 pictures

3.4.2 StopRealPlayEx

Table 3-12 Stop live view

Item	Description	
Name	Stop the live view.	
Function	def StopRealPlayEx(cls, realplay_id: int) -> int	
Parameter	[in] realplay_id	Return value of RealPlayEx.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.5 Record Playback

3.5.1 SetDeviceMode

Table 3-13 Set working mode

Item	Description	
Name	Set working mode.	
Function	def SetDeviceMode(cls, login_id: int, emType: int, value: c_void_p) -> c_int	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] emType	Working mode enumeration.
	[in] value	Structure correspondes to working mode.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

Table 3-14 Working mode and corresponding structure

emType Enumeration	Meaning	Structure
RECORD_STREAM_TYPE	Set the record stream type to query and playback by time. ● 0: Main and sub stream	None

emType Enumeration	Meaning	Structure
	● 1: Main stream ● 2: Sub stream	
RECORD_TYPE	Set the record file type to playback and download by time.	EM_RECORD_TYPE

3.5.2 QueryRecordFile

Table 3-15 Query for all the record files within a period

Item	Description	
Name	Query for all the record files within a period.	
Function	def QueryRecordFile(cls, login_id: int, channel_id: int, recordfile_type: int, start_time: NET_TIME, end_time: NET_TIME, card_id: str, wait_time:int, is_querybytime:bool) -> tuple	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] channel_id	Device channel.
	[in] recordfile_type	Query type. Refer to EM_QUERY_RECORD_TYPE.
	[in] start_time	Start time.
	[in] end_time	End time.
	[in] card_id	Card ID.
	[in] wait_time	Waiting time.
	[in] is_querybytime	Whether to query by time.
	[out] file_count	Returned file number.
	[out] recordfile_infos	File info of returned records. The strcture group of NET_RECORDFILE_INFO.
Return value	● Success: 1. ● Failure: 0.	
Note	Before playback, call this interface to query the video records. When the info of searched record within the entered time is greater than the defined buffer size, SDK only returns the records that can be stored in the buffer. You can continue to query as needed.	

3.5.3 PlayBackByTimeEx2

Table 3-16 Playback by time

Item	Description
Name	Playback by time.

Item	Description	
Function	def PlayBackByTimeEx2(cls, login_id: int, channel_id: int, in_param: NET_IN_PLAY_BACK_BY_TIME_INFO, out_param: NET_OUT_PLAY_BACK_BY_TIME_INFO) -> int:	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] channel_id	Device channel No..
	[in] in_param	Query input condition.
	[out] out_param	Query output information.
Return value	● Success: Non-0. ● Failure: 0.	
Note	For the callback declaration cbDownLoadPos and fDownLoadDataCallBack in NET_IN_PLAY_BACK_BY_TIME_INFO, see "3.11 错误!表格结果无效。" The parameters hWnd and fDownLoadDataCallBack in pstNetIn cannot be None at the same time; otherwise, the interface calling will be failed returned.	

3.5.4 StopPlayBack

Table 3-17 Stop playback

Item	Description	
Name	Stop playback.	
Function	def StopPlayBack(cls, playback_id: int) -> int	
Parameter	[in] playback_id	Playback handle. Return value of PlayBackByTimeEx2.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.5.5 PausePlayBack

Table 3-18 Pause or resume playback

Item	Description	
Name	Pause or resume playback.	
Function	def PausePlayBack(cls, playback_id: int, is_pause: bool) -> int:	
Parameter	[in] playback_id	Playback handle. Return value of PlayBackByTimeEx2.
	[in] is_pause	Pause or resume. True: pause; False: resume.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.6 Record Download

3.6.1 DownloadByTimeEx

Table 3-19 Download by time

Item	Description	
Name	Download by time.	
Function	def DownloadByTimeEx(cls, login_id: int, channel_id: int, recordfile_type: int, start_time: NET_TIME, end_time: NET_TIME, save_filename: str, callback_timedownloadpos: fTimeDownLoadPosCallBack, time_UserData: C_LDWORD, callback_timedownloaddata: fDataCallBack, data_UserData: C_LDWORD, pReserved: int = 0) -> int	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] channel_id	Device channel No., starting from 0.
	[in] recordfile_type	Record file type.
	[in] start_time	Start time.
	[in] end_time	End time.
	[in] save_filename	Record file name to be save. Full path.
	[in] callback_timedownloadpos	Download progress callback.
	[in] time_UserData	Customized data of download progress callback.
	[in] callback_timedownloaddata	Download data callback.
	[in] data_UserData	Customized data of download data callback.
	[in] pReserved	Reserved parameter.
Return value	● Success: Non-0. ● Failure: 0.	
Note	For callback declaration of callback_timedownloadpos and callback_timedownloaddata, see "3.11 错误!表格结果无效。" sSavedFileName is not blank, and the record data is input into the file corresponding with the path. fDownLoadDataCallBack is not blank, and the record data is returned through callback.	

3.6.2 StopDownload

Table 3-20 Stop record download

Item	Description	
Name	Stop record download.	
Function	def StopDownload(cls, download_id: int) -> int	
Parameter	[in] download_id	Return value of DownloadByTimeEx.
Return value	● Success: 1. ● Failure: 0.	
Note	Stop downloading after it is completed or partially completed according to particular situation.	

3.7 Device Control

3.7.1 GetDevConfig

Table 3-21 Get device configuration info

Item	Description	
Name	Get device configuration info.	
Function	def GetDevConfig(cls, login_id: C_LLONG, cfg_type: C_DWORD, channel_id: C_LONG, out_buffer: C_LLONG, outbuffer_size: C_DWORD, wait_time: int = 5000) -> int	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] cfg_type	Query time. For details, see the EM_DEV_CFG_TYPE enumeration in the SDK_Enum.py file.
	[in] channel_id	Quey channel No..
	[out] out_buffer	Obtained strcture data.
	[in] outbuffer_size	Data length of out_buffer.
	[in] wait_time	Timeout.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

Table 3-22 Configuration type enumeration

emType Enumeration	Description
TIMECFG	Time configuration. GetDevConfig and SetDevConfig are used together.

