

NetSDK (Camera)

Programming Manual



Foreword

Purpose

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

SDK, also known as network device SDK, is a development kit for developer to develop the interfaces for network communication among surveillance products such as Network Video Recorder (NVR), Network Video Server (NVS), IP Camera (IPC), Speed Dome (SD), and intelligence devices.

The Manual describes the SDK interfaces and processes of the general function modules for IPC, SD and Thermal IP Camera (TPC). For more function modules and data structures, refer to *NetSDK Development Manual*.






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
V1.0.4	<ul style="list-style-type: none">Deleted function library avnetsdk.dll and libavnetsdk.so related content, and changed font.Deleted fisheye correction library.	May 2021
V1.0.3	Change callback functions of login and device search.	March 2020
V1.0.2	Add "2.10 Firepoint", " 2.11 Thermal PTZ Control", "3.10 Thermal Firepoint Control", "4.7 fAnalyzerDataCallback" and "4.8 fMessCallback."	May 2019
V1.0.1	Delete some library files in "Table 1-1."	January 2019
V1.0.0	First release.	December 2017

Privacy Protection Notice

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

About the Manual

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

Glossary

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

Term	Definition
Main Stream	A type of video stream that usually has better resolution and clarity and provides a better experience if the network resource is not restricted.
Sub Stream	A type of video stream that usually has lower resolution and clarity than the main stream but demands less network resources. The user can choose the stream type according to the particular scenes.
Video Channel	The video is numbered from 0 and each video receives a channel number. Currently, except the TPC, the other types of devices usually have only one channel that is numbered as 0.
Login Handle	The first step to access to the device is login (authentication). The device receives a unique ID that refers to the login handle upon the successful login. This handle will be used by the subsequent procedures and stay valid until logout.
Relative Positioning	A fast positioning method in PTZ control by providing the difference value of the PTZ coordinates (X-axis and Y-axis) to the device which accord to the present PTZ location and the difference value to calculate and transfer to the final location. This method also supports ZOOM control.
Absolute Positioning	A fast positioning method in PTZ control which provides certain horizontal and vertical coordinates (angular coordinate) to the device. The device directly transfers to the user specified location. This method also supports ZOOM control.
PCM	Pulse Code Modulation is one of the coding methods of digital communication and converts the analog signal into digital signal without encoding loss. It is suitable for the user who requires higher data transfer rate and bandwidth.
PTZ	Pan Tilt Zoom is all-round movement and lens zoom control.
Heat Map (Temperature)	Corresponds to the temperature distribution data of a scene and each pixel has a data point.
Heat Map (Activity)	Indicates a picture with using different colors to show the statistical activity of an area during a certain period.

Table of Contents

Foreword	I
Glossary	III
1 Overview	1
1.1 General.....	1
1.2 Applicability	2
2 Function Modules	3
2.1 SDK Initialization	3
2.1.1 Introduction.....	3
2.1.2 Interface Overview	3
2.1.3 Process	4
2.1.4 Example Code	5
2.2 Device Initialization	5
2.2.1 Introduction.....	5
2.2.2 Interface Overview	5
2.2.3 Process	6
2.2.4 Example Code	8
2.3 Device Login	10
2.3.1 Introduction.....	10
2.3.2 Interface Overview	10
2.3.3 Process	11
2.3.4 Example Code	12
2.4 Real-time Monitoring	13
2.4.1 Introduction.....	13
2.4.2 Interface Overview	13
2.4.3 Process	13
2.4.4 Example Code	17
2.5 Video Snapshot.....	18
2.5.1 Introduction.....	18
2.5.2 Interface Overview	18
2.5.3 Process	18
2.5.4 Example Code	21
2.6 PTZ Control.....	21
2.6.1 Introduction.....	21
2.6.2 Interface Overview	21
2.6.3 Process	22
2.6.4 Example Code	24
2.7 Voice Talk.....	24
2.7.1 Introduction.....	24
2.7.2 Interface Overview	24
2.7.3 Process	25
2.7.4 Example Code	26
2.8 Heat Map (Temperature).....	27
2.8.1 Introduction.....	27
2.8.2 Interface Overview	27

2.8.3 Process	28
2.8.4 Example Code	29
2.9 Heat Map (Activity)	30
2.9.1 Introduction	30
2.9.2 Interface Overview	30
2.9.3 Process	31
2.9.4 Example Code	31
2.10 Firepoint	32
2.10.1 Introduction	32
2.10.2 Interface Overview	32
2.10.3 Process	33
2.10.4 Example Code	39
2.11 Thermal PTZ Control	41
2.11.1 Introduction	41
2.11.2 Interface Overview	42
2.11.3 Process	43
2.11.4 Example Code	48
3 Interface Definition	49
3.1 SDK Initialization	49
3.1.1 SDK CLIENT_Init	49
3.1.2 CLIENT_Cleanup	49
3.1.3 CLIENT_SetAutoReconnect	49
3.1.4 CLIENT_SetNetworkParam	50
3.2 Device Initialization	50
3.2.1 CLIENT_StartSearchDevicesEx	50
3.2.2 CLIENT_InitDevAccount	50
3.2.3 CLIENT_GetDescriptionForResetPwd	51
3.2.4 CLIENT_CheckAuthCode	52
3.2.5 CLIENT_ResetPwd	52
3.2.6 CLIENT_GetPwdSpecification	53
3.2.7 CLIENT_StopSearchDevices	53
3.3 Device Login	53
3.3.1 CLIENT_LoginWithHighLevelSecurity	53
3.3.2 CLIENT_Logout	54
3.4 Real-time Monitoring	55
3.4.1 CLIENT_RealPlayEx	55
3.4.2 CLIENT_StopRealPlayEx	56
3.4.3 CLIENT_SaveRealData	56
3.4.4 CLIENT_StopSaveRealData	56
3.4.5 CLIENT_SetRealDataCallBackEx2	57
3.5 Video Snapshot	57
3.5.1 CLIENT_SnapPictureToFile	57
3.5.2 CLIENT_CapturePictureEx	58
3.6 PTZ Control	58
3.6.1 CLIENT_DHPTZControlEx2	58
3.7 Voice Talk	62
3.7.1 CLIENT_StartTalkEx	62

3.7.2 CLIENT_StopTalkEx	63
3.7.3 CLIENT_RecordStartEx	63
3.7.4 CLIENT_RecordStopEx	63
3.7.5 CLIENT_TalkSendData	63
3.7.6 CLIENT_AudioDecEx	64
3.8 Heat Map (Temperature)	64
3.8.1 CLIENT_RadiometryAttach	64
3.8.2 CLIENT_RadiometryDetach	65
3.8.3 CLIENT_RadiometryFetch	65
3.8.4 CLIENT_RadiometryDataParse	65
3.9 Heat Map (Activity)	66
3.9.1 CLIENT_QueryDevState	66
3.10 Thermal Firepoint Control	67
3.10.1 CLIENT_SetConfig	67
3.10.2 CLIENT_GetConfig	67
3.10.3 CLIENT_PacketData	68
3.10.4 CLIENT_SetNewDevConfig	69
3.10.5 CLIENT_GetNewDevConfig	69
3.10.6 CLIENT_ParseData	70
3.10.7 CLIENT_RealLoadPictureEx	71
3.10.8 CLIENT_StopLoadPic	71
3.10.9 CLIENT_SetDVRMessCallBack	72
3.10.10 CLIENT_StartListenEx	72
3.10.11 CLIENT_StopListen	72
3.10.12 CLIENT_OperateCalibrateInfo	73
3.11 Thermal PTZ Control	73
3.11.1 CLIENT_SetConfig	73
3.11.2 CLIENT_GetConfig	74
3.11.3 CLIENT_PTZSetPanGroup	75
3.11.4 CLIENT_PTZGetPanGroup	75
3.11.5 CLIENT_PTZSetPanGroupLimit	75
3.11.6 CLIENT_PTZGotoPanPosition	76
3.11.7 CLIENT_PausePtzAction	76
3.11.8 CLIENT_ResumePtzLastTask	77
4 Callback Definition	78
4.1 fSearchDevicesCB	78
4.2 fSearchDevicesCBEx	78
4.3 fDisConnect	78
4.4 fHaveReConnect	79
4.5 fRealDataCallBackEx2	79
4.6 pfAudioDataCallBack	80
4.7 fRadiometryAttachCB	81
4.8 fAnalyzerDataCallBack	81
4.9 fMessCallBack	82
Appendix 1 Cybersecurity Recommendations	84

1 Overview

1.1 General

The Manual introduces SDK interfaces reference information that includes main function modules, interface definition, and callback definition.

The following are the main functions:

SDK initialization, device login, real-time monitoring, PTZ control, voice talk, video snapshot, heat map (temperature) and heat map (activity).

The development kit might be different dependent on the environment.

Table 1-1 Files of Windows development kit

Library type	Library file name	Library file description
Function library	dhnetsdk.h	Header file
	dhnetsdk.lib	Lib file
	dhnetsdk.dll	Library file
	avnetsdk.dll	Library file
Configuration library	avglobal.h	Header file
	dhconfigsdk.h	Configuration Header file
	dhconfigsdk.lib	Lib file
	dhconfigsdk.dll	Library file
Auxiliary library of playing (coding and decoding)	dhplay.dll	Playing library
Auxiliary library of "dhnetsdk.dll"	lvsDrawer.dll	Image display library
	StreamConvertor.dll	Transcoding library

Table 1-2 Files of Linux development kit

Library type	Library file name	Library file description
Function library	dhnetsdk.h	Header file
	libdhnetsdk.so	Library file
	libavnetsdk.so	Library file
Configuration library	avglobal.h	Header file
	dhconfigsdk.h	Configuration Header file
	libdhconfigsdk.so	Configuration library
Auxiliary library of "libdhnetsdk.so"	libStreamConvertor.so	Transcoding library



- The function library and configuration library are necessary libraries.
- The function library is the main body of SDK, which is used for communication interaction between client and products, remotely controls device, queries device data, configures device data information, as well as gets and handles the streams.
- The configuration library packs and parses the structures of configuration functions.

- It is recommended to use auxiliary library of playing (coding and decoding) to parse and play the streams.
- The auxiliary library decodes the audio and video streams for the functions such as monitoring and voice talk, and collects the local audio.

1.2 Applicability

- Recommended memory: No less than 512 M
- System supported by SDK:
 - ◇ Windows
Windows 10/Windows 8.1/Windows 7 and Windows Server 2008/2003
 - ◇ Linux
The common Linux systems such as Red Hat/SUSE

2 Function Modules

There are seven function modules in this chapter. Each function module includes SDK initialization, device login, logout, and SDK resource release. The optional processes do not affect the use of other processes.

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 **CLIENT_Cleanup** to release SDK resource.

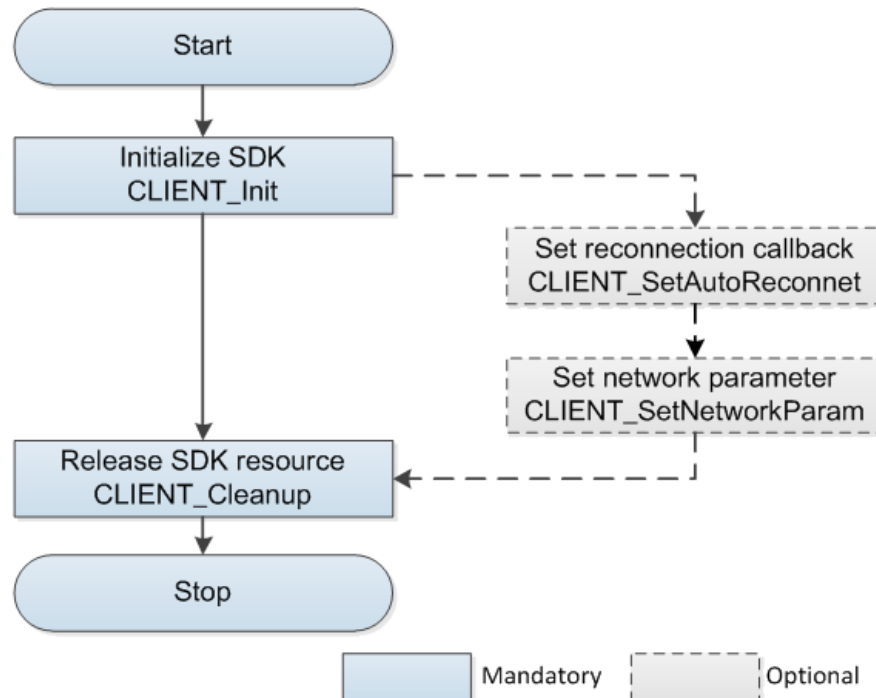
2.1.2 Interface Overview

Table 2-1 Interfaces of SDK initialization

Interface	Implication
CLIENT_Init	SDK initialization
CLIENT_Cleanup	SDK cleaning up
CLIENT_SetAutoReconnect	Setting of reconnection after disconnection
CLIENT_SetNetworkParam	Setting of network environment

2.1.3 Process

Figure 2-1 Process of SDK initialization



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 (Optional) Call **CLIENT_SetAutoReconnect** to set reconnection callback to allow the auto reconnecting after disconnection.
- Step 3 (Optional) Call **CLIENT_SetNetworkParam** to set network login parameter that includes connection timeout and connection attempts.
- Step 4 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Call **CLIENT_Init** and **CLIENT_Cleanup** in pairs. It supports single thread multiple calling but it is suggested to call the pair for only one time overall.
- Initialization: Calling **CLIENT_Init** multiple times is only for internal count without repeating applying resources.
- Cleaning up: The interface **CLIENT_Cleanup** clears all the opened processes, such as login, real-time monitoring, and alarm subscription.
- Reconnection: SDK can set the reconnection function for the situations such as network disconnection and power off. SDK will keep logging the device until succeeded. Only the real-time monitoring, alarm and snapshot subscription can be resumed after reconnection is successful.

2.1.4 Example Code

```
// Set this callback through CLIENT_Init. When the device is disconnected, SDK informs the user through this callback.
void CALLBACK DisConnectFunc(LLONG ILoginID, char *pchDVRIP, LONG nDVRPort, DWORD dwUser)
{
    printf("Call DisConnectFunc: ILoginID[0x%x]\n", ILoginID);
}

// Initialize SDK
CLIENT_Init(DisConnectFunc, 0);

// .... Call the functional interface to handle the process

// Clean up the SDK resource
CLIENT_Cleanup();
```

2.2 Device Initialization

2.2.1 Introduction

The device is uninitialized by default. Please initialize the device before starting use.

- The uninitialized device cannot be logged in.
- A password will be set for the default admin account during initialization.
- You can reset the password if you forgot it.

2.2.2 Interface Overview

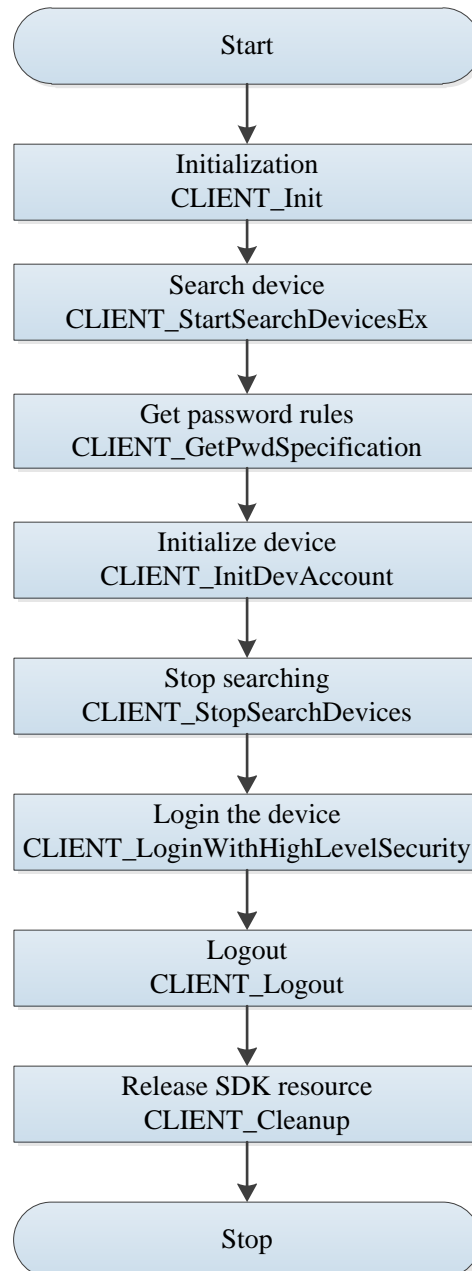
Table 2-2 Interfaces of device initialization

Interface	Implication
CLIENT_StartSearchDevicesEx	Search in the LAN to find the uninitialized devices.
CLIENT_InitDevAccount	Initialization interface.
CLIENT_GetDescriptionForResetPwd	Get the password reset information: mobile phone number, email address, and QR code.
CLIENT_CheckAuthCode	Check the validity of security code.
CLIENT_ResetPwd	Reset password.
CLIENT_GetPwdSpecification	Get the password rules.
CLIENT_StopSearchDevices	Stop searching.