3.7.2 SetDevConfig

Table 3-23 Set device configuration info

Item	Description	
Name	Set device configuration info.	
Function	def SetDevConfig(cls, login_id: C_LLONG, cfg_type: C_DWORD, channel_id: C_LONG, in_buffer: C_LLONG, inbuffer_size: C_DWORD, wait_time: int = 5000) -> int	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
	[in] cfg_type	Query type. For details, see the EM_DEV_CFG_TYPE enumeration in the SDK_Enum.py file.
	[in] channel_id	Quey channel No..
	[in] in_buffer	Imported strcture data.
	[in] inbuffer_size	Data length of in_buffer.
	[in] wait_time	Timeout.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.7.3 RebootDev

Table 3-24 Restart device

Item	Description	
Name	Restart device.	
Function	def RebootDev(cls, login_id: int) -> int:	
Parameter	[in] login_id	Return value of LoginWithHighLevelSecurity.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.8 Remote Snapshot

3.8.1 SetSnapRevCallBack

Table 3-25 Set remote snapshot callback

Item	Description	
Name	Set snapshot callback.	
Function	def SetSnapRevCallBack(cls, OnSnapRevMessage: fSnapRev, dwUser: C_LDWORD) -> None	
Parameter	[in] OnSnapRevMessage	Remote snapshot callback.
	[in] dwUser	User data. SDK will return data to users by fSnapRev.
Return value	● None.	
Note	● Call SetSnapRevCallBack before calling SnapPictureEx.	

3.8.2 SnapPictureEx

Table 3-26 Snapshot command intension interface

Item	Description	
Name	Snapshot command intension interface.	
Function	def SnapPictureEx(cls, lLoginID:C_LLONG, par:SNAP_PARAMS, reserved=0)->c_int	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
	[in] par	Snapshot parameters. For detalis, see SNAP_PARAMS structure.
	[in] reserved	Picture format.
Return value	● Success: 1. ● Failure: 0.	
Note	● None.	

3.9 Alarm Upload

3.9.1 SetDVRMessCallBackEx1

Table 3-27 Set alarm callback

Item	Description	
Name	Set alarm callback.	
Function	def SetDVRMessCallBackEx1(cls, cbMessage:fMessCallBackEx1, dwUser:C_LDWORD)->None	
Parameter	[in] cbMessage	Alarm callback. For details, see fMessCallBackEx1.

Item	Description	
	[in] dwUser	User data. SDK will return data to users by fMessCallBackEx1.
Return value	● None.	
Note	Call StartListenEx before calling SetDVRMessCallBackEx1.	

3.9.2 StartListenEx

Table 3-28 Start alarm subscription

Item	Description	
Name	Extension interface of device alarm subscribtion.	
Function	def StartListenEx(cls, lLoginID:C_LLONG)->c_int	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
Return value	● Success: 1. ● Failure: 0.	
Note	All alarm evnets of devices are fed back by callback set in SetDVRMessCallBackEx1	

3.9.3 StopListen

Table 3-29 Stop alarm subscription

Item	Description	
Name	Stop alarm subscribtion.	
Function	def StopListen(cls, lLoginID:C_LLONG)->c_int	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.10 Intelligent Traffic Event Upload

3.10.1 RealLoadPictureEx

Table 3-30 Intelligent image alarm subscription

Item	Description
Name	Intelligent image alarm subscription.

Item	Description	
Function	def RealLoadPictureEx(cls, lLoginID: C_LLONG, nChannelID: c_int, dwAlarmType: c_ulong, bNeedPicFile: c_int, cbAnalyzerData: fAnalyzerDataCallBack, dwUser: C_LDWORD = 0, reserved: c_void_p = None) -> C_LLONG	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
	[in] nChannelID	Channel No. of intelligent image alarm subscribtion, starting from 0.
	[in] dwAlarmType	Alarm event type expected to subscribe. Refer to EM_EVENT_IVS_TYPE.
	[in] bNeedPicFile	Subscribe to image file or not? ● 1: Subscribe to image. ● 0: Not subscribe to image.
	[in]cbAnalyzerData	Callback of intelligent event. When there is intelligent image alarm be uploaded, NetSDK will returns data to users.
	[in] dwUser	User data. SDK will return data to users by fAnalyzerDataCallBack.
	[in] reserved	Reserved parameter.
Return value	● Success: ID of Intelligent image alarm subscription. ● Failure: 0, and it will be the parameter of StopLoadPic.	
Note	If you need to subscribe to several events on one channel, set the evnt type as EM_EVENT_IVS_ALL to subscribe to all event types when calling RealLoadPictureEx, and then process the evnets you need.	

3.10.2 StopLoadPic

Table 3-31 Stop subscription of intelligent event

Item	Description	
Name	Stop subscribtion of intelligent event.	
Function	def StopLoadPic(cls, lAnalyzerHandle:C_LLONG)->c_int	
Parameter	[in] lAnalyzerHandle	Return value of RealLoadPictureEx.
Return value	● Success: 1. ● Failure: 0.	
Note	None.	