2.2.3 Process

2.2.3.1 Device Initialization

Figure 2-2 Process of device initialization



Process Description

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

Step 2 Call **CLIENT_StartSearchDevicesEx** to search the devices within the LAN and get the device information.



Multi-thread calling is not supported.

Step 3 Call **CLIENT_GetPwdSpecification** to get the password rules.

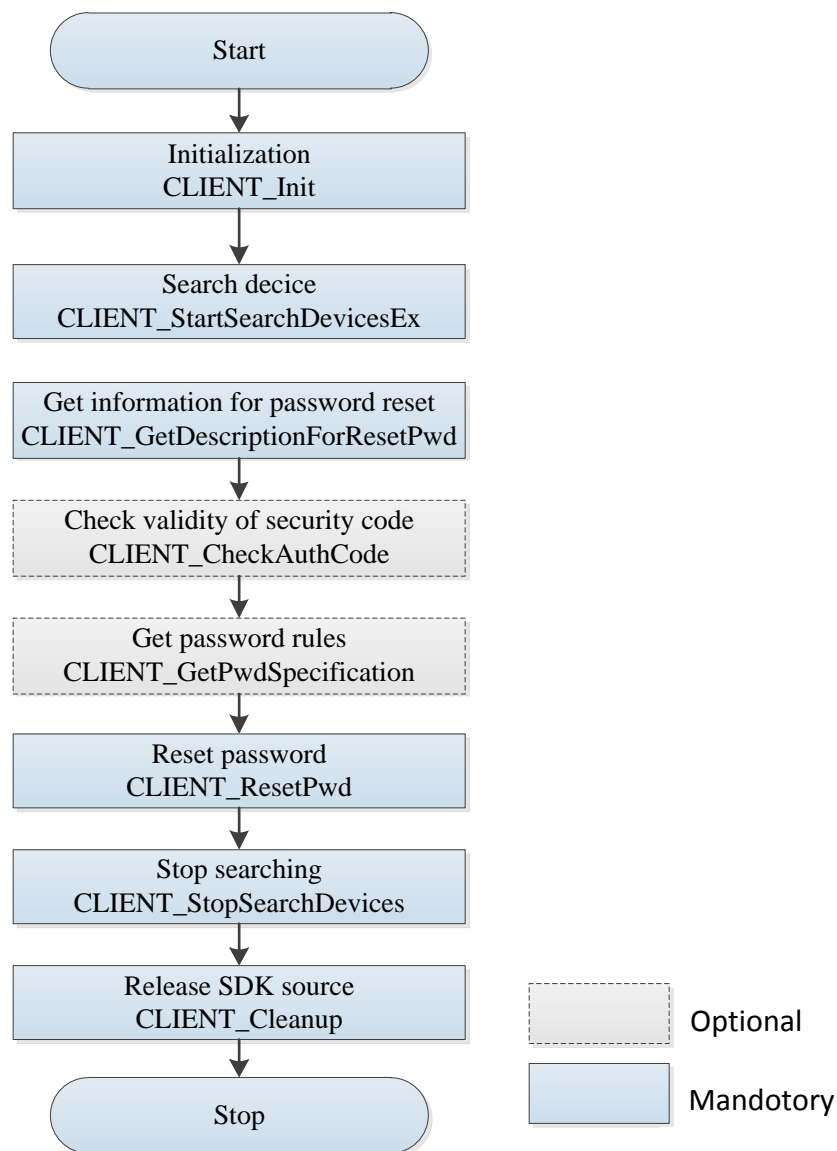
- Step 4 Call **CLIENT_InitDevAccount** to initialize device.
- Step 5 Call **CLIENT_StopSearchDevices** to stop searching.
- Step 6 Call **CLIENT_LoginWithHighLevelSecurity** and login the admin account with the configured password.
- Step 7 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 8 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

Because the interface is working in multicast, the host PC and device must be in the same multicast group.

2.2.3.2 Password Reset

Figure 2-3 Process of device initialization



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.

Step 2 Call **CLIENT_StartSearchDevicesEx** to search the devices within the LAN and get the device information.



Multi-thread calling is not supported.

Step 3 Call **CLIENT_GetDescriptionForResetPwd** to get the information for password reset.

Step 4 (Optional) Scan the QR code obtained from the previous step to get the security code, and then validate it through **CLIENT_CheckAuthCode**.

Step 5 (Optional) Call **CLIENT_GetPwdSpecification** to get the password rules.

Step 6 Call **CLIENT_ResetPwd** to reset the password.

Step 7 Call **CLIENT_StopSearchDevices** to stop searching.

Step 8 Call **CLIENT_LoginWithHighLevelSecurity** and login the admin account with the configured password.

Step 9 After using the function module, call **CLIENT_Logout** to logout the device.

Step 10 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

Because the interface is working in multicast, the host PC and device must be in the same multicast group.

2.2.4 Example Code

2.2.4.1 Device Initialization

```
//Firstly, call CLIENT_StartSearchDevicesEx to get the device information.
//Get the password rules
NET_IN_PWD_SPECI stIn = {sizeof(stIn)};
strncpy(stIn.szMac, szMac, sizeof(stIn.szMac) - 1);
NET_OUT_PWD_SPECI stOut = {sizeof(stOut)};
CLIENT_GetPwdSpecification(&stIn, &stOut, 3000, NULL);//In the case of single network card, the last
parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter.
Set the password according to the rules which are used for preventing user from setting the passwords that
are not supported by the device.

//Device initialization
NET_IN_INIT_DEVICE_ACCOUNT sInitAccountIn = {sizeof(sInitAccountIn)};
NET_OUT_INIT_DEVICE_ACCOUNT sInitAccountOut = {sizeof(sInitAccountOut)};
sInitAccountIn.byPwdResetWay = 1;//1 stands for password reset by mobile phone number, and 2 stands for
password reset by email
strncpy(sInitAccountIn.szMac, szMac, sizeof(sInitAccountIn.szMac) - 1);//Set mac value
strncpy(sInitAccountIn.szUserName, szUserName, sizeof(sInitAccountIn.szUserName) - 1);//Set user name
strncpy(sInitAccountIn.szPwd, szPwd, sizeof(sInitAccountIn.szPwd) - 1);//Set password
```

```

strncpy(slnitAccountIn.szCellPhone, szRig, sizeof(slnitAccountIn.szCellPhone) - 1); //If the byPwdResetWay is set
as 1, please set szCellPhone field; if the byPwdResetWay is set as 2, please set slnitAccountIn.szMail field.
CLIENT_InitDevAccount(&slnitAccountIn, &slnitAccountOut, 5000, NULL);

```

2.2.4.2 Password Reset

```

//Firstly, call CLIENT_StartSearchDevicesEx to get the device information.
//Get the information for password reset
NET_IN_DESCRIPTION_FOR_RESET_PWD stIn = {sizeof(stIn)};
strncpy(stIn.szMac, szMac, sizeof(stIn.szMac) - 1); //Set mac value
strncpy(stIn.szUserName, szUserName, sizeof(stIn.szUserName) - 1); //Set user name
stIn.byInitStatus = bStstus; //bStstus is the value of return field byInitStatus of device search interface (Callback
of CLIENT_SearchDevices, CLIENT_StartSearchDevice and CLIENT_StartSearchDevicesEx, and
CLIENT_SearchDevicesByIPs)
NET_OUT_DESCRIPTION_FOR_RESET_PWD stOut = {sizeof(stOut)};
char szTemp[360];
stOut.pQrCode = szTemp;
CLIENT_GetDescriptionForResetPwd(&stIn, &stOut, 3000, NULL); //In the case of single network card, the last
parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter.
After successful connection, stout will output a QR code with address of stOut.pQrCode. Scan this QR code to
get the security code for password reset. This security code will be sent to the reserved mobile phone or email
box.
//(Optional) Check the security code
NET_IN_CHECK_AUTHCODE stIn1 = {sizeof(stIn1)};
strncpy(stIn1.szMac, szMac, sizeof(stIn1.szMac) - 1); //Set mac value
strncpy(stIn1.szSecurity, szSecu, sizeof(stIn1.szSecurity) - 1); // szSecu is the security code sent to the reserved
mobile phone or email box
NET_OUT_CHECK_AUTHCODE stOut1 = {sizeof(stOut1)};
bRet = CLIENT_CheckAuthCode(&stIn1, &stOut1, 3000, NULL); //In the case of single network card, the last
parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter
//Get password rules
NET_IN_PWD_SPECI stIn2 = {sizeof(stIn2)};
strncpy(stIn2.szMac, szMac, sizeof(stIn2.szMac) - 1); //Set mac value
NET_OUT_PWD_SPECI stOut2 = {sizeof(stOut2)};
CLIENT_GetPwdSpecification(&stIn2, &stOut2, 3000, NULL); // In the case of single network card, the last
parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter.
Set the password according to the rules which are used for preventing user from setting the passwords that
are not supported by the device
//Reset password
NET_IN_RESET_PWD stIn3 = {sizeof(stIn3)};
strncpy(stIn3.szMac, szMac, sizeof(stIn3.szMac) - 1); //Set mac value
strncpy(stIn3.szUserName, szUserName, sizeof(stIn3.szUserName) - 1); //Set user name

```



```

strncpy(stln3.szPwd, szPassWd, sizeof(stln3.szPwd) - 1); //szPassWd is the password reset according to the rules
strncpy(stln3.szSecurity, szSecu, sizeof(stln1.szSecurity) - 1); //szSecu is the security code sent to the reserved
mobile phone or email box
stln3.byInitStaus = bStstus; //bStstus is the value of return field byInitStatus of device search interface
(Callback of CLIENT_SearchDevices, CLIENT_StartSearchDevice and CLIENT_StartSearchDevicesEx, and
CLIENT_SearchDevicesByIPs)
stln3.byPwdResetWay = bPwdResetWay; // bPwdResetWay is the value of return field byPwdResetWay of
device search interface (Callback of CLIENT_SearchDevices and CLIENT_StartSearchDevice, and
CLIENT_SearchDevicesByIPs)
NET_OUT_RESET_PWD stOut3 = {sizeof(stOut3)};
CLIENT_ResetPwd(&stln3, &stOut3, 3000, NULL); //In the case of single network card, the last parameter can be
left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter.

```

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 login to the device and should call login ID before using other SDK interfaces. The login ID becomes invalid once logged out.

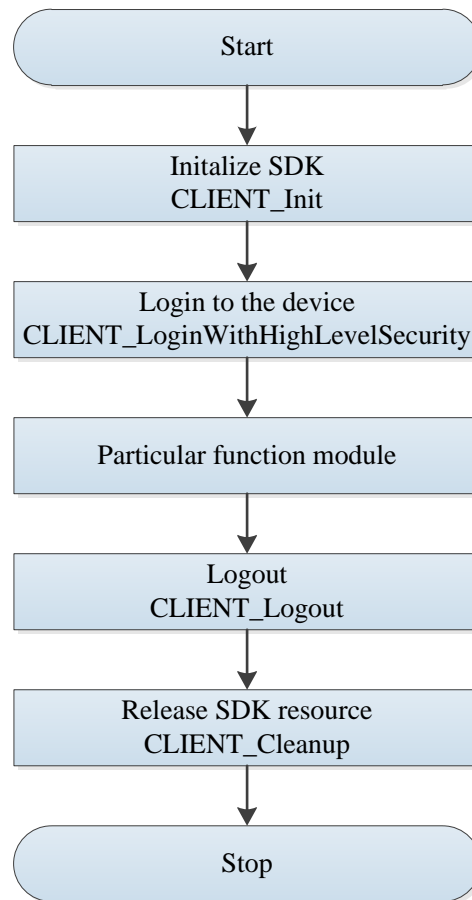
2.3.2 Interface Overview

Table 2-3 Interfaces of device login

Interface	Implication
CLIENT_LoginWithHighLevelSecurity	Log in to the device with high level security. CLIENT_LoginEx2 can still be used, but there are security risks, so it is highly recommended to use the interface CLIENT_LoginWithHighLevelSecurity to log in to the device.
CLIENT_Logout	Logout.

2.3.3 Process

Figure 2-4 Process of device login



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call CLIENT_LoginWithHighLevelSecurity to login the device.
- Step 3 After successful login, you can realize the required function module.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- 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 handle at each login. If there is not special function module, it is suggested to login only one time. The login handle can be repeatedly used on other function modules.
- Logout: The interface will release the opened functions internally, but it is not suggested to rely on the cleaning up function. For example, if you opened the monitoring function, you should call the interface that stops the monitoring function when it is no longer required.
- Use login and logout in pairs: The login consumes some memory and socket information and release sources once logout.
- Login failure: It is suggested to check the failure through the error parameter of the login interface.

Table 2-4 Common error code

Interface	Implication
1	Password is wrong
2	User name does not exist
3	Login timeout
4	The account has been logged in
5	The account has been locked
6	The account is blacklisted
7	Out of resources, the system is busy
8	Sub connection failed
9	Main connection failed
10	Exceeded the maximum user connections
11	Lack of avnetsdk or avnetsdk dependent library
12	USB flash disk is not inserted into device, or the USB flash disk information error
13	The client IP is not authorized with login

For more information about error codes, see "CLIENT_LoginWithHighLevelSecurity interface" in *Network SDK Development Manual.chm*. The example code to avoid error code 3 is as follows.

```
NET_PARAM stuNetParam = {0};
stuNetParam.nWaittime = 8000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

2.3.4 Example Code

```
NET_DEVICEINFO_Ex stDevInfo = {0};
int nError = 0;
// Login the device
NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY stInparam;
memset(&stInparam, 0, sizeof(stInparam));
stInparam.dwSize = sizeof(stInparam);
strncpy(stInparam.szIP, csIp.GetBuffer(0), sizeof(stInparam.szIP) - 1);
strncpy(stInparam.szPassword, csPwd.GetBuffer(0), sizeof(stInparam.szPassword) - 1);
strncpy(stInparam.szUserName, csName.GetBuffer(0), sizeof(stInparam.szUserName) - 1);
stInparam.nPort = sPort;
stInparam.emSpecCap = EM_LOGIN_SPEC_CAP_TCP;
NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY stOutparam;
memset(&stOutparam, 0, sizeof(stOutparam));
stOutparam.dwSize = sizeof(stOutparam);

LLONG lLoginHandle = CLIENT_LoginWithHighLevelSecurity(&stInparam, &stOutparam);

// Logout the device
if (0 != lLoginHandle)
{
```

```
CLIENT_Logout(ILLoginHandle);
```

```
}
```

2.4 Real-time Monitoring

2.4.1 Introduction

Real-time monitoring 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 2-5 Interfaces of real-time monitoring

Interface	Implication
CLIENT_RealPlayEx	Start real-time monitoring extension interface.
CLIENT_StopRealPlayEx	Stop real-time monitoring extension interface.
CLIENT_SaveRealData	Start saving the real-time monitoring data to the local path.
CLIENT_StopSaveRealData	Stop saving the real-time monitoring data to the local path.
CLIENT_SetRealDataCallBackEx2	Set real-time monitoring data callback function extension interface.

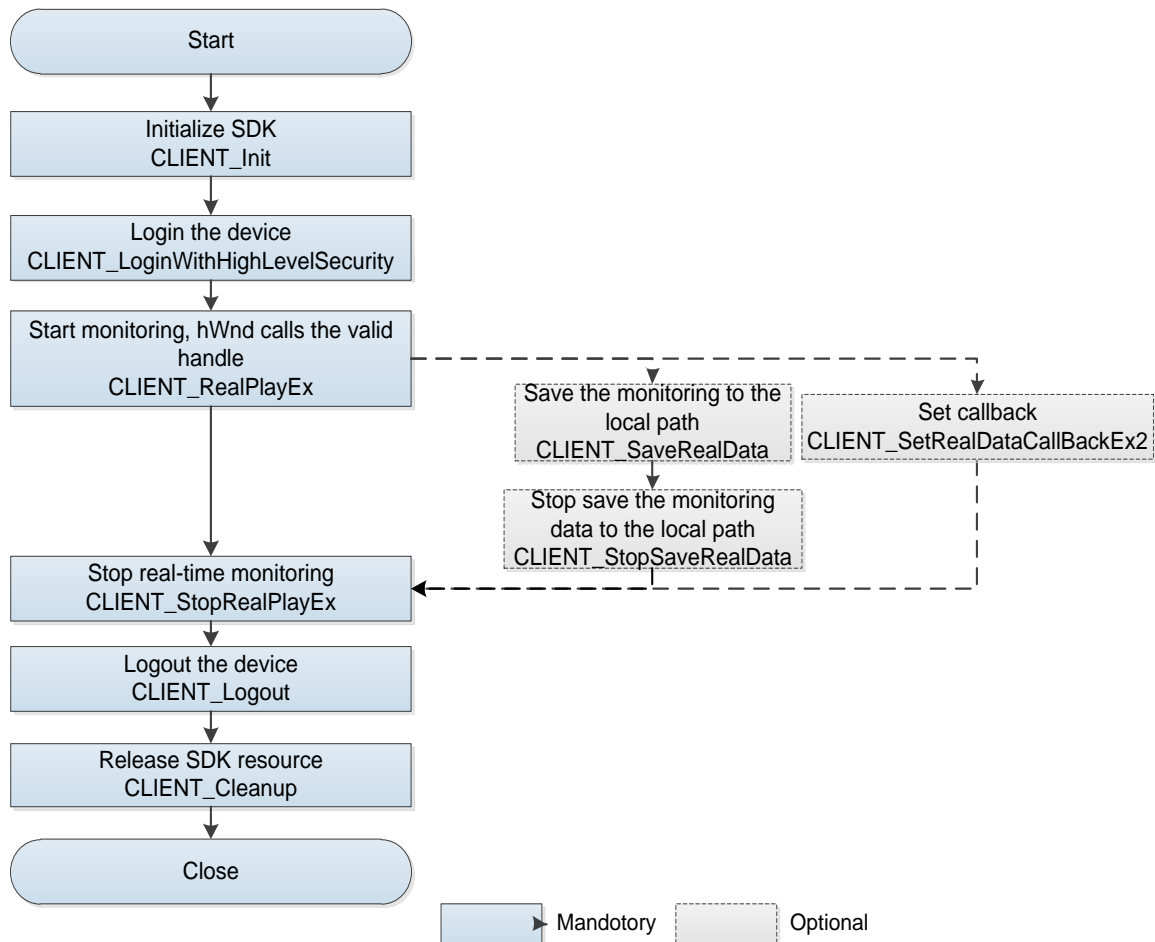
2.4.3 Process

You can realize the real-time monitoring through SDK decoding library or your play library.

2.4.3.1 SDK Decoding Play

Call PlaySDK library from the SDK auxiliary library to realize real-time play.

Figure 2-5 Process of playing by SDK decoding library



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_RealPlayEx** to enable the real-time monitoring. The parameter hWnd is a valid window handle.
- Step 4 (Optional) Call **CLIENT_SaveRealData** to start saving the monitoring data.
- Step 5 (Optional) Call **CLIENT_StopSaveRealData** to end the saving process and generate the local video file.
- Step 6 (Optional) If you call **CLIENT_SetRealDataCallBackEx2**, you can choose to save or forward the video file. If saved as the video file, see the step 4 and step 5.
- Step 7 After using the real-time function, call **CLIENT_StopRealPlayEx** to stop real-time monitoring.
- Step 8 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 9 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- SDK decoding play only supports Windows system. You need to call the decoding after getting the stream in other systems.

- Multi-thread calling: Multi-thread calling is not supported for the functions within the same login session; however, multi-thread calling can deal with the functions of different login sessions although such calling is not recommended.
- Timeout: The request on applying for monitoring resources should have made some agreement with the device before requiring the monitoring data. There are some timeout settings (see "NET_PARAM structure"), and the field about monitoring is nGetConnInfoTime. If there is timeout due to the reasons such as bad network connection, you can modify the value of nGetConnInfoTime bigger.

The example code is as follows. Call it for only one time after having called **CLIENT_Init**.

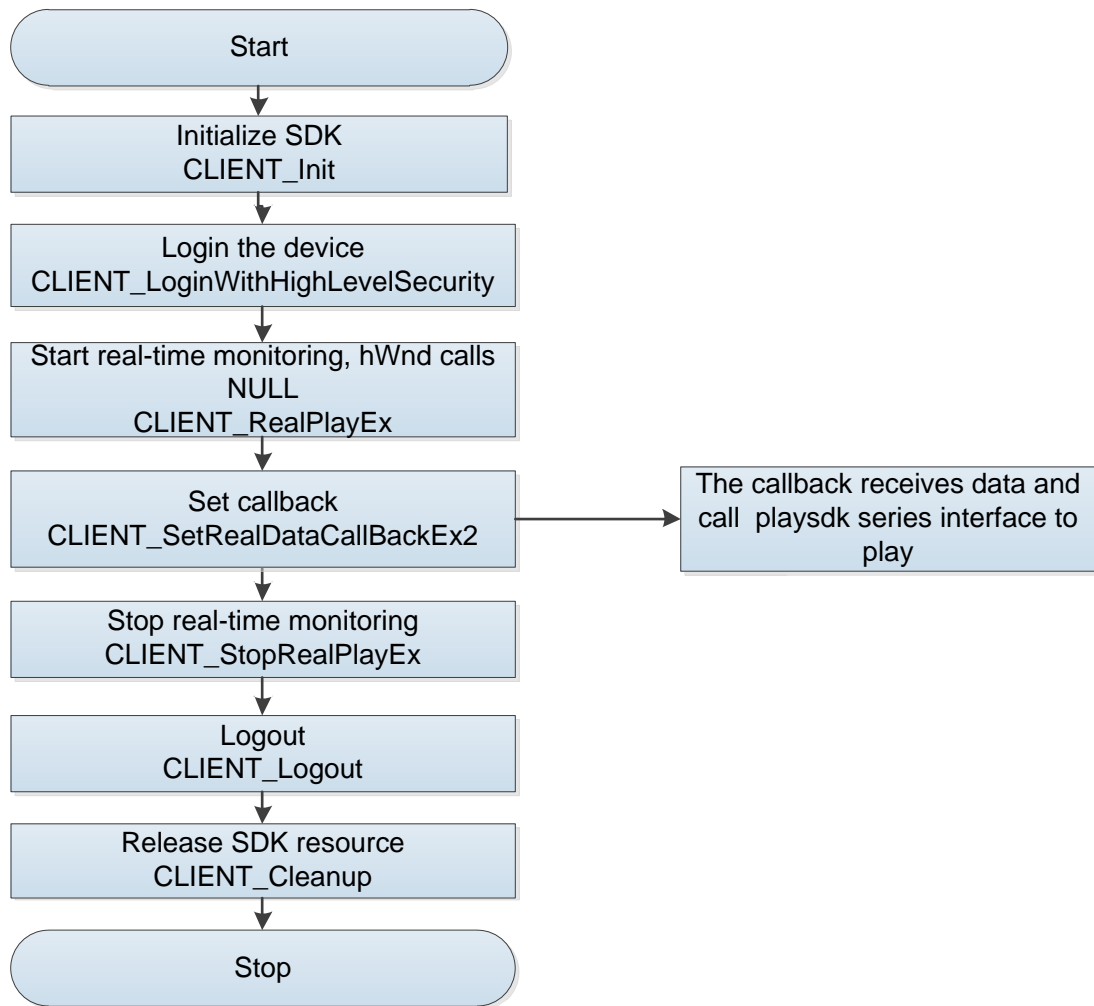
```
NET_PARAM stuNetParam = {0};
stuNetParam.nGetConnInfoTime = 5000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- Failed to repeat opening: For some models, the same channel cannot be opened for multiple times during the one entire logged in status. If you are trying to open it repeatedly, you will success in the first try but get failed afterwards. In this case, you can try the following:
 - ◇ Close the opened channel. For example, if you already opened the main stream video on the channel 1 and still want to open the sub stream video on the same channel, you can close the main stream first and then open the sub stream.
 - ◇ Login twice to obtain two login handles to deal with the main stream and sub stream respectively.
- Calling succeeded but no image: SDK decoding needs to use dhplay.dll. It is suggested to check if dhplay.dll and its auxiliary library are missing under the running directory. See Table 1-1.
- If the system resource is insufficient, the device might return error instead of stream. You can receive an event DH_REALPLAY_FAILED_EVENT in the alarm callback that is set in CLIENT_SetDVRMessCallBack. This event includes the detailed error codes. See "DEV_PLAY_RESULT Structure" in *Network SDK Development Manual.chm*.
- 32 channels limit: The decoding consumes resources especially for the high definition videos. Considering the limited resources at the client, currently the maximum channels are set to be 32. If more than 32, it is suggested to use third party play library. See "2.4.3.2 Call Third Party Play Library."

2.4.3.2 Call Third Party Play Library

SDK calls back the real-time monitoring stream to you and you call PlaySDK to decode and play.

Figure 2-6 Process of calling the third party play library



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 After successful login, call **CLIENT_RealPlayEx** to enable real-time monitoring. The parameter hWnd is NULL.
- Step 4 Call **CLIENT_SetRealDataCallBackEx2** to set the real-time data callback.
- Step 5 In the callback, pass the data to PlaySDK to finish decoding.
- Step 6 After completing the real-time monitoring, call **CLIENT_StopRealPlayEx** to stop real-time monitoring.
- Step 7 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 8 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Stream format: It is recommended to use PlaySDK for decoding.
- Lag image
 - ◇ When using PlaySDK for decoding, there is a default channel cache size (the **PLAY_OpenStream** interface in playsdk) for decoding. If the stream resolution value is big, it is recommended to modify the parameter value smaller such as 3 M.

- ◇ SDK callbacks can only moves into the next process after returning from you. It is not recommended for you to consume time for the unnecessary operations; otherwise the performance could be affected.

2.4.4 Example Code

2.4.4.1 SDK Decoding Play

```
// Take opening the main stream monitoring of channel 1 as an example. The parameter hWnd is a handle of
interface window.
LLONG IRealHandle = CLIENT_RealPlayEx(ILoginHandle, 0, hWnd, DH_RType_Realplay);
if (NULL == IRealHandle)
{
    printf("CLIENT_RealPlayEx: failed! Error code: %x.\n", CLIENT_GetLastError());
}
printf("input any key to quit!\n");
getchar();
// Stop preview
if (NULL != IRealHandle)
{
    CLIENT_StopRealPlayEx(IRealHandle);
}
```

2.4.4.2 Call Play Library

```
void CALLBACK RealDataCallBackEx(LLONG IRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD
dwBufSize, LLONG param, LDWORD dwUser);
// Take opening the main stream monitoring of channel 1 as an example.
LLONG IRealHandle = CLIENT_RealPlayEx(ILoginHandle, 0, NULL, DH_RType_Realplay);
if (NULL == IRealHandle)
{
    printf("CLIENT_RealPlayEx: failed! Error code: %x.\n", CLIENT_GetLastError());
}
else
{
    DWORD dwFlag = REALDATA_FLAG_RAW_DATA; //Initial data labels
    CLIENT_SetRealDataCallBackEx2(IRealHandle, &RealDataCallBackEx, NULL, dwFlag);
}

printf("input any key to quit!\n");
getchar();
```



```
// Stop preview
if (0 != IRealHandle)
{
    CLIENT_StopRealPlayEx(IRealHandle);
}

void CALLBACK RealDataCallBackEx(LLONG IRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD
dwBufSize, LLONG param, LDWORD dwUser)
{
    // Call PlaySDK interface to get the stream data from the device. See SDK monitoring demo source data for
    more details.

    printf("receive real data, param: IRealHandle[%p], dwDataType[%d], pBuffer[%p], dwBufSize[%d]\n",
    IRealHandle, dwDataType, pBuffer, dwBufSize);
}
```

2.5 Video Snapshot

2.5.1 Introduction

Video snapshot can get the picture data of the playing video. This section introduces the following two snapshot ways:

- Network snapshot: Call the SDK interface which sends the snapshot command to the device. The device will snapshot the current image and send to SDK through network, and then SDK returns the image data to you.
- Local snapshot: When the monitoring is opened, you can save the monitoring data to the picture format which is the frame information that does not have interaction with the device.

2.5.2 Interface Overview

Table 2-6 Interfaces of video snapshot

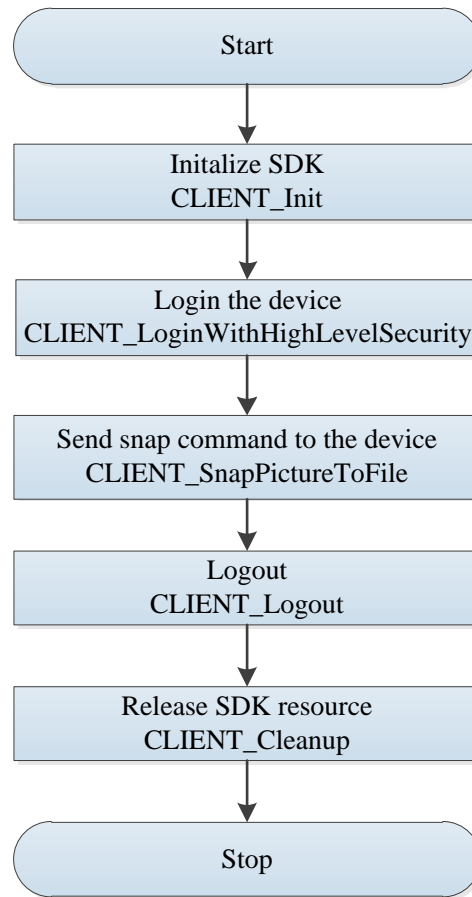
Interface	Implication
CLIENT_SnapPictureToFile	Snap picture and send to the user.
CLIENT_CapturePictureEx	Snap local pictures and the parameters could be monitoring handle or playback handle.

2.5.3 Process

Video snapshot is consisted of network snapshot and local snapshot.

2.5.3.1 Network Snapshot

Figure 2-7 Process of network snapshot



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_SnapPictureToFile** to get the picture data.
- Step 4 Call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Picture size limit: SDK allocates the fixed memory to receive the picture data returned from the device. If the picture is larger than the fixed memory, SDK will return the truncated data.
- SDK provides the interface to modify the default memory. If the picture (for example, the high definition picture) is truncated, you can modify the value of nPicBufSize bigger. The example code is as follows. After calling **CLIENT_Init**, call the example code just one time.