3.11 Enabling Live View Transcoding Interface

Table 3-32 RealPlayByDataType

Item	Description	
Name	Enable the live view transcoding interface.	
Function	def RealPlayByDataType(cls, lLoginID: int, pstInParam: NET_IN_REALPLAY_BY_DATA_TYPE, pstOutParam: NET_OUT_REALPLAY_BY_DATA_TYPE, dwWaitTime: int = 5000) -> C_LLONG	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
	[in] pstInParam	Input parameter structure.
	[out] pstOutParam	Output parameter structure.
	[in] dwWaitTime	Waiting time.
Return value	Success: Non-0. Failure: 0	
Description	None.	

3.12 Enabling Record Playback Transcoding Interface

Table 3-33 PlayBackByDataType

Item	Description	
Name	Enable the record playback transcoding interface.	
Function	def PlayBackByDataType(cls, lLoginID: int, pstInParam: NET_IN_PLAYBACK_BY_DATA_TYPE, pstOutParam: NET_OUT_PLAYBACK_BY_DATA_TYPE, dwWaitTime: int = 5000) -> C_LLONG	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
	[in] pstInParam	Input parameter structure.
	[out] pstOutParam	Output parameter structure.
	[in] dwWaitTime	Waiting time.
Return value	Success: Non-0. Failure: 0	
Description	None.	

3.13 Enabling Record Download Transcoding Interface

Table 3-34 DownloadByDataType

Item	Description	
Name	Enable the record download transcoding interface.	
Function	def DownloadByDataType(cls, lLoginID: int, pstInParam: NET_IN_DOWNLOAD_BY_DATA_TYPE, pstOutParam: NET_OUT_DOWNLOAD_BY_DATA_TYPE, dwWaitTime: int = 5000) -> C_LLONG	
Parameter	[in] lLoginID	Return value of LoginWithHighLevelSecurity.
	[in] pstInParam	Input parameter structure.
	[out] pstOutParam	Output parameter structure.

Item	Description	
	[in] dwWaitTime	Waiting time.
Return value	Success: Non-0. Failure: 0	
Description	None.	

4 Callback　Definition

4.1 fDisConnect

Table 4-1 Disconnection callback

Item	Description	
Name	Disconnection callback.	
Precondition	None.	
Function	fDisConnect = WINFUNCTYPE(None, C_LLONG, c_char_p, c_long, C_LDWORD)	
Parameter	lLoginID	Login handle.
	pchDVRIP	IP address.
	nDVRPort	Port.
	dwUser	User data.
Return value	None.	
Note	Be triggered when the device is disconnected. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call.	

4.2 fHaveReConnect

Table 4-2 Reconnection callback

Item	Description	
Name	Reconnection callback.	
Precondition	None.	
Function	fHaveReConnect = WINFUNCTYPE(None, C_LLONG, c_char_p, c_long, C_LDWORD)	
Parameter	lLoginID	Login handle.
	pchDVRIP	IP address.
	nDVRPort	Port.
	dwUser	User data.
Return value	None.	
Note	Be triggered when the device is disconnected. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call.	

4.3 fSearchDevicesCBEx

Table 4-3 Async device search callback

Item	Description
Name	Device search callback prototype.

Item	Description	
Precondition	None.	
Function	fSearchDevicesCBEx = WINFUNCTYPE(None, C_LLONG, POINTER(DEVICE_NET_INFO_EX2), c_void_p)	
Parameter	lSearchHandle	Search handle.
	pDevNetInfo	● Device info.
	pUserData	User data info.
Return value	None.	
Note	Device search callback. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call. Set the callback by StartSearchDeviceEx. When a device is searched, the SDK will call this callback.	

4.4 fSearchDevicesCB

Table 4-4 Device search callback

Item	Description	
Name	Device search callback prototype.	
Precondition	None.	
Function	fSearchDevicesCB = WINFUNCTYPE(None, POINTER(DEVICE_NET_INFO_EX), c_void_p)	
Parameter	pDevNetInfo	Info.
	pUserData	User data info.
Return value	None.	
Note	Device search callback. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call. Set the callback by SearchDevicesByIPs. When a device is searched, the SDK will call this callback.	

4.5 fDownLoadPosCallBack

Table 4-5 Playback progress callback

Item	Description	
Name	Playback progress callback.	
Precondition	None.	
Function	fDownLoadPosCallBack = WINFUNCTYPE(None, C_LLONG, C_DWORD, C_DWORD, C_LDWORD)	
Parameter	lPlayHandle	Return handel of PlayBackByTimeEx.
	dwTotalSize	Total size of download.

Item	Description	
	dwDownLoadSize	The size that has been downloaded.. ● -1: Current download or playback has been finished. ● -2: The user does not have permission to download or playback.
	dwUser	User data.
Return value	None.	
Note	Playback progress callback. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call.	

4.6 fDataCallBack

Table 4-6 Playback data callback

Item	Description	
Name	Playback data callback.	
Precondition	None.	
Function	fDataCallBack = WINFUNCTYPE(c_int, C_LLONG, C_DWORD, POINTER(c_ubyte), C_DWORD, C_LDWORD)	
Parameter	lPlayHandle	Playback data handle.
	dwDataType	Data type.
	pBuffer	Data buffer. Memory is released internally by NetSDK.
	dwBufSize	Data buffer size.
	dwUser	User data.
Return value	● 1: Succeed to call back. ● 0: Failed to call back. The next callback will return the subsequent data.	
Note	Data callback of downloading records.. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call.	

4.7 fTimeDownLoadPosCallBack

Table 4-7 Callback of download by time callback

Item	Description	
Name	Callback of download by time.	
Precondition	None.	
Function	fTimeDownLoadPosCallBack = WINFUNCTYPE(None, C_LLONG, C_DWORD, C_DWORD, c_int, NET_RECORDFILE_INFO, C_LDWORD)	
Parameter	lPlayHandle	Return handel of DownloadByTimeEx.
	dwTotalSize	Total size of download.
	dwDownLoadSize	The size that has been downloaded..
	Index	Index.
	Recordfileinfo	Record file information.