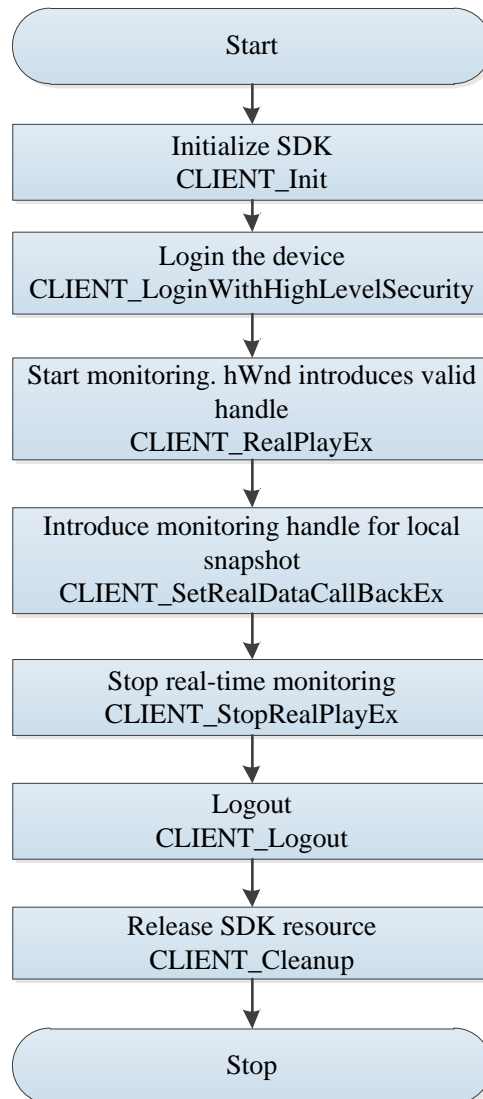
```
NET_PARAM stuNetParam = {0};  
stuNetParam.nPicBufSize = 4*1024*1024; // unit byte  
CLIENT_SetNetworkParam (&stuNetParam);
```

- Multi-thread calling: Multi-thread calling is not supported for the functions within the same login session.

- Snapshot configuration: You can configure the items such as quality and definition for the snapshot. However, if the default configurations are satisfactory, do not modify them. For more details of the example code, see the SDK package on the website
- Picture save format: The picture data returns as memory and the interface supports saving it as file (the precondition is that you have set the `szFilePath` field of `NET_IN_SNAP_PIC_TO_FILE_PARAM`).

2.5.3.2 Local Snapshot

Figure 2-8 Process of local snapshot



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_RealPlayEx** to start monitoring and obtain the monitoring handle.
- Step 4 Call **CLIENT_CapturePictureEx** to introduce the monitoring handle.
- Step 5 Call **CLIENT_StopRealPlayEx** to stop the real-time monitoring.
- Step 6 Call **CLIENT_Logout** to logout the device.
- Step 7 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

2.5.4 Example Code

```
// Network snapshot example
NET_IN_SNAP_PIC_TO_FILE_PARAM stuInParam = {sizeof(stuInParam)};
NET_OUT_SNAP_PIC_TO_FILE_PARAM stuOutParam = {sizeof(stuOutParam)};
SNAP_PARAMS stuSnapParams = {0};
stuSnapParams.Channel = 0;    // Take the first channel as an example
int nBufferLen = 2*1024*1024;
char* pBuffer = new char[nBufferLen]; // Picture cache
memset(pBuffer, 0, nBufferLen);
stuOutParam.szPicBuf = pBuffer;
stuOutParam.dwPicBufLen = nBufferLen;
if (FALSE == CLIENT_SnapPictureToFile(ILoginHandle, &stuSnapParams))
{
    printf("CLIENT_SnapPictureEx Failed!Last Error[%x]\n", CLIENT_GetLastError());
}
delete[] pBuffer;

// Example of local snapshot. The handle hPlayHandle is obtained from opening monitoring.
if (FALSE == CLIENT_CapturePictureEx(hPlayHandle, "test.jpg", NET_CAPTURE_JPEG))
{
    printf("CLIENT_CapturePictureEx Failed!Last Error[%x]\n", CLIENT_GetLastError());
}
```

2.6 PTZ Control

2.6.1 Introduction

PTZ is a mechanical platform that carries the device and the protective enclosure and performs remote control in all directions.

PTZ is consisted of two motors that can perform horizontal and vertical movement to provide the all-around vision.

This section provides guidance for you about how to control directions (there are eight directions: upper, lower, left, right, upper left, upper right, bottom left, and bottom right), focus, zoom, iris, fast positioning, and 3-dimensional positioning through SDK.

2.6.2 Interface Overview

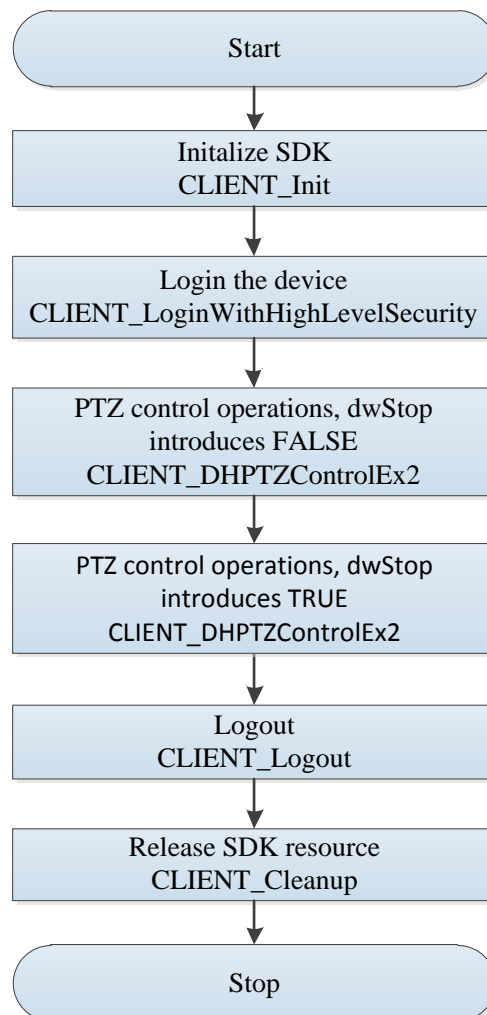
Table 2-7 Interface of PTZ control

Interface	Implication
CLIENT_DHPTZControlEx2	PTZ control extension interface

2.6.3 Process

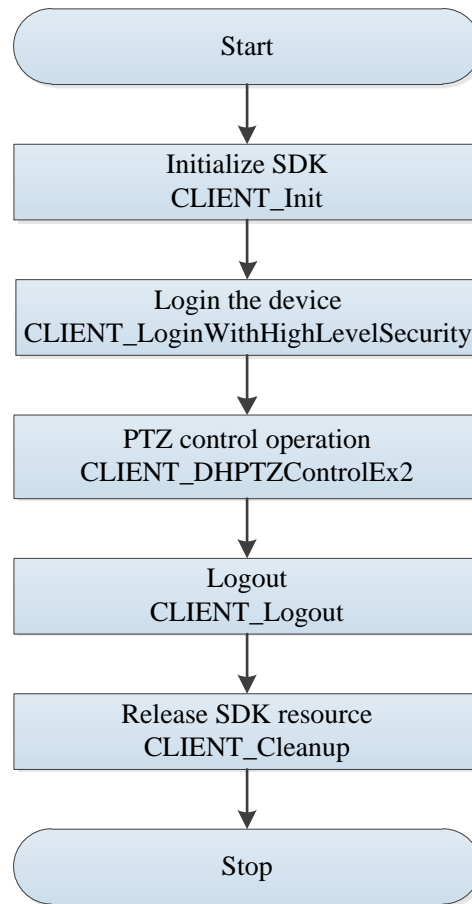
Direction control, focus, zoom and iris are the continuous operations. SDK provides start and stop interfaces for you for timing control.

Figure 2-9 Process of PTZ control (continuous)



Both fast positioning and 3-dimensional positioning belong to one-time action, which needs to call the PTZ control interface just one time.

Figure 2-10 Process of PTZ control (one-time)



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_DHPTZControlEx2** to operate the PTZ according to the situation. Different PTZ command might need different parameters, and part of commands need to call the corresponding stop command, such as moving left and moving right. For details, see "2.6.4 Example Code."
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Fast positioning: For the SD, take the current monitoring image center as origin, and the valid range of horizontal and vertical coordinates is $[-8191, 8191]$. For example, if the horizontal coordinate is 2000 and the vertical is 2000, the SD moves toward upper right and gets a new origin, which means the coordinate specified every time is only relative to the current location.
- 3-dimensional positioning: For the SD, there is an initial position first. The horizontal coordinate is $[0, 3600]$ and the vertical is $[-1800, 1800]$. The coordinate specified each time is the absolute coordinate and is irrelevant to the location of the SD image last time.
- For more example code see the SDK package on the website.

2.6.4 Example Code

```
LONG IParam1 = 0; // Rotating speed in horizontal direction.
LONG IParam2 = 4; // Rotating speed in vertical direction.
LONG IParam3 = 0;
// Continuous operation: Take moving upward as example.
// Start moving upward.
BOOL bRet = CLIENT_DHPTZControlEx2(ILoginHandle, nChannelId, DH_PTZ_UP_CONTROL, IParam1, IParam2,
                                   IParam3, FALSE, NULL);

// Stop moving upward.
bRet = CLIENT_DHPTZControlEx2(ILoginHandle, nChannelId, DH_PTZ_UP_CONTROL, IParam1, IParam2,
                              IParam3, TRUE, NULL);

// One-time operation movement: take fast positioning as an example.
IParam1 = 2000; // Horizontal coordinate, valid range[-8191,8191]
IParam2 = 2000; // Vertical coordinate, valid range [-8191,8191]
IParam3 = 1;    // Zoom, valid range (-16-16),1 indicates rotating without zooming
bRet = CLIENT_DHPTZControlEx2(ILoginHandle, nChannelId, DH_EXTPTZ_FASTGOTO, IParam1, IParam2,
                              IParam3, FALSE, NULL);
```

2.7 Voice Talk

2.7.1 Introduction

Voice talk realizes the voice interaction between the local platform and the environment where front-end devices are located.

This section introduces how to use SDK to realize the voice talk with the front-end devices.

2.7.2 Interface Overview

Table 2-8 Interfaces of voice talk

Interface	Implication
CLIENT_StartTalkEx	Start voice talk
CLIENT_StopTalkEx	Stop voice talk
CLIENT_RecordStartEx	Start client record (valid only in Windows system)
CLIENT_RecordStopEx	Stop client record (valid only in Windows system)
CLIENT_TalkSendData	Send voice data to the device
CLIENT_AudioDecEx	Decode audio data (valid only in Windows system)

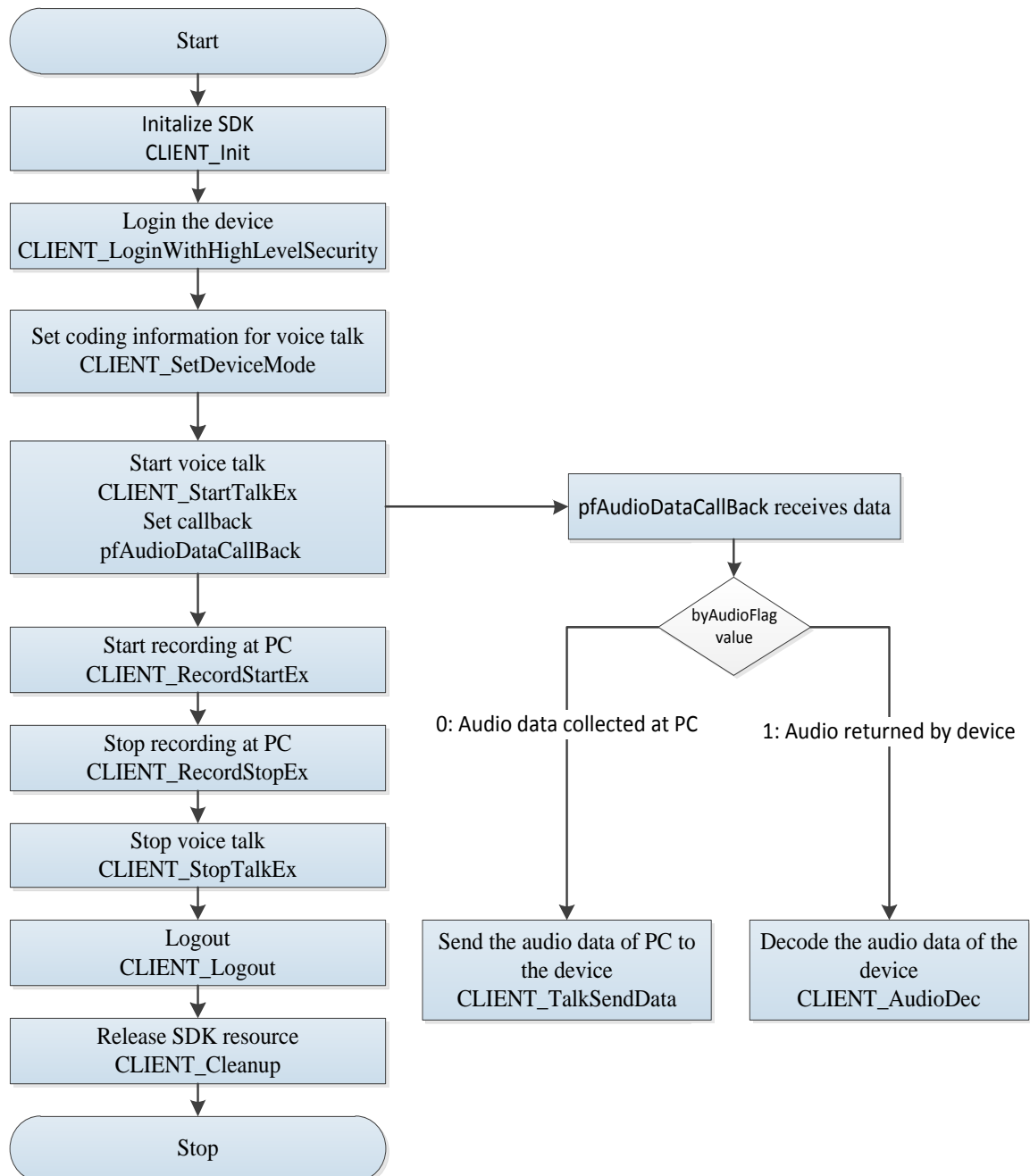
2.7.3 Process

When SDK has collected the audio data from the local audio card, or SDK has received the audio data from the front-end devices, SDK will call the callback of audio data.

You can call the SDK interface in the callback parameters to send the local audio data to the front-end devices, or call SDK interface to decode and play the audio data received from the front-end devices.

This process is valid only in Windows system. For the process of voice talk, see Figure 2-11.

Figure 2-11 Process of voice talk



Process Description

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

Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.

- Step 3 Call **CLIENT_SetDeviceMode** to set decoding information of voice talk. Set parameter emType as DH_TALK_ENCODE_TYPE.
- Step 4 Call **CLIENT_StartTalkEx** to set callback and start voice talk. In the callback, call **CLIENT_AudioDec** to decode the audio data that is sent from the decoding device, and call **CLIENT_TalkSendData** to send the audio data of the PC end to the device.
- Step 5 Call **CLIENT_RecordStartEx** to start recording at PC. After this interface is called, the voice talk callback in CLIENT_StartTalkEx will receive the local audio data.
- Step 6 After using the voice talk function, call **CLIENT_RecordStopEx** to stop recording.
- Step 7 Call **CLIENT_StopTalkEx** to stop voice talk.
- Step 8 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 9 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Voice encoding format: The example uses the common PCM format. SDK supports accessing the voice encoding format supported by the device. For more details of the example code, see the SDK package on the website. If the default PCM can satisfy the requirement, it is not recommended to obtain the voice encoding format from the device.
- No sound at the device: The audio data needs to be collected by the device such as microphone. It is recommended to check if the microphone or other equivalent device is plugged in and if the CLIENT_RecordStartEx succeeded in returning.

2.7.4 Example Code

```
// Set the voice talk encoding data. Take PCM as an example.
DHDEV_TALKDECODE_INFO curTalkMode;
curTalkMode.encodeType = DH_TALK_PCM;
curTalkMode.nAudioBit = 16;
curTalkMode.dwSampleRate = 8000;
curTalkMode.nPacketPeriod = 25;
CLIENT_SetDeviceMode(ILoginHandle, DH_TALK_ENCODE_TYPE, &curTalkMode);
// Start voice talk
ITalkHandle = CLIENT_StartTalkEx(ILoginHandle, AudioDataCallBack, (LDWORD)NULL);
if(0 != ITalkHandle)
{
    BOOL bSuccess = CLIENT_RecordStartEx(ILoginHandle);
}

// Stop local recording
if (!CLIENT_RecordStopEx(ILoginHandle))
{
    printf("CLIENT_RecordStop Failed!Last Error[%x]\n", CLIENT_GetLastError());
}

// Stop voice talk
```

```

if (0 != ITalkHandle)
{
    CLIENT_StopTalkEx(ITalkHandle);
}

void CALLBACK AudioDataCallBack(LLONG ITalkHandle, char *pDataBuf, DWORD dwBufSize, BYTE byAudioFlag,
LDWORD dwUser)
{
    if(0 == byAudioFlag)
    {
        // Send the sound data detected by the PC to the device
        LONG ISendLen = CLIENT_TalkSendData(ITalkHandle, pDataBuf, dwBufSize);
        if(ISendLen != (LONG)dwBufSize)
        {
            printf("CLIENT_TalkSendData Failed!Last Error[%x]\n" , CLIENT_GetLastError());
        }
    }
    else if(1 == byAudioFlag)
    {
        // Send the voice data from the device to SDK for encoding and play.
        CLIENT_AudioDec(pDataBuf, dwBufSize);
    }
}

```

2.8 Heat Map (Temperature)

2.8.1 Introduction

Heat map (temperature) function obtains the temperature distribution data, and the gray map and temperature map.



Only the devices equipped with temperature measurement support this function.

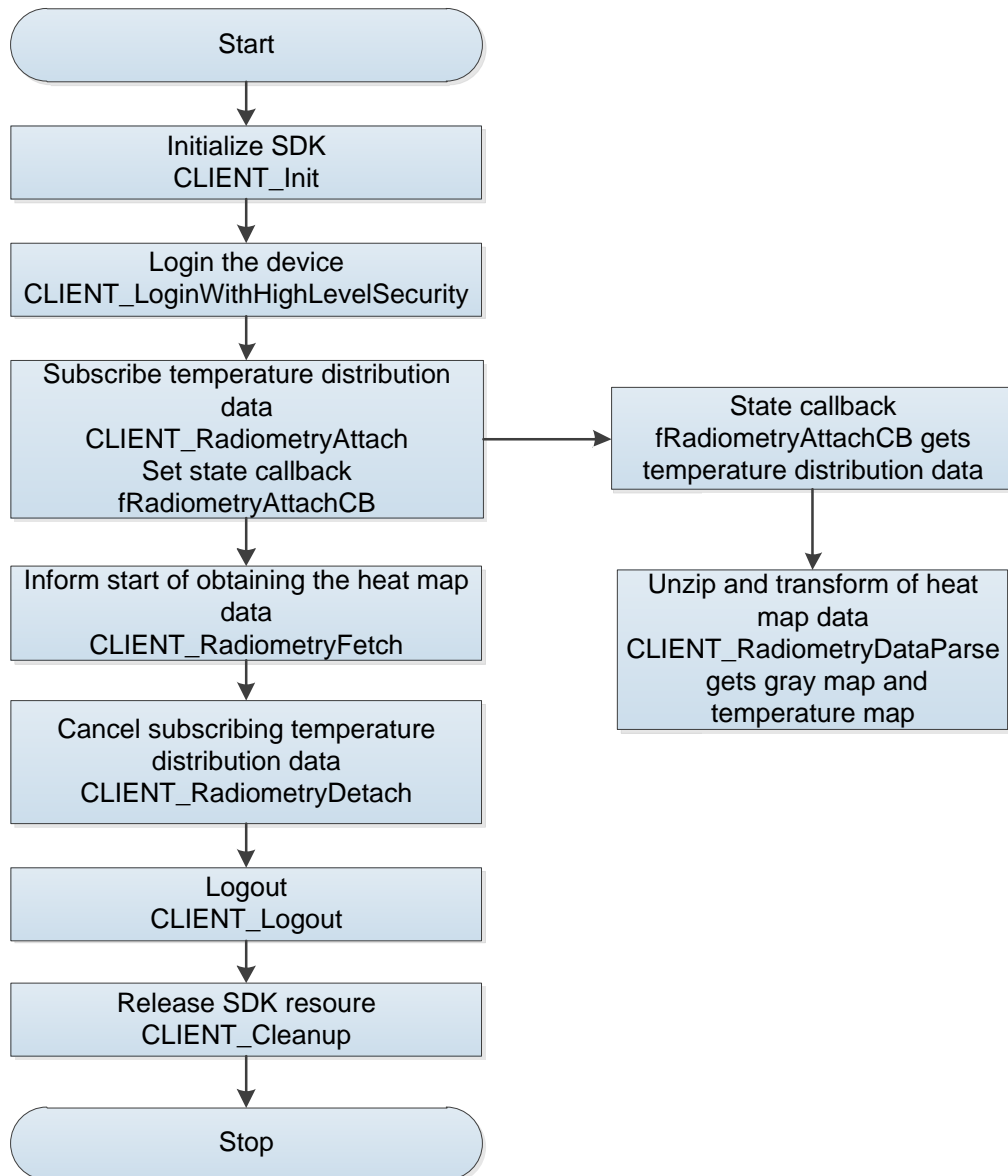
2.8.2 Interface Overview

Table 2-9 Interfaces of heat map (temperature)

Interface	Implication
CLIENT_RadiometryAttach	Subscribe the temperature distribution data (heat map).
CLIENT_RadiometryDetach	Cancel subscribing the temperature distribution data.
CLIENT_RadiometryFetch	Inform start of obtaining the heat map data.
CLIENT_RadiometryDataParse	Unzip the heat map data and convert it to the gray data and temperature data in the unit of pixel.

2.8.3 Process

Figure 2-12 Process of heat map (temperature)



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_RadiometryAttach** to subscribe the temperature distribution data and register status callback. After the device has reported the temperature status, the callback **fRadiometryAttachCB** will inform you.
- Step 4 Call **CLIENT_RadiometryFetch** to inform the device to start obtaining the heat map data. Call this interface whenever you need the heat map data.
- Step 5 After receiving the temperature status information, call **CLIENT_RadiometryDataParse** to get the gray data and temperature data of each pixel.
- Step 6 Call **CLIENT_RadiometryDetach** to cancel subscribing the temperature distribution data after finishing use.
- Step 7 After using the function module, call **CLIENT_Logout** to logout the device.

Step 8 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- No uploaded heat map data: You will get the heat map data only after calling **CLIENT_RadiometryFetch**. If you call **CLIENT_RadiometryDetach** before receiving the heat map data, you will not receive the heat map data.
- Subscribing the channel number: If the camera has only one sensor, the subscription channel number is 0; if the camera has double sensors, the subscription channel is 0 or 1. For the TPC with double sensors, the second channel is TPC.
- The heat map data returned by the interface is not picture and needs to be converted to pictures based on the rules provided by the device.

2.8.4 Example Code

```
// Temperature state callback
void CALLBACK cbRadiometryAttachCB(LLONG lAttachHandle, NET_RADIOMETRY_DATA* pBuf, int nBufLen,
LDWORD dwUser)
{
    int nPixel = pBuf->stMetaData.nWidth*pBuf->stMetaData.nHeight;
    unsigned short *pGray = new unsigned short[nPixel];
    memset(pGray,0,nPixel);

    float *pTemp = new float[nPixel];
    memset(pTemp,0,nPixel);

    CLIENT_RadiometryDataParse(pBuf,pGray,pTemp);
    delete[] pGray;
    delete[] pTemp;
}

// Subscribe the heat map
NET_IN_RADIOMETRY_ATTACH stIn = {sizeof(stIn), 1, cbRadiometryAttachCB};
NET_OUT_RADIOMETRY_ATTACH stOut = {sizeof(stOut)};

LLONG attachHandle = CLIENT_RadiometryAttach(loginId, &stIn, &stOut, 3000);
if (NULL == attachHandle)
{
    // Subscription failed
}

// Inform the device to start collecting the data.
```

```

NET_IN_RADIOMETRY_FETCH stInFetch = {sizeof(stInFetch), 1};
NET_OUT_RADIOMETRY_FETCH stOutFetch = {sizeof(stOutFetch)};

CLIENT_RadiometryFetch(m_ILoginID, &stInFetch, &stOutFetch, 3000);

// Stop subscription after finishing using this function
CLIENT_RadiometryDetach(attachHandle);

```

2.9 Heat Map (Activity)

2.9.1 Introduction

Heat map (activity) function is a picture using different colors to show the statistical activity of an area during a certain period.



The Heat map (activity) here is different from that of temperature in "2.8 Heat Map (Temperature)".

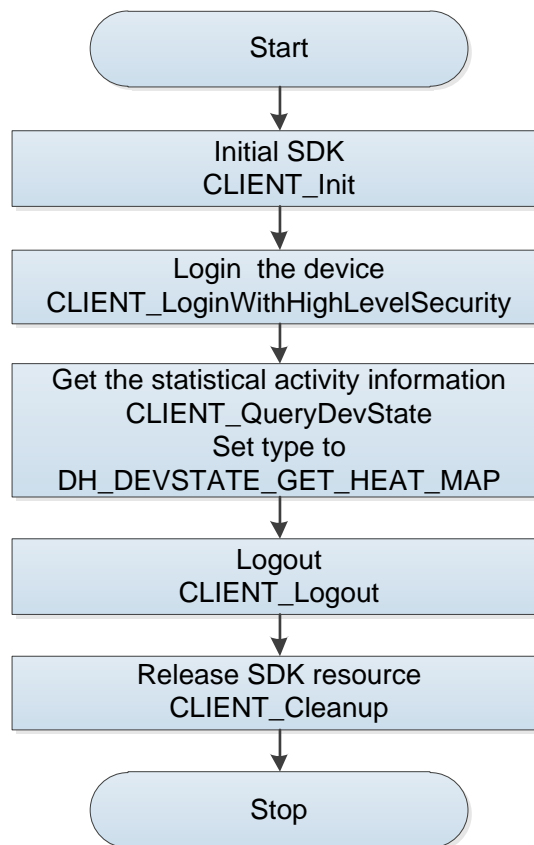
2.9.2 Interface Overview

Table 2-10 Interface of heat map (activity)

Interface	Implication
CLIENT_QueryDevState	Get the statistical activity information.

2.9.3 Process

Figure 2-13 Process of heat map (activity)



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_QueryDevState** to get the statistical activity information, and set the **type** parameter to **DH_DEVSTATE_GET_HEAT_MAP**.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

The heat map data returned by the interface is not picture data and needs to be converted to pictures based on the rules provided by the device.

2.9.4 Example Code

```
int nRetLen = 0;
NET_QUERY_HEAT_MAP stHeatMap = {sizeof(stHeatMap)};
stHeatMap.stuIn.nChannel = 0;
NET_TIME_EX stBegin = {2017,10,1,0,0,0,0};
NET_TIME_EX stEnd = {2017,10,1,1,1,1,0};
```

```

stHeatMap.stuIn.stuBegin = stBegin;
stHeatMap.stuIn.stuEnd = stEnd;
stHeatMap.stuIn.nPlanID = 1;
stHeatMap.stuIn.emDataType = EM_HEAT_PIC_DATA_TYPE_GRAYDATA;// Support gray data by default
stHeatMap.stuOut.nBufLen = 10*1024*1024;
stHeatMap.stuOut.pBufData = new char[stHeatMap.stuOut.nBufLen]; // Apply for memory according to
stHeatMap.stuOut.nBufLen
memset(stHeatMap.stuOut.pBufData, 0, stHeatMap.stuOut.nBufLen);
CLIENT_QueryDevState(g_ILoginHandle, DH_DEVSTATE_GET_HEAT_MAP,(char
*)&stHeatMap,stHeatMap.dwSize, &nRetLen ,3000);

```

2.10 Firepoint

2.10.1 Introduction

Firepont includes three main functions:

- Parameter configuration: Includes firepoint detection mode, master configuration parameters setting, fireponit snapshot and so on.
- Subscribe thermal firepoint alarm event and parse alarm information and callback information.
- Information communication: Get, set, and delete calibration information for firepoint positioning calculation during firepoint alarm event.

2.10.2 Interface Overview

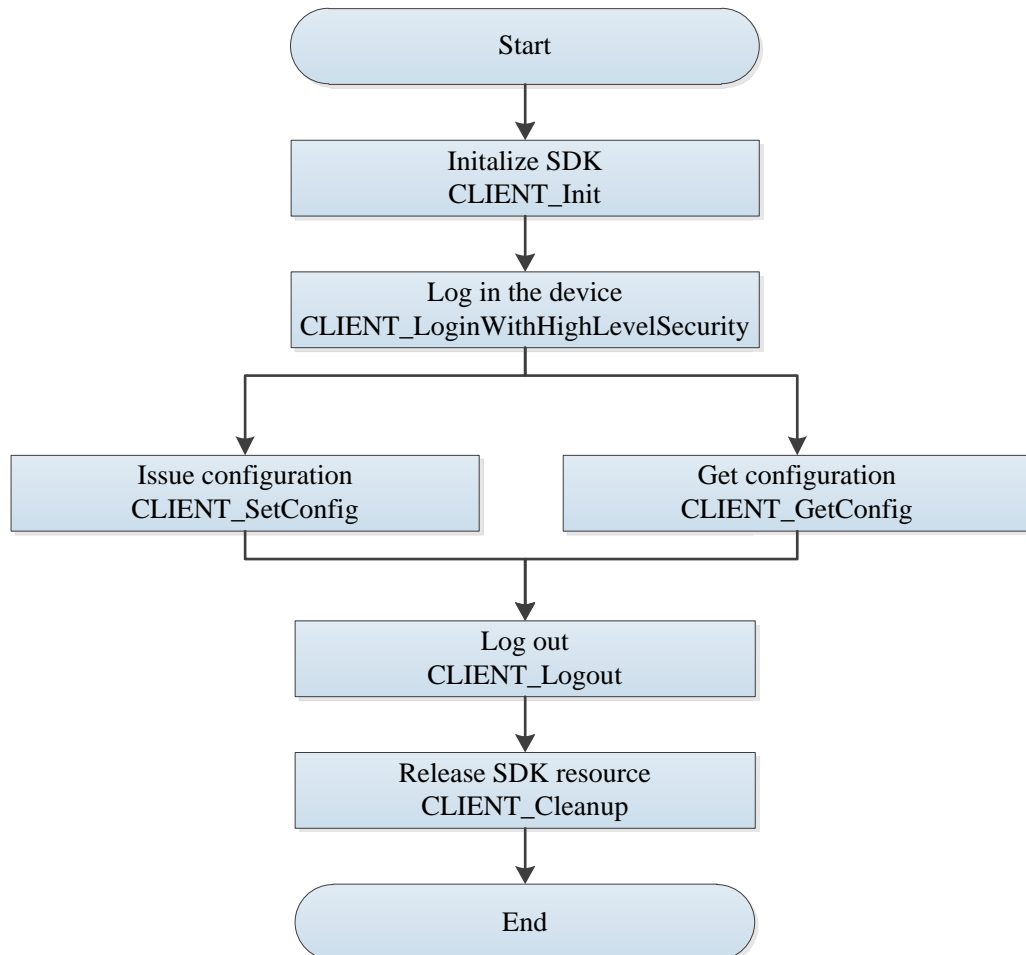
Table 2-11 Interfaces of firepoint

Interface	Implication
CLIENT_SetConfig	Set firepoint detection mode and firepoint master configuration parameters.
CLIENT_GetConfig	Get firepoint detection mode, firepoint master configuration parameters.
CLIENT_SetNewDevConfig	Set fireponit snapshot.
CLIENT_GetNewDevConfig	Get fireponit snapshot.
CLIENT_ParseData	Parse fireponit snapshot.
CLIENT_PacketData	Pack fireponit snapshot.
CLIENT_RealLoadPictureEx	Subscribe firepoint event.
CLIENT_StopLoadPic	Stop subscribing firepoint event.
CLIENT_SetDVRMessCallBack	Set alarm callback function
CLIENT_StartListenEx	Subscribe firepoint alarm.
CLIENT_StopListen	Stop subscribing firepoint alarm.
CLIENT_OperateCalibrateInfo	Get, set, and delete all information of calibration point.