Item	Description	
	dwUser	User data.
Return value	None.	
Note	Download progress callback. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call.	

4.8 fAnalyzerDataCallBack

Table 4-8 Intelligent image alarm callback

Item	Description	
Name	Intelligent image alarm callback.	
Precondition	None.	
Function	fAnalyzerDataCallBack = WINFUNCTYPE(None, C_LLONG, C_DWORD, c_void_p, POINTER(c_ubyte), C_DWORD, C_LDWORD, c_int, c_void_p)	
Parameter	lAnalyzerHandle	Return handel of RealLoadPictureEx.
	dwAlarmType	Event type of EM_EVENT_IVS_TYPE.
	pAlarmInfo	Event info.
	pBuffer	Image data buffer.
	dwBufSize	Image data buffer size.
	dwUser	User data info entered by RealLoadPictureEx..
	nSequence	Situation of the same uploaded image. ● 0: First time to appear. ● 1: Same image will appear from this time on. ● 2: Last time to appear or only once.
	Reserved	Indicate the status of current called back data when int nState = (int)reserved. ● 0: Current data is real-time data. ● 1: Current data is offline data. ● 2: Offline data transmission ends.
Return value	None.	
Note	Intelligent image alarm callback. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call. Set the callback by RealLoadPictureEx. When an intelligent image event is uploaded, the SDK will call this callback. The dwAlarmType value varies according to different data type of pAlarmInfo.	

4.9 fSnapRev

Table 4-9 Snapshot callback

Item	Description
Name	Snapshot callback prototype.

Item	Description	
Precondition	None.	
Function	fSnapRev = WINFUNCTYPE(None, C_LLONG, POINTER(c_ubyte), c_uint, c_uint, C_DWORD, C_LDWORD)	
Parameter	lLoginID	● Login handle.
	pBuf	● Image buffer.
	RevLen	Image size.
	EncodeType	● Encode type: ● 10: Jpeg image. ● 0: I frame of mpeg4.
	CmdSerial	Command serial No..
	dwUser	User data entered by SetSnapRevCallBack.
Return value	None.	
Note	Snapshot callback function. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call. Set this callback by SetSnapRevCallBack. When the snapshot data is sent, the SDK will call this callback.	

4.10 fMessCallBackEx1

Table 4-10 Alarm upload callback

Item	Description	
Name	Alarm upload callback prototype.	
Precondition	None.	
Function	fMessCallBackEx1 = WINFUNCTYPE(None, c_long, C_LLONG, POINTER(c_char), C_DWORD, POINTER(c_char), c_long, c_int, c_long, C_LDWORD)	
Parameter	lCommand	Alarm type.
	lLoginID	● Login handle.
	pBuf	● Alarm info.
	dwBufLen	Alarm info size.
	pchDVRIP	● IP address.
	nDVRPort	Port.
	bAlarmAckFlag	● 1: The event can be confirmed. ● 0: The event cannot be confirmed.
	nEventID	Used to assign values to the input parameters of the AlarmAck. hen bAlarmAckFlag is 1, the data is valid.
	dwUser	User data entered by SetDVRMessCallBackEx1.
Return value	None.	
Note	All registered devices use one alarm upload callback. You can identify the uploaded device by parameter lLoginI.D. Data type of pBuf varies according to lCommand value. It is not recommended to call any NetSDK interface in this callback. If the callback in the Demo calls, then you can follow and call.	

4.11 fDataCallBackEx

Table 4-11 fDataCallBackEx

Item	Description
Name	Extension 2 of the prototype for the live view transcoding data callback function.
Precondition	None.
Function	fDataCallBackEx = CB_FUNCTYPE(c_int, C_LLONG, POINTER(NET_DATA_CALL_BACK_INFO), C_LDWORD)
Parameter	● lRealHandle Live view handle. Return value of CLIENT_RealPlayEx and other interfaces for pulling live view streams. ● pDataCallBack Callback data. See NET_DATA_CALL_BACK_INFO for the structure details. ● dwUserData User data. Same as the input user data when users set the fRealDataCallBackEx callback function.
Return value	None.
Note	None.

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.

Appendix 2 Intelligent events

Type	Number	Notes
ALL	0x00000001	subscriptionallevent
CROSSLINEDETECTION	0x00000002	crosslineevent(CorrespondingtoDEV_EVENT_CROSSLINE_INFO)
CROSSREGIONDETECTION	0x00000003	crossregionevent(CorrespondingtoDEV_EVENT_CROSSREGION_INFO)
STAYDETECTION	0x00000006	Stay event (Corresponding to NET_A_DEV_EVENT_STAY_INFO)
WANDERDETECTION	0x00000007	Wanderevent(Correspondingto NET_A_DEV_EVENT_WANDER_INFO)
MOVEDETECTION	0x00000009	moveevent(CorrespondingtoDEV_EVENT_MOVE_INFO)
FIREDETECTION	0x0000000C	Fire event (Corresponding to NET_A_DEV_EVENT_FIRE_INFO)
SMOKEDETECTION	0x0000000D	Smokeevent(Correspondingto NET_A_DEV_EVENT_SMOKE_INFO)
FIGHTDETECTION	0x0000000E	fightevent(CorrespondingtoDEV_EVENT_FIGHT_INFO)
NUMBERSTAT	0x00000010	Quantity count event (Corresponding to NET_A_DEV_EVENT_NUMBERSTAT_INFO)
TRAFFICCONTROL	0x00000015	Traffic control event (Corresponding to NET_A_DEV_EVENT_TRAFFICCONTROL_INFO)
TRAFFICACCIDENT	0x00000016	Traffic accident event (Corresponding to NET_A_DEV_EVENT_TRAFFICACCIDENT_INFO)
TRAFFICJUNCTION	0x00000017	trafficjunctionevent(CorrespondingtoDEV_EVENT_TRAFFICJUNCTION_INFO)
TRAFFICGATE	0x00000018	Traffic ANPR event (Corresponding to NET_A_DEV_EVENT_TRAFFICGATE_INFO)
TRAFFICSNAPSHOT	0x00000019	Traffic snapshot event (Corresponding to NET_A_DEV_EVENT_TRAFFICSNAPSHOT_INFO)
FACEDETECT	0x0000001A	facedetection(CorrespondingtoDEV_EVENT_FACEDETECT_INFO)
TRAFFICJAM	0x0000001B	Traffic congestion (Corresponding to NET_A_DEV_EVENT_TRAFFICJAM_INFO)
TRAFFIC_NONMOTORINMOTORROUTE	0x0000001C	Non-motor vehicle in motor vehicle lane (Corresponding to NET_A_DEV_EVENT_TRAFFIC_NONMOTORINMOTORROUTE_INFO)
TRAFFIC_RUNREDLIGHT	0x00000100	Traffic violation: Running red light