2.10.3 Process

2.10.3.1 Firepoint Detection Mode and Firepoint Master Configuration

Figure 2-14 Process of firepoint detection mode and firepoint master configuration



Process Description

Setting

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call CLIENT_LoginWithHighLevelSecurity to log in the device.
- Step 3 Call **CLIENT_SetConfig** to set firepoint detection mode and firepoint master configuration.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Getting

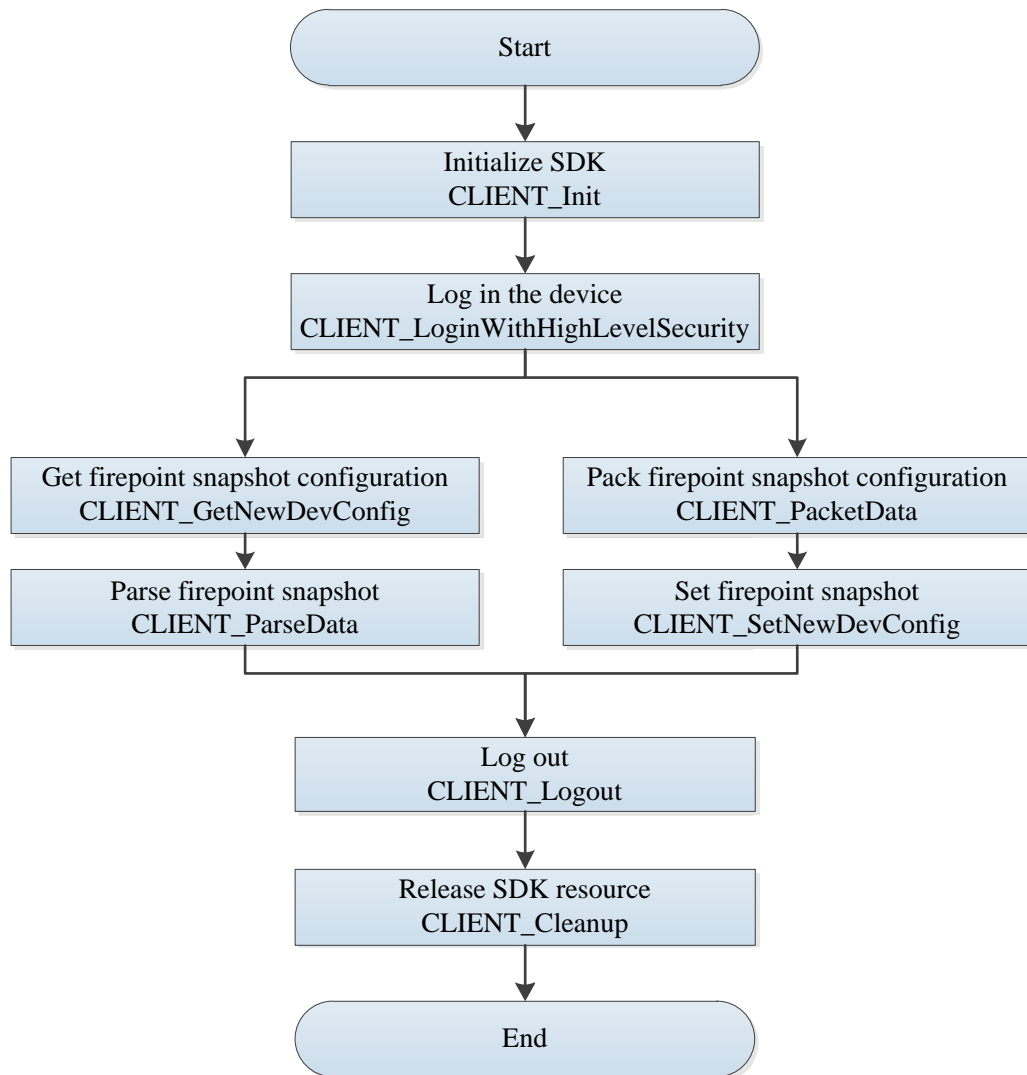
- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call CLIENT_LoginWithHighLevelSecurity to log in the device.
- Step 3 Call **CLIENT_GetConfig** to get firepoint detection mode and firepoint master configuration.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Firepoint detection mode includes Global and Preset mode. In Preset mode, set preset before adding firepoint detection configuration. In Global mode, you need to set firepoint process mode (auto or manual). If set auto mode, you need to set duration.
- Set channel number: If the camera is single-sensor camera, the subscribing channel number is 0; If the camera is multi-sensor camera, the subscribing channel number is 0 or 1; and channel 1 is thermal.

2.10.3.2 Firepoint Snapshot

Figure 2-15 Process of firepoint snapshot



Process Description

Getting

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_GetNewDevConfig** to get firepoint snapshot configuration.
- Step 4 Call **CLIENT_ParseData** to parse firepoint snapshot, and get the corresponding struct.

Step 5 After using the function module, call **CLIENT_Logout** to logout the device.

Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Setting

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

Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.

Step 3 Call **CLIENT_PacketData** to pack firepoint snapshot configuration.

Step 4 Call **CLIENT_SetNewDevConfig** to set firepoint snapshot configuration.

Step 5 After using the function module, call **CLIENT_Logout** to logout the device.

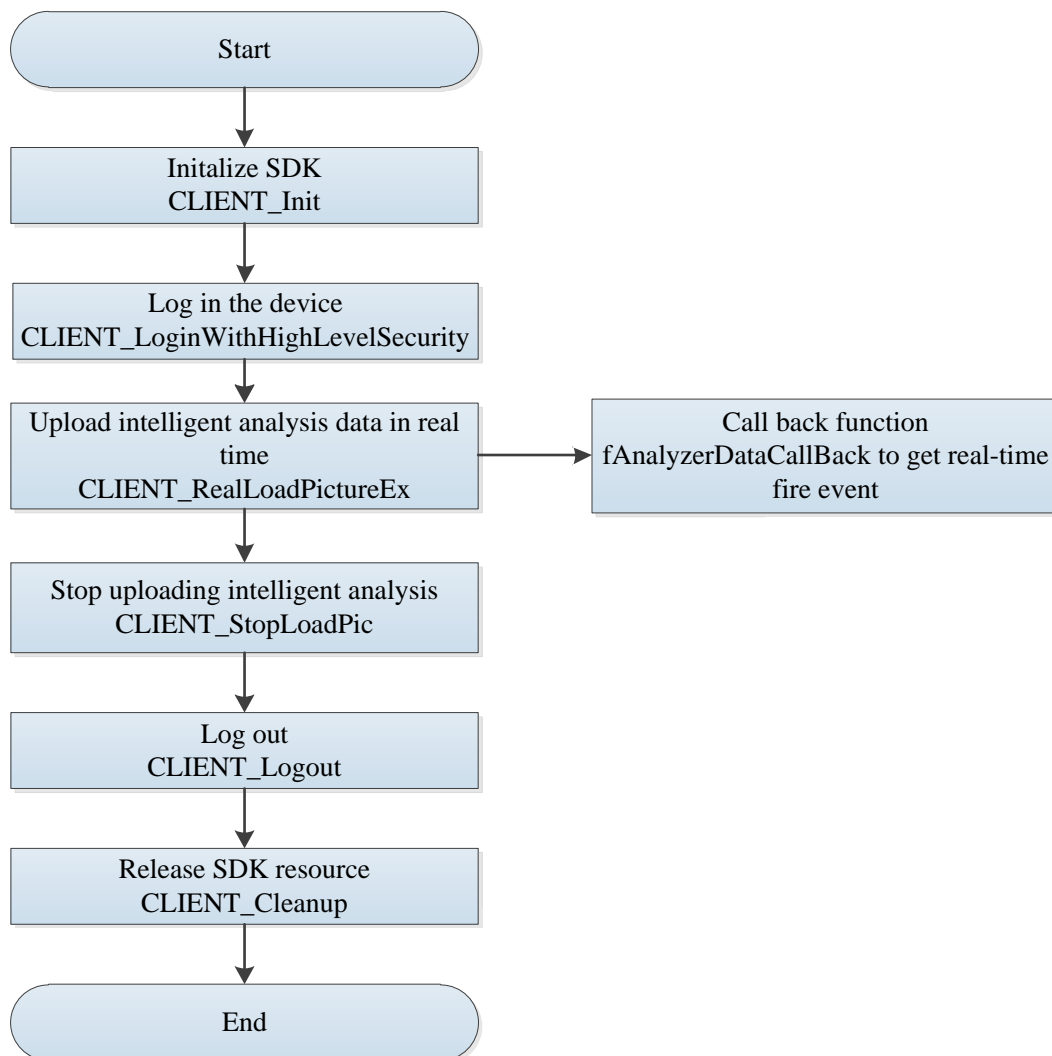
Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Firepoint linking snapshot is mainly for setting the snapshot strategy, including three pictures: two are from visual camera and one is from thermal camera. The first pictures of visual camera and thermal camera are panoramas; when firepoint event is triggered, the second picture of visual camera is for details. Only configure multi-sensor linkage in firepoint master configuration can you capture the detail snapshot.
- Set channel number: If the camera is single-sensor camera, the subscribing channel number is 0; If the camera is multi-sensor camera, the subscribing channel number is 0 or 1; and channel 1 is thermal.

2.10.3.3 Subscribing Firepoint Event

Figure 2-16 Process of subscribing firepoint event,



Process Description

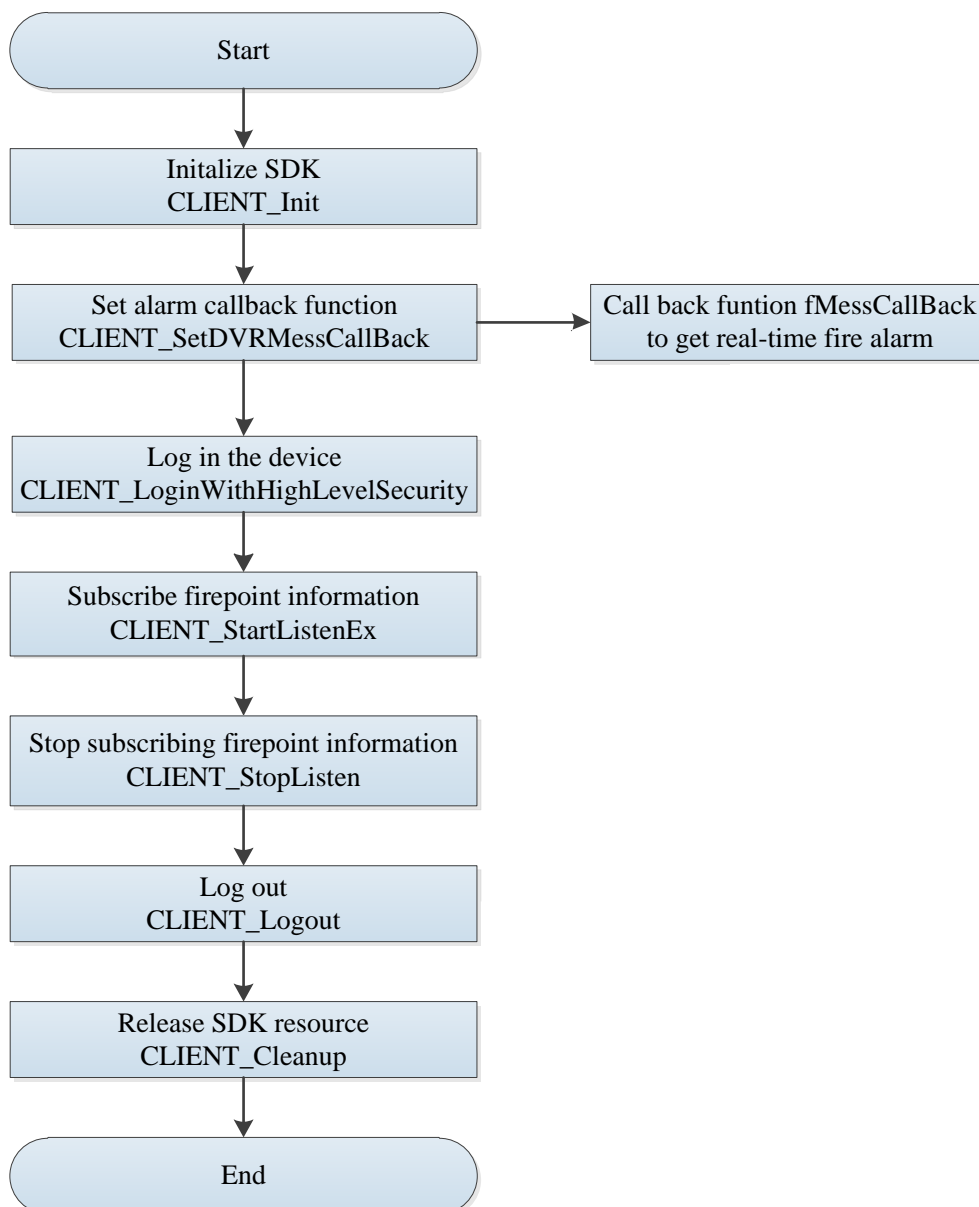
- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_RealLoadPictureEx** to subscribe event, and when event triggers, fAnalyzerDataCallBack will remind you.
- Step 4 Call **CLIENT_StopLoadPic** to stop subscribing firepoint event.
- Step 5 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Subscribing firepoint information corresponds to the FireWarning event on device. When the event is triggered, it will do snapshot operation. When firepoint appears, it sends the word start; when firepoint disappears, it sends the word stop.

2.10.3.4 Subscribing Firepoint Information

Figure 2-17 Process of subscribing firepoint information



Process Description

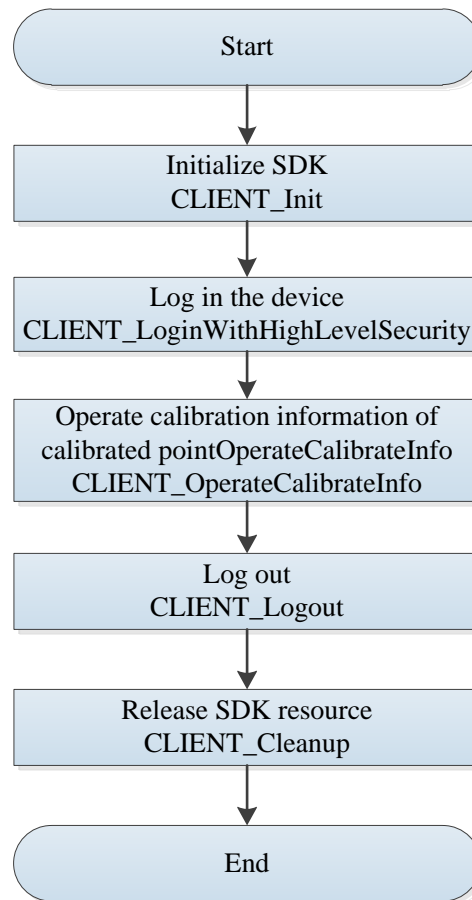
- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_SetDVRMessCallBack** set alarm callback function, and when event triggers, fMessCallBack will remind you.
- Step 3 Call CLIENT_LoginWithHighLevelSecurity to log in the device.
- Step 4 Call **CLIENT_StartListenEx** to subscribe firepoint information.
- Step 5 Call **CLIENT_StopListen** to stop subscribing firepoint information.
- Step 6 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 7 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Subscribing firepoint information corresponds to the FireWarningInfo event on device. After FireWarning is triggered, the event with firepoint details will be reported, including firepoint number, firepoint positioning calculation, PTZ value of the device, and GPS. The reports are made periodically for timely update of the firepoint.
- FireWarningInfo is state update event, and does not have common linkages, including snapshot and record and so on.