		(Corresponding to NET_A_DEV_EVENT_TRAFFIC_RUNREDLIGHT _INFO)
TRAFFIC_OVERLINE	0x00000101	Traffic violation: Driving on lane (Corresponding to NET_A_DEV_EVENT_TRAFFIC_OVERLINE_INF O)
TRAFFIC_RETROGRADE	0x00000102	Traffic violation: Wrong-way driving (Corresponding to NET_A_DEV_EVENT_TRAFFIC_RETROGRADE_ INFO)
TRAFFIC_TURNLEFT	0x00000103	Traffic violation: Turning left (Corresponding to NET_A_DEV_EVENT_TRAFFIC_TURNLEFT_INF O)
TRAFFIC_TURNRIGHT	0x00000104	Traffic violation: Turning right (Corresponding to NET_A_DEV_EVENT_TRAFFIC_TURNRIGHT_IN FO)
TRAFFIC_UTURN	0x00000105	Traffic violation: U turn (Corresponding to NET_A_DEV_EVENT_TRAFFIC_UTURN_INFO)
TRAFFIC_OVERSPEED	0x00000106	Traffic violation: Speeding (Corresponding to NET_A_DEV_EVENT_TRAFFIC_OVERSPEED_IN FO)
TRAFFIC_UNDERSPEED	0x00000107	Traffic violation: Driving too slow (Corresponding to NET_A_DEV_EVENT_TRAFFIC_UNDERSPEED_ INFO)
TRAFFIC_PARKING	0x00000108	Traffic violation: Parking (Corresponding to NET_A_DEV_EVENT_TRAFFIC_PARKING_INFO)
TRAFFIC_WRONGROUTE	0x00000109	Traffic violation: Wrong lane (Corresponding to NET_A_DEV_EVENT_TRAFFIC_WRONGROUTE _INFO)
TRAFFIC_CROSSLANE	0x0000010A	Traffic violation: Illegal lane change (Corresponding to NET_A_DEV_EVENT_TRAFFIC_CROSSLANE_I NFO)
TRAFFIC_OVERYELLOWLINE	0x0000010B	Traffic violation: Crossing solid yellow line (Corresponding to NET_A_DEV_EVENT_TRAFFIC_OVERYELLOWL INE_INFO)
TRAFFIC_DRIVINGONSHOULDER	0x0000010C	Traffic violation: Driving on shoulder (Corresponding to NET_A_DEV_EVENT_TRAFFIC_DRIVINGONSH OULDER_INFO)

TRAFFIC_YELLOWPLATEINLANE	0x0000010E	Traffic violation: Vehicle with yellow plate in lane (Corresponding to NET_A_DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO)
TRAFFIC_PEDESTRAINPRIORITY	0x0000010F	Traffic violation: Pedestrian priority on zebra crossing (Corresponding to NET_A_DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO)
TRAFFIC_NOPASSING	0x00000111	Traffic violation: No passing event (Corresponding to NET_A_DEV_EVENT_TRAFFIC_NOPASSING_INFO)
FACERECOGNITION	0x00000117	targetrecognition(CorrespondingtoNET_DEV_EVENT_FACERECOGNITION_INFO)
TRAFFIC_FLOWSTATE	0x00000119	Trafficflowstate (CorrespondingtoDEV_EVENT_TRAFFIC_FLOW_STATE)
TRAFFIC_BACKING	0x00000125	Illegal backing event (Corresponding to NET_A_DEV_EVENT_IVS_TRAFFIC_BACKING_INFO)
TRAFFIC_PEDESTRAIN	0x0000012D	Pedestrian violation (Corresponding to NET_A_DEV_EVENT_TRAFFIC_PEDESTRAIN_INFO)
TRAFFIC_THROW	0x0000012E	Littering event (Corresponding to NET_A_DEV_EVENT_TRAFFIC_THROW_INFO)
TRAFFIC_DRIVER_SMOKING	0x00000139	Driversmokingevent(CorrespondingtoNET_A_DEV_EVENT_TRAFFIC_DRIVER_SMOKING)
TRAFFIC_DRIVER_CALLING	0x0000013A	Drivercallingevent(CorrespondingtoNET_A_DEV_EVENT_TRAFFIC_DRIVER_CALLING)
VIDEOBLIND	0x00000153	Videoocclusionevent(CorrespondingtoNET_A_DEV_EVENT_ALARM_VIDEOBLIND)
TUMBLE_DETECTION	0x00000177	Fall alarm event (Corresponding to NET_A_DEV_EVENT_ALARM_TUMBLE_DETECTION_INFO)
ACCESS_CTL	0x00000204	Acccesscontrolevents(CorrespondingtoDEV_EVENT_ACCESS_CTL_INFO)
TRAFFIC_TIREDPHYSIOLOGICAL	0x00000207	Physiologicalfatiguedrivingevent(CorrespondingtoNET_A_DEV_EVENT_TIREDPHYSIOLOGICAL_INFO)
TRAFFIC_TIREDLOWERHEAD	0x0000020A	Startuplowheadalarmevent(CorrespondingtoNET_A_DEV_EVENT_TIREDLOWERHEAD_INFO)
TRAFFIC_DRIVERLOOKAROUND	0x0000020B	Drivingleftandrightlookingalarmevents(CorrespondingtoNET_A_DEV_EVENT_DRIVERLOOKAROUND_INFO)
TRAFFIC_DRIVERLEAVEPOST	0x0000020C	Leavingpostduringdrivingalarmevent(CorrespondingtoNET_A_DEV_EVENT_DRIVERLEAVE

		POST_INFO)
MAN_NUM_DETECTION	0x0000020E	Stereo vision: Area people counting (Corresponding to NET_A_DEV_EVENT_MANNUM_DETECTION_INFO)
TRAFFIC_DRIVERYAWN	0x00000210	Yawningincidentduringdriving(CorrespondingtoNET_A_DEV_EVENT_DRIVERYAWN_INFO)
TRAFFIC_QUEUEJUMP	0x0000021C	Vehicle jumping queue (Corresponding to NET_A_DEV_EVENT_TRAFFIC_QUEUEJUMP_INFO)
CROWDDETECTION	0x0000022C	Eventofcrowddetection(CorrespondingtoDEV_EVENT_CROWD_DETECTION_INFO)
FIREWARNING	0x00000245	Fire event (Corresponding to NET_A_DEV_EVENT_FIREWARNING_INFO)
SHOPPRESENCE	0x00000246	Roadside stall (Corresponding to NET_A_DEV_EVENT_SHOPPRESENCE_INFO)
SPILLEDMATERIAL_DETECTION	0x00000248	Littering detection event (Corresponding to NET_A_DEV_EVENT_SPILLEDMATERIAL_DETECTION_INFO)
TRAFFIC_NONMOTOR_WITHOUTSAFEHAT	0x0000024C	Non-motor vehicle without helmet (Corresponding to NET_A_DEV_EVENT_TRAFFIC_NONMOTOR_WITHOUTSAFEHAT_INFO)
FLOWBUSINESS	0x0000024E	Unlicensed stall vendor (Corresponding to NET_A_DEV_EVENT_FLOWBUSINESS_INFO)
DUSTBIN_OVER_FLOW	0x00000260	Full garbage can detection (Corresponding to DEV_EVENT_DUSTBIN_OVER_FLOW_INFO)
CLASSROOM_BEHAVIOR	0x0000026A	Classroom behavior analysis (Corresponding to NET_A_DEV_EVENT_CLASSROOM_BEHAVIOR_INFO)
WORKCLOTHES_DETECT	0x0000026E	Workwear detection event, including helmet. (Corresponding to NET_A_DEV_EVENT_WORKCLOTHES_DETECT_INFO)
TRAFFIC_ROAD_BLOCK	0x00000271	Traffic barrier detection (Corresponding to NET_A_DEV_EVENT_TRAFFIC_ROAD_BLOCK_INFO)
TRAFFIC_ROAD_CONSTRUCTION	0x00000272	Road construction detection (Corresponding to NET_A_DEV_EVENT_TRAFFIC_ROAD_CONSTRUCTION_INFO)
OPEN_INTELLI	0x0000039D	Openintelligentevent(correspondingtodev_event_open_intelli_info)
TRAFFIC_MANUALSNAP	0x00000118	Trafficmanualcaptureevent(correspondingtodev_event_traffic_manualsnap_info)

SMARTMOTION_HUMAN	0x00000279	Intelligentvideomotiondetectionevent(person),(correspondingtoDEV_EVENT_SMARTMOTION_HUMAN_INFO)
RADAR_REGION_DETECTION	0x00000295	EventofRadarcrossregiondetection(CorrespondingtoNET_A_DEV_EVENT_RADAR_REGION_DETECTION_INFO)
DIALRECOGNITION	0x00000371	Instrumentdetectionevent(correspondingtoDEV_EVENT_DIALRECOGNITION_INFO)
ELECTRICFAULT_DETECT	0x00000372	Instrumentdefectdetectionevent(correspondingtoDEV_EVENT_ELECTRICFAULTDETECT_INFO)
CAR_DRIVING_IN_OUT	0x0000027B	Vehicle entering or exiting status (Corresponding to DEV_EVENT_CAR_DRIVING_IN_OUT_INFO)
PARKINGSPACE_STATUS	0x0000027C	Parking space status (Corresponding to DEV_EVENT_PARKINGSPACE_STATUS_INFO)
ANIMAL_DETECTION	0x00000286	Animal detection event (Corresponding to NET_A_DEV_EVENT_ANIMAL_DETECTION_INFO)
DREGS_UNCOVERED	0x00000299	Dump truck load uncovered (Corresponding to NET_A_DEV_EVENT_DREGS_UNCOVERED_INFO)
ANATOMY_TEMP_DETECT	0x00000303	Intelligent body temperature measurement event (Corresponding to DEV_EVENT_ANATOMY_TEMP_DETECT_INFO)
NONMOTOR_ENTRYING	0x0000030C	Non-motor vehicle in elevator (Corresponding to DEV_EVENT_NONMOTOR_ENTRYING_INFO)
TRAFFIC_ROAD_ALERT	0x0000030E	Road security early warning (Corresponding to NET_A_DEV_EVENT_TRAFFIC_ROAD_ALERT_INFO)
TRAFFIC_VEHICLE_IN_EMERGENCY_LANE	0x00000311	Emergency lane occupancy event (Corresponding to NET_A_DEV_EVENT_TRAFFIC_VEHICLE_IN_EMERGENCY_LANE_INFO)
TRAFFIC_SPECIAL_VEHICLE_DETECT	0x00000333	Special vehicle detection (Corresponding to NET_A_DEV_EVENT_TRAFFIC_SPECIAL_VEHICLE_INFO)
TRAFFIC_NONMOTOR	0x00000335	Non-motor vehicle event detection, used for intelligent server (NET_A_DEV_EVENT_TRAFFIC_NONMOTOR_INFO)
TRAFFIC_BOARD	0x00000336	Illegal pick-up and drop-off event detection (Corresponding to