2.10.3.5 Firepoint calibration information

Figure 2-18 Process of firepoint calibration information



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call CLIENT_LoginWithHighLevelSecurity to log in the device.
- Step 3 Call **CLIENT_OperateCalibrateInfo** to operate calibration information of calibrated point. Set emType to get, set and delete calibration information of a certain point, and get all calibration information.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Firepoint calibration is mainly to provide calibration information after firepoint event is triggered, which can help to calculate the GPS coordinate of the firepoint, so that user can arrive at scene and handle the firepoint quickly.
- During calibration, the more accurate the actual reference point, the more calibrating point, and more precious the calculated result.
- Calibration selection depends on detection range of the device.

2.10.4 Example Code

```
//Get firepoint detection mode
NET_FIREWARNING_MODE_INFO m_stuFirewarningmodelInfo;
BOOL bRet = CLIENT_GetConfig(m_LoginID, NET_EM_CFG_FIRE_WARNINGMODE,m_nChannel,&m_
stuFirewarningmodelInfo,sizeof(m_ stuFirewarningmodelInfo), 5000);
//Set firepoint detection mode
NET_FIREWARNING_MODE_INFO m_stuFirewarningmodelInfo;
BOOL bRet = CLIENT_SetConfig(m_LoginID, NET_EM_CFG_FIRE_WARNINGMODE,m_nChannel,&m_
stuFirewarningmodelInfo,sizeof(m_ stuFirewarningmodelInfo), 5000);
//Set firepoint configuration
CFG_FIREWARNING_EXT_INFO stuInfo = {0};
stuInfo. bVisualOverviewEnable = ture;
stuInfo. bVisualDetailEnable= ture;
stuInfo. nThermoSnapTimes= 1;
char szJsonBuf[512 * 40] = {0};
BOOL bRet = CLIENT_PacketData(CFG_CMD_FIRE_WARNING_EXT,(LPVOID)&stuInfo, sizeof(stuInfo), szJsonBuf,
sizeof(szJsonBuf));
if (bRet){
    int nerror = 0;
    int nrestart = 0;
    int nChannelID = -1;
    bRet = CLIENT_SetNewDevConfig(m_iLoginID, CFG_CMD_FIRE_WARNING_EXT, nChannelID, szJsonBuf,
512*40, &nerror, &nrestart, 3000);
}
//Get firepoint configuration
char szJsonBuf[1024 * 40] = {0};
int nerror = 0;
int nChannel = -1;

BOOL ret = CLIENT_GetNewDevConfig(m_iLoginID,
CFG_CMD_FIRE_WARNING_EXT,nChannel,szJsonBuf,1024*40,&nerror,3000);
if (0 != ret)
```

```

{
    CFG_FIREWARNING_EXT_INFO stuInfo = {0};
    DWORD dwRetLen = 0;
    ret =
CLIENT_ParseData(CFG_CMD_FIRE_WARNING_EXT,szJsonBuf,(char*)&stuInfo,sizeof(stuInfo),&dwRetLen);
    if (!ret)
    {
        //Fail to get configuration
        return ;
    }
    else
    {
    }
}
else
{
    // Fail to get configuration
    return ;
}
//Subscribe firepoint event
LDWORD dwUser = 0;
g_IRealLoadHandle = CLIENT_RealLoadPictureEx(gILoginHandle, 0, EVENT_IVS_FIREWARNING, TRUE,
AnalyzerDataCallBack, dwUser, NULL);
if (0 == g_IRealLoadHandle)
{
    //Fail to subscribe firepoint event
    return;
}
//Stop subscribing firepoint event
if (0 != g_IRealLoadHandle)
{
    if (FALSE == CLIENT_StopLoadPic(g_IRealLoadHandle))
    {
        // Fail to stop subscribing firepoint event
    }
    else
    {
        g_IRealLoadHandle = 0;
    }
}
}

```

```

//Subscribe firepoint information
CLIENT_SetDVRMessCallBack(MessCallBack , NULL);
if( CLIENT_StartListenEx(g_ILoginHandle))
{
    g_bStartListenFlag = TRUE;
    printf("Listen Success!\nJust Wait Event....\n");
}
else
{
    //Fail to subscribe firepoint information
}
//Stop subscribing firepoint information
if (g_bStartListenFlag)
{
    if (!CLIENT_StopListen(g_ILoginHandle))
    {
        //Fail to stop subscribing firepoint information
    }
    else
    {
        {
            g_bStartListenFlag = FALSE;
        }
    }
}
//Calibration information operation
NET_IN_DELETE_CALIBRATEINFO_INFO stuIn = {sizeof(stuIn)};
NET_OUT_DELETE_CALIBRATEINFO_INFO stuOut = {sizeof(stuOut)};
stuIn.nID = 1;
BOOL bRet = CLIENT_OperateCalibrateInfo(loginHandle, EM_CALIBRATEINFO_OPERATE_DELETE, &stuIn,
&stuOut, 3000);

```

2.11 Thermal PTZ Control

2.11.1 Introduction

The function is a PTZ movement for 360° global firepoint detection.

- Pan group limit mode configuration: Set pan group limit enable and disable.
- Pan group limit information: Set, delete, and get pan limit information.
- PTZ control: Pause the current PTZ action and resume PTZ last task.

2.11.2 Interface Overview

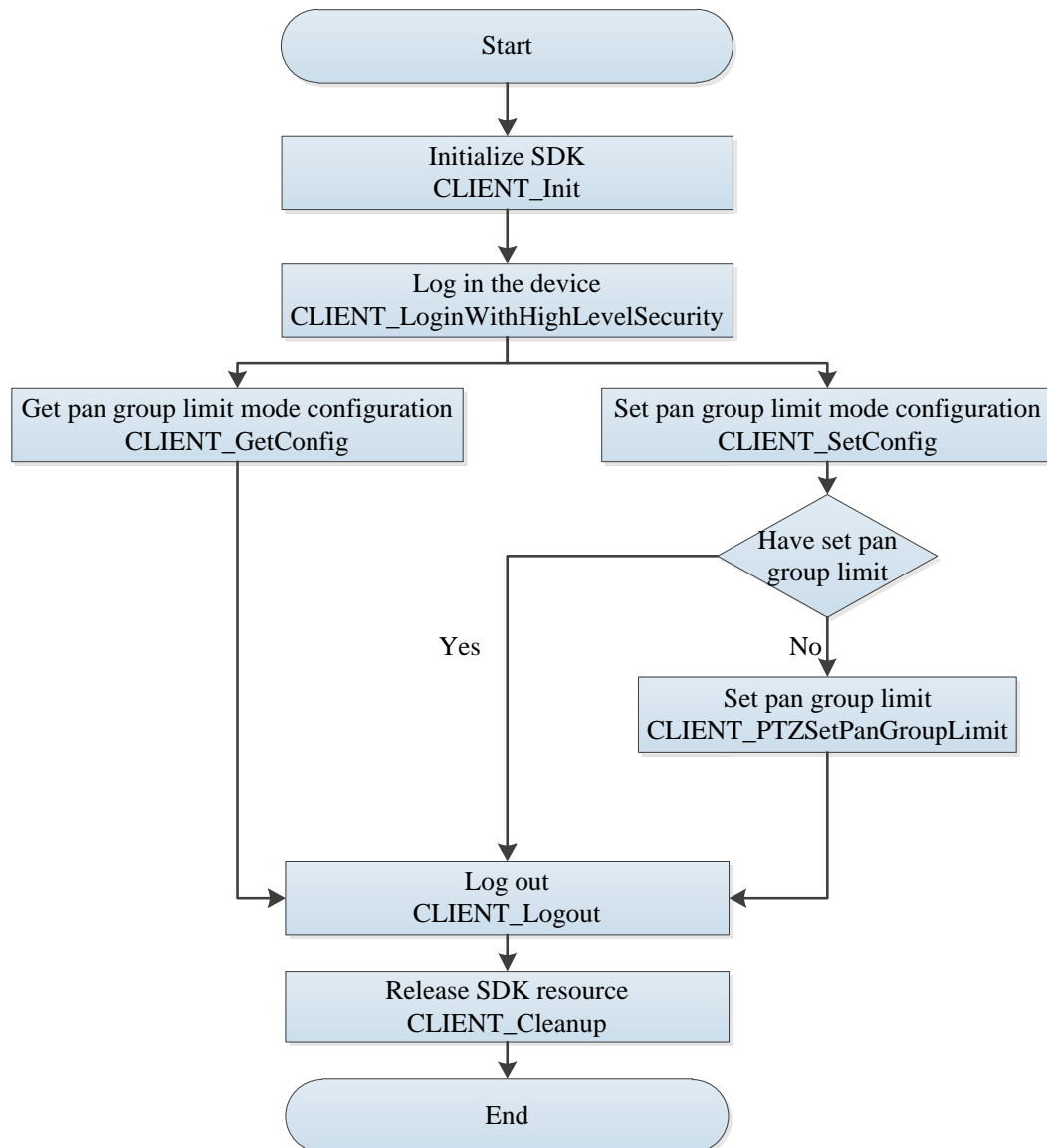
Table 2-12 Interfaces of thermal PTZ control

Interface	Implication
CLIENT_GetConfig	Get pan group limit mode configuration
CLIENT_SetConfig	Set pan group limit mode configuration
CLIENT_PTZSetPanGroup	Set pan interface
CLIENT_PTZGetPanGroup	Get pan group information
CLIENT_PTZGotoPanPosition	Go to pan position
CLIENT_PTZSetPanGroupLimit	Set pan group limit
CLIENT_ResumePtzLastTask	Resume PTZ last task
CLIENT_PausePtzAction	Pause the current PTZ action

2.11.3 Process

2.11.3.1 Pan Group Limit Mode Configuration

Figure 2-19 Process of pan group limit mode configuration



Process Description

Setting

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_SetConfig** to set pan group limit mode configuration.
- Step 4 If you have not set pan group limit before, call **CLIENT_PTZSetPanGroupLimit** to set the limit.
- Step 5 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Getting

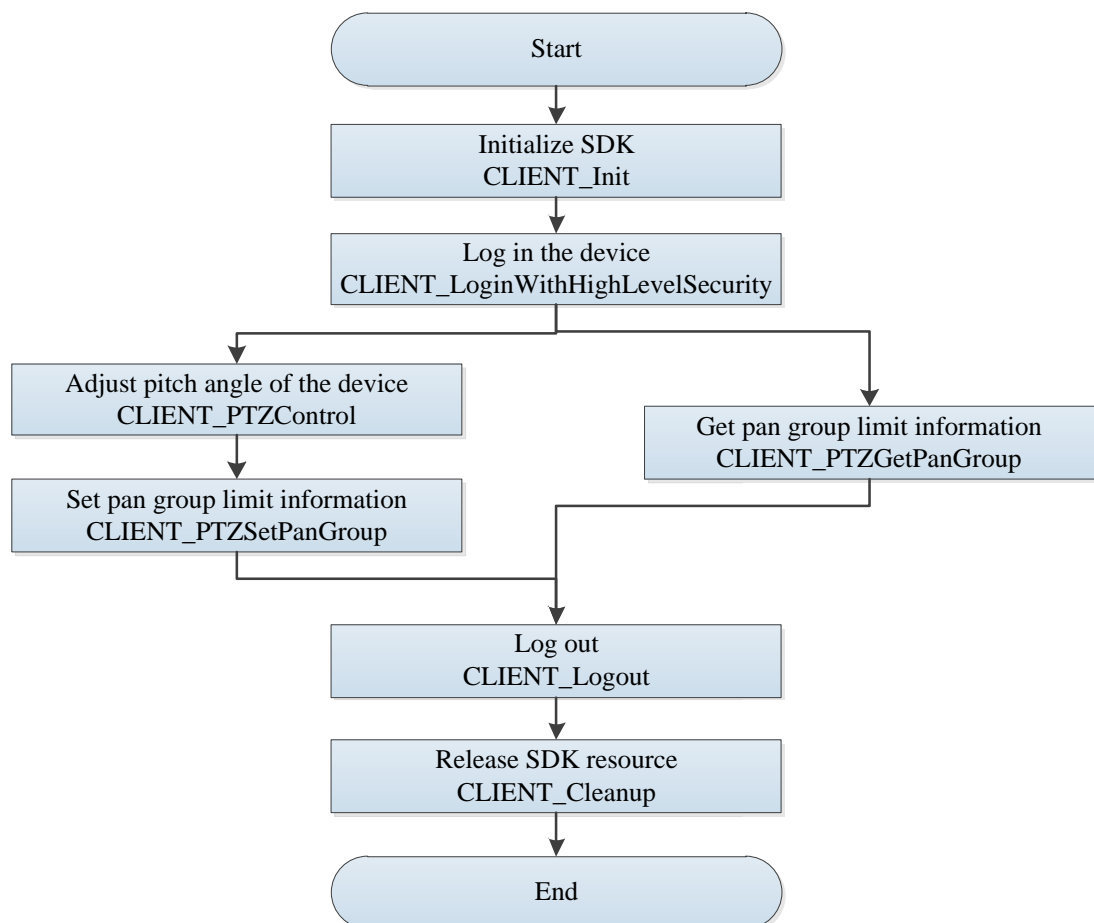
- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_GetConfig** to get pan group limit mode configuration.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Call **CLIENT_PTZSetPanGroupLimit** to set pan group limit after setting pan group limit mode configuration.
- When enabling left and right limit, you need set left limit and right limit; when enabling area scan, you need to set upper limit, down limit, left limit, and right limit.
- Mode setting unified process all pans, cannot set single pan.

2.11.3.2 Set and Get Pan Group Limit Information

Figure 2-20 Process of setting and getting pan group limit information



Process Description

Setting

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.

- Step 3 Call **CLIENT_PTZControl** to adjust pitch angle of the device.
- Step 4 Call **CLIENT_PTZSetPanGroup** to set pan group limit information.
- Step 5 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Getting

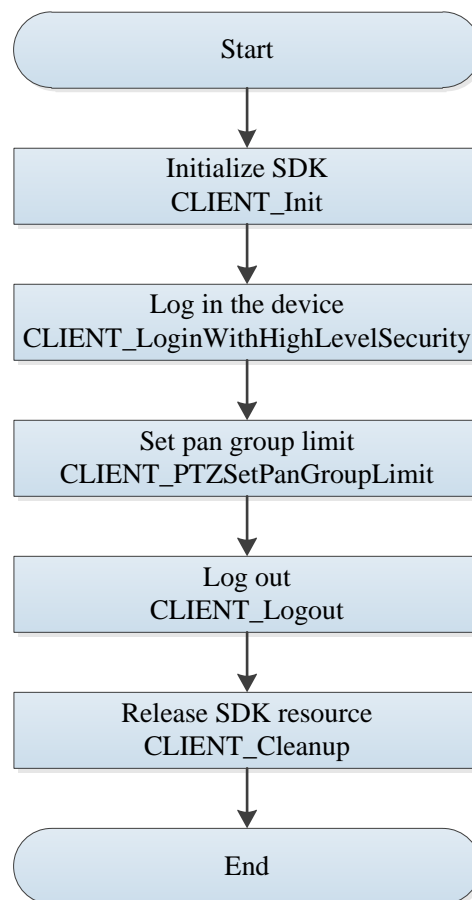
- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to login the device.
- Step 3 Call **CLIENT_GetConfig** to get pan group limit information.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- You can Max. configure 8 pans.
- Call PTZ mobile interface to configure each pan, adjust pitch angle of the device and then call and set pan port.