		NET_A_DEV_EVENT_TRAFFIC_BOARD_INFO)
TRAFFIC_VISIBILITY	0x00000337	Traffic visibility event detection (Corresponding to NET_A_DEV_EVENT_TRAFFIC_VISIBILITY_INFO)
TRAFFIC_VEHICLE_CLEANLINESS	0x00000338	Vehicle cleanliness detection event (Corresponding to NET_A_DEV_EVENT_TRAFFIC_VEHICLE_CLEANLINESS_INFO)
TRAFFIC_SPEED_CHANGE_DETECTION	0x0000034E	Variable speed detection event (Corresponding to NET_A_DEV_EVENT_TRAFFIC_SPEED_CHANGE_DETECTION_INFO)
CONVEYER_BELT_BULK	0x00000351	Large foreign body detection for conveyor belt (Corresponding to DEV_EVENT_CONVEYER_BELT_BULK_INFO)
HEAT_IMAGING_TEMPER	0x0000035C	Abnormal temperature alarm for thermal temperature monitoring point (Corresponding to structure DEV_EVENT_HEAT_IMAGING_TEMPER_INFO)
TRAFFIC_CHANGE_LANE_CONTINUES	0x00000387	Continuous lane change of motor vehicle (Corresponding to NET_A_DEV_EVENT_TRAFFIC_CHANGE_LANE_CONTINUES_INFO)
TRAFFIC_CROSSING_SPEEDY	0x00000408	Not slowing down at crosswalk (Corresponding to NET_A_DEV_EVENT_TRAFFIC_CROSSING_SPEEDY_INFO)
TRAFFIC_LARGECAR_NO_STOP	0x00000409	Truck right-turn non-stop event (Corresponding to NET_A_DEV_EVENT_TRAFFIC_LARGECAR_NO_STOP_INFO)
TRAFFIC_TRUCK_OCCUPIED	0x0000040B	Large vehicle in lane (Corresponding to NET_A_DEV_EVENT_TRAFFIC_TRUCK_OCCUPIED_INFO)
TRAFFIC_SERPENTINE_CHANGE_LANE	0x0000040F	S-shaped lane change (Corresponding to NET_A_DEV_EVENT_TRAFFIC_SERPENTINE_CHANGE_LANE_INFO)
TANK_CAPACITY_DETECTION	0x00000412	Storage tank level detection (Corresponding to DEV_EVENT_IVS_TANK_CAPACITY_DETECTION_INFO)
TANK_DUMPING_DETECTION	0x00000413	Detection for material disposal in transfer boxes (Corresponding to DEV_EVENT_IVS_TANK_DUMPING_DETECTION_INFO)
TANK_OVERFLOW_DETECTION	0x00000414	Full material box detection (Corresponding to

		DEV_EVENT_IVS_TANK_OVERFLOW_DETECTION_INFO)
USERMANAGER_FOR_TWSDK	0x00000441	Reporting user information event (Corresponding to NET_DEV_EVENT_USERMANAGER_FOR_TWSDK_INFO)
TIMECHANGE_FOR_TWSDK	0x0000044F	System time change event (Corresponding to NET_DEV_EVENT_TIMECHANGE_FOR_TWSDK_INFO)
SAME_OBJECT_SEARCH_DETECT	0x00000472	Search target by image - object detection event (Corresponding to NET_DEV_EVENT_SAME_OBJECT_SEARCH_DETECT_INFO)
SAME_OBJECT_SEARCH_COUNT	0x00000480	Search target by image - object counting event (Corresponding to NET_DEV_EVENT_SAME_OBJECT_SEARCH_COUNT_INFO)