2.11.3.3 Setting Pan Group Limit

Figure 2-21 Process of setting pan group limit



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.

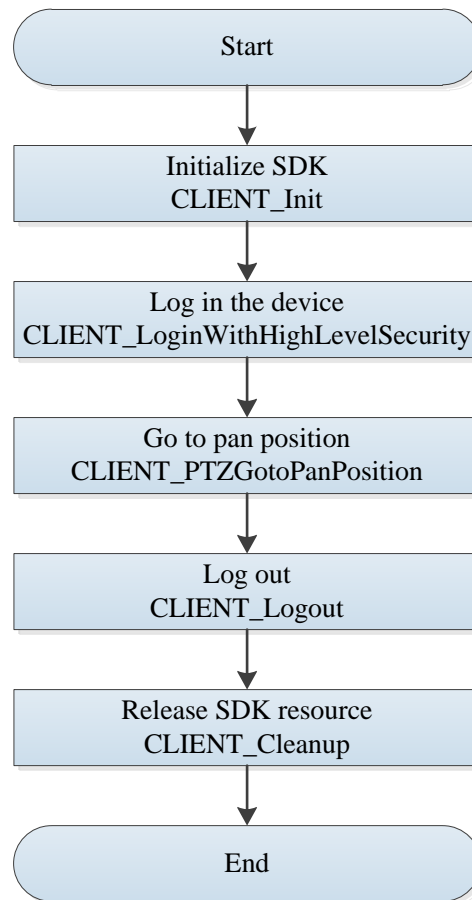
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_PTZSetPanGroupLimit** to set pan group limit.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

Using with pan group limit mode configuration.

2.11.3.4 Going to pan position

Figure 2-22 Process of going to pan position



Process Description

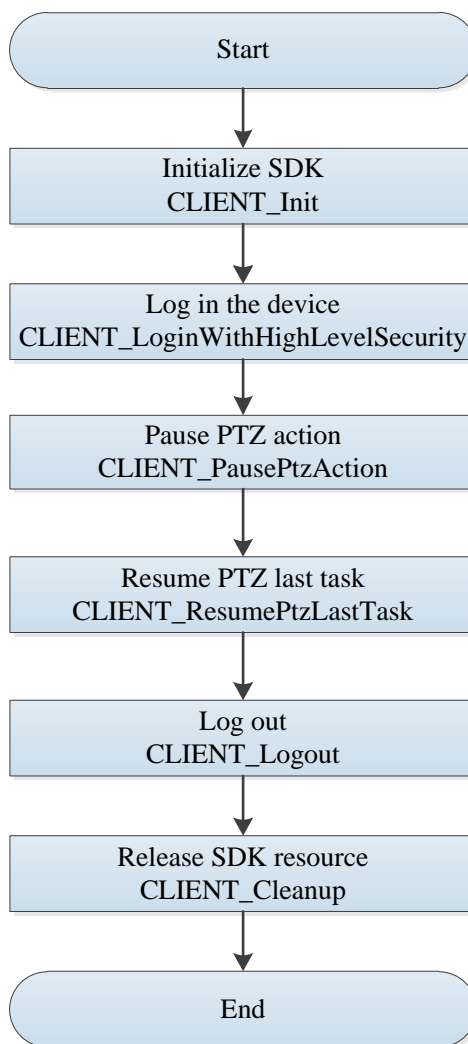
- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_PTZGotoPanPosition** to go pan position.
- Step 4 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

Check whether the pan exists, and then call the pan on a certain position to view the pitch angle of the pan, then you can check whether the surveillance covers all on the vertical direction.

2.11.3.5 Pausing and Resuming PTZ Action

Figure 2-23 Process of pausing and resuming PTZ action



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in the device.
- Step 3 Call **CLIENT_PausePtzAction** to pause PTZ action.
- Step 4 Call **CLIENT_ResumePtzLastTask** to resume PTZ last task.
- Step 5 After using the function module, call **CLIENT_Logout** to logout the device.
- Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- In tour, pattern or pan mode, if you interrupt the operation by moving PTZ manually (move in right/left/up/down direction or zoom operation), the command is used for resuming PTZ operation.
- After interrupting the current command manually, fire linkage stops. Click continue to enable the linkage.

2.11.4 Example Code

```
//Get pan group limit mode configuration
NET_CFG_HORIZONTAL_ROTATION_GROUP_SCAN_INFO  m_stuInfo;
BOOL bRet = CLIENT_GetConfig(m_LoginID,
NET_EM_CFG_PTZ_HORIZONTAL_ROTATION_GROUP_SCAN,m_nChannel,&m_stuInfo,sizeof(m_stuInfo),
5000);

//Set pan group limit mode configuration
NET_CFG_HORIZONTAL_ROTATION_GROUP_SCAN_INFO  m_stuInfo;
BOOL bRet = CLIENT_SetConfig(m_LoginID,
NET_EM_CFG_PTZ_HORIZONTAL_ROTATION_GROUP_SCAN,m_nChannel,&m_stuInfo,sizeof(m_stuInfo),
5000);

//Set pan port
NET_IN_SET_PAN_GROUP_PARAM  m_stuinInfo;
NET_OUT_SET_PAN_GROUP_PARAM m_stuoutInfo;
BOOL bRet =CLIENT_PTZSetPanGroup(m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);

//Get sigle pan port
NET_IN_GET_PAN_GROUP_PARAM  m_stuinInfo;
NET_OUT_GET_PAN_GROUP_PARAM m_stuoutInfo;
BOOL bRet =CLIENT_PTZGetPanGroup(m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);

//Set pan group limit
NET_IN_PAN_GROUP_LIMIT_INFO  m_stuinInfo;
NET_OUT_PAN_GROUP_LIMIT_INFO m_stuoutInfo;
BOOL bRet =CLIENT_PTZSetPanGroupLimit (m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);

//Go to pan position
NET_IN_GOTO_PAN_POSITION  m_stuinInfo;
NET_OUT_GOTO_PAN_POSITION m_stuoutInfo;
BOOL bRet =CLIENT_PTZGotoPanPosition (m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);

// Resume PTZ last task
NET_IN_RESUME_PTZ_LASTTASK_INFO  m_stuinInfo;
NET_OUT_RESUME_PTZ_LASTTASK_INFO m_stuoutInfo;
BOOL bRet =CLIENT_ResumePtzLastTask(m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);

//Pause the current PTZ action
NET_IN_PAUSE_PTZ_ACTION_INFO  m_stuinInfo;
NET_OUT_PAUSE_PTZ_ACTION_INFO m_stuoutInfo;
BOOL bRet =CLIENT_PausePtzAction (m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);
```

3 Interface Definition

3.1 SDK Initialization

3.1.1 SDK CLIENT_Init

Table 3-1 Initialize SDK

Item	Description	
Name	Initialize SDK.	
Function	BOOL CLIENT_Init(fDisconnect cbDisconnect, LDWORD dwUser);	
Parameter	[in]cbDisconnect	Disconnection callback.
	[in]dwUser	User parameter of disconnection callback.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	<ul style="list-style-type: none">• The precondition for calling other function modules.• If the callback is set as NULL, the callback will not be sent to the user after the device is disconnected.	

3.1.2 CLIENT_Cleanup

Table 3-2 Clean up SDK

Item	Description
Name	Clean up SDK.
Function	void CLIENT_Cleanup();
Parameter	None.
Return value	None.
Note	Call the SDK cleanup interface before the process ends.

3.1.3 CLIENT_SetAutoReconnect

Table 3-3 Set reconnection callback

Item	Description	
Name	Set auto reconnection callback.	
Function	void CLIENT_SetAutoReconnect(fHaveReConnect cbAutoConnect, LDWORD dwUser);	
Parameter	[in]cbAutoConnect	Reconnection callback.
	[in]dwUser	User parameter of disconnection callback.
Return value	None.	

Item	Description
Note	Set the reconnection callback interface. If the callback is set as NULL, it will not connect automatically.

3.1.4 CLIENT_SetNetworkParam

Table 3-4 Set network parameter

Item	Description
Name	Set the related parameters for network environment.
Function	void CLIENT_SetNetworkParam(NET_PARAM *pNetParam);
Parameter	[in]pNetParam Parameters such as network delay, reconnection times, and cache size.
Return value	None.
Note	Adjust the parameters according to the actual network environment.

3.2 Device Initialization

3.2.1 CLIENT_StartSearchDevicesEx

Table 3-5 Search for device

Item	Description
Name	Search the device.
Function	LLONG CLIENT_StartSearchDevicesEx (NET_IN_STARTSERACH_DEVICE* pInBuf, NET_OUT_STARTSERACH_DEVICE* pOutBuf);
Parameter	[in] pInBuf Output parameter. Refer to NET_IN_STARTSERACH_DEVICE
	[out] pOutBuf Output parameter. Refer to NET_OUT_STARTSERACH_DEVICE
Return value	Searching handle.
Note	Multi-thread calling is not supported.

3.2.2 CLIENT_InitDevAccount

Table 3-6 Initialize device

Item	Description
Name	Initialize the device.

Item	Description	
Function	<pre> BOOL CLIENT_InitDevAccount(const NET_IN_INIT_DEVICE_ACCOUNT *pInitAccountIn, NET_OUT_INIT_DEVICE_ACCOUNT *pInitAccountOut, DWORD dwWaitTime, char *szLocallp); </pre>	
Parameter	[in]pInitAccountIn	Corresponds to structure of NET_IN_INIT_DEVICE_ACCOUNT.
	[out]pInitAccountOut	Corresponds to structure of NET_OUT_INIT_DEVICE_ACCOUNT.
	[in]dwWaitTime	Timeout.
	[in]szLocallp	<ul style="list-style-type: none"> In case of single network card, the last parameter is not required to be filled. In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.2.3 CLIENT_GetDescriptionForResetPwd

Table 3-7 Get information for password reset

Name	Description	
Name	Get information for password reset.	
Function	<pre> BOOL CLIENT_GetDescriptionForResetPwd(const NET_IN_DESCRIPTION_FOR_RESET_PWD *pDescriptionIn, NET_OUT_DESCRIPTION_FOR_RESET_PWD *pDescriptionOut, DWORD dwWaitTime, char *szLocallp); </pre>	
Parameter	[in]pDescriptionIn	Corresponds to structure of NET_IN_DESCRIPTION_FOR_RESET_PWD.
	[out]pDescriptionOut	Corresponds to structure of NET_OUT_DESCRIPTION_FOR_RESET_PWD.
	[in]dwWaitTime	Timeout.
	[in]szLocallp	<ul style="list-style-type: none"> In case of single network card, the last parameter is not required to be filled. In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.2.4 CLIENT_CheckAuthCode

Table 3-8 Check the validity of security code

Item	Description	
Name	Check the validity of security code.	
Function	<pre> BOOL CLIENT_CheckAuthCode(const NET_IN_CHECK_AUTHCODE *pCheckAuthCodeIn, NET_OUT_CHECK_AUTHCODE *pCheckAuthCodeOut, DWORD dwWaitTime, char *szLocalIp); </pre>	
Parameter	[in]pCheckAuthCodeIn	Corresponds to structure of NET_IN_CHECK_AUTHCODE.
	[out]pCheckAuthCodeOut	Corresponds to structure of NET_OUT_CHECK_AUTHCODE.
	[in]dwWaitTime	Timeout.
	[in]szLocalIp	<ul style="list-style-type: none"> In case of single network card, the last parameter is not required to be filled. In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.2.5 CLIENT_ResetPwd

Table 3-9 Reset the password

Item	Description	
Name	Reset the password.	
Function	<pre> BOOL CLIENT_ResetPwd(const NET_IN_RESET_PWD *pResetPwdIn, NET_OUT_RESET_PWD *pResetPwdOut, DWORD dwWaitTime, char *szLocalIp); </pre>	
Parameter	[in]pResetPwdIn	Corresponds to structure of NET_IN_RESET_PWD.
	[out]pResetPwdOut	Corresponds to structure of NET_OUT_RESET_PWD.
	[in]dwWaitTime	Timeout.
	[in]szLocalIp	<ul style="list-style-type: none"> In case of single network card, the last parameter is not required to be filled. In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.2.6 CLIENT_GetPwdSpecification

Table 3-10 Get password rules

Item	Description	
Name	Get password rules.	
Function	BOOL CLIENT_GetPwdSpecification(const NET_IN_PWD_SPECI *pPwdSpecIn, NET_OUT_PWD_SPECI *pPwdSpeciOut, DWORD dwWaitTime, char *szLocalIp);	
Parameter	[in]pPwdSpecIn	Corresponds to structure of NET_IN_PWD_SPECI.
	[out]pPwdSpeciOut	Corresponds to structure of NET_OUT_PWD_SPECI.
	[in]dwWaitTime	Timeout.
	[in]szLocalIp	<ul style="list-style-type: none">• In case of single network card, the last parameter is not required to be filled.• In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	None.	

3.2.7 CLIENT_StopSearchDevices

Table 3-11 Stop searching device

Item	Description	
Name	Stop searching.	
Function	BOOL CLIENT_StopSearchDevices (LLONG ISearchHandle);	
Parameter	[in] ISearchHandle	Searching handle.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	Multi-thread calling is not supported.	

3.3 Device Login

3.3.1 CLIENT_LoginWithHighLevelSecurity

Table 3-12 Log in with high level security

Item	Description
Name	Login the device with high level security.

Item	Description		
Function	LLONG CLIENT_LoginWithHighLevelSecurity (NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY* pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY* pstOutParam);		
参数	[in] pstInParam	[in] dwSize	Structure size.
		[in] szIP	Device IP.
		[in] nPort	Device port.
		[in] szUserName	User name.
		[in] szPassword	Password.
		[in] emSpecCap	Login type.
		[in] pCapParam	Login type parameter.
	[out] pstOutParam	[in] dwSize	Structure size.
		[out] stuDeviceInfo	Device information.
		[out] nError	Error code.
Return value	<ul style="list-style-type: none"> ● Success: Not 0. ● Failure: 0. 		
Note	Login the device with high level security. CLIENT_LoginEx2 can still be used, but there are security risks, so it is highly recommended to use the latest interface CLIENT_LoginWithHighLevelSecurity to log in to the device.		

Table 3-13 Error code and meaning

Error code	Meaning
1	Wrong password.
2	The user name does not exist.
3	Login timeout.
4	The account has logged in.
5	The account has been locked.
6	The account has been blacklisted.
7	The device resource is insufficient and the system is busy.
8	Sub connection failed.
9	Main connection failed.
10	Exceeds the maximum allowed number of user connections.
11	Lacks the dependent libraries such as avnetsdk or avnetsdk.
12	USB flash disk is not inserted or the USB flash disk information is wrong.
13	The IP at client is not authorized for login.

3.3.2 CLIENT_Logout

Table 3-14 Log out

Item	Description
Name	User logout the device.
Function	BOOL CLIENT_Logout(LLONG ILoginID);

Item	Description	
Parameter	[in]ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.4 Real-time Monitoring

3.4.1 CLIENT_RealPlayEx

Table 3-15 Start the real-time monitoring

Item	Description	
Name	Open the real-time monitoring.	
Function	<pre>LLONG CLIENT_RealPlayEx(LONG ILoginID, int nChannelID, HWND hWnd, DH_RealPlayType rType);</pre>	
Parameter	[in]ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in]nChannelID	Video channel number is a round number starting from 0.
	[in]hWnd	Window handle valid only under Windows system.
	[in]rType	Preview type.
Return value	<ul style="list-style-type: none"> Success: not 0 Failure: 0 	
Note	<p>Windows system:</p> <ul style="list-style-type: none"> When hWnd is valid, the corresponding window displays picture. When hWnd is NULL, get the video data through setting a callback and send to user for handle. 	

Table 3-16 Live view type and meaning

Preview type	Meaning
DH_RType_Realplay	Real-time preview