Appendix 3 General PTZ control command enumeration SDK_PTZ_ControlType

General PTZ control command	Number	Notes
UP_CONTROL	0	Up. Speed corresponds to param2 (1–8).
DOWN_CONTROL	1	Down. Speed corresponds to param2 (1–8).
LEFT_CONTROL	2	Left. Speed corresponds to param2 (1–8).
RIGHT_CONTROL	3	Right. Speed corresponds to param2 (1–8).
ZOOM_ADD_CONTROL	4	Zoom+ corresponds to param2.
ZOOM_DEC_CONTROL	5	Zoom- corresponds to param2.
FOCUS_ADD_CONTROL	6	Focus- corresponds to param2.
FOCUS_DEC_CONTROL	7	Focus+ corresponds to param2.
APERTURE_ADD_CONTROL	8	Aperture+ corresponds to param2.
APERTURE_DEC_CONTROL	9	Aperture- corresponds to param2.
POINT_MOVE_CONTROL	10	Move to preset. The preset number is param2.
POINT_SET_CONTROL	11	Set. The preset number is param2. The maximum number can be obtained from the PTZ capability set. You can enter the preset name into param4. The maximum valid value for the name is 63 bytes. Param3 needs to be of type bool. When param3 is true, the preset name must not exceed 255 characters.
POINT_DEL_CONTROL	12	Delete. The preset number is param2.
POINT_LOOP_CONTROL	13	Tour between points. The tour route is param1.
LAMP_CONTROL	14	Light wiper corresponds to param1 (1: On, 0: Off).
LEFTTOP	32	Left top
RIGHTTOP	33	Right top
LEFTDOWN	34	Left bottom
RIGHTDOWN	35	Right bottom
ADDTOLOOP	36	Add preset to tour. Tour route. Value of preset.
DELFROMLOOP	37	Delete preset in tour. Tour route. Value of preset.
CLOSELOOP	38	Delete tour. Tour route.
STARTPANCRUISE	39	Start horizontal rotation.
STOPPANCRUISE	40	Stop horizontal rotation.
SETLEFTBORDER	41	Set left border.
SETRIGHTBORDER	42	Set right border.
STARTLINESCAN	43	Start line scan.
CLOSELINESCAN	44	Stop line scan.
SETMODESTART	45	Set mode start. Mode route.
SETMODESTOP	46	Set mode stop. Mode route.
RUNMODE	47	Running mode. Mode route.
STOPMODE	48	Stop mode. Mode route.

DELETEMODE	49	Delete mode. Mode route.
REVERSECOMM	50	Reverse command.
FASTGOTO	51	Fast positioning. Horizontal coordinate (-8191–8191). Vertical coordinate (-8191–8191). Zoom (-16–16).
OPENMENU	54	Open PTZ camera menu.
CLOSEMENU	55	Close PTZ camera menu.
MENUOK	56	Menu confirm.
MENUCANCEL	57	Menu cancel.
MENUUP	58	Menu up.
MENUDOWN	59	Menu down.
MENULEFT	60	Menu left.
MENURIGHT	61	Menu right.
ALARMHANDLE	64	Link alarm to PTZ. Param1: Alarm input channel. Param2: Alarm linkage type. 1: Preset. 2: Line scan. 3: Tour. Param3: Linkage value, such as preset number.
MATRIXSWITCH	65	Matrix switch. Param1: Monitor device number (video output number). Param2: Video input number. Param3: Matrix number.
LIGHTCONTROL	66	Light controller.
EXACTGOTO	67	3D positioning. Param1: Horizontal angle (0–3600). Param2: Vertical coordinate (-1800–1800). Param3: Zoom (1–128), which is the level, rather than the actual zoom ratio.
RESETZERO	68	Reset 3D positioning to zero.
MOVE_ABSOLUTELY	69	Absolute movement control command. Param4 corresponds to structure PTZ_CONTROL_ABSOLUTELY
MOVE_CONTINUOUSLY	70	Continuous movement control command. Param4 corresponds to structure NET_A_PTZ_CONTROL_CONTINUOUSLY.
SET_ABS_ZOOMFOCUS	77	Set the absolute focal length and focus value. Param1 is focal length, range: [0,255]. Param2 is focus, range: [0,255]. Param3 and param4 are invalid.
RESTART	81	PTZ restart command. Param1, param2, param3 and param4 are invalid. Set dwStop to false.
INTELLI_TRACKMOVE	82	Continuous movement of PTZ. Exclusive for smart tracking. Param4 corresponds to structure PTZ_CONTROL_INTELLI_TRACKMOVE.
PAUSELINESCAN	84	Stop line scan. Param1, param2, param3 and param4 are invalid. Set dwStop to false.
UP_TELE	112	Up + TELE. Param1: Speed (1–8), the same below.
DOWN_TELE	113	Down + TELE.
LEFT_TELE	114	Left + TELE.

RIGHT_TELE	115	Right + TELE.
LEFTUP_TELE	116	Leftup +TELE.
LEFTDOWN_TELE	117	Leftdown + TELE.
TIGHTUP_TELE	118	Rightup + TELE.
RIGHTDOWN_TELE	119	Rightdown + TELE.
UP_WIDE	120	Up + WIDE. Param1: Speed (1–8), the same below.
DOWN_WIDE	121	Down + WIDE.
LEFT_WIDE	122	Left + WIDE.
RIGHT_WIDE	123	Right + WIDE.
LEFTUP_WIDE	124	Leftup + WIDE.
LEFTDOWN_WIDE	125	Leftdown + WIDE.
TIGHTUP_WIDE	126	Rightup + WIDE
RIGHTDOWN_WIDE	127	Rightdown + WIDE
GOTOPRESETSNAP	128	Turn to preset and take snapshots.
DIRECTIONCALIBRATION	130	Calibrate PTZ direction from two directions.
BASE_MOVE_ABSOLUTELY	134	Accurate and absolute movement control command. Param4 corresponds to structure NET_IN_PTZBASE_MOVEABSOLUTELY_INFO. Use the CFG_CAP_CMD_TYPE. PTZ command to get PTZ capability set CFG_PTZ_PROTOCOL_CAPS_INFO. If bSupportReal is true, it means that the operation can be performed on the device.
BASE_MOVE_ABSOLUTELY_ONLYPT	137	Absolute positioning allows you to control PTZ to move at a speed measured in degrees per second. Param4 corresponds to structure NET_IN_PTZBASE_MOVEABSOLUTELY_ONLYPT_I NFO.
BASE_MOVE_ABSOLUTELY_ONLYZOOM	138	Absolute positioning allows you to control zoom and the zooming speed. Param4 corresponds to structure NET_IN_PTZBASE_MOVEABSOLUTELY_ONLYZOO M_INFO.
STOP_MOVE	139	Stop PTZ movement and tour mode. Param4 corresponds to structure NET_IN_STOP_MOVE_INFO.
TOTAL	140	Maximum command value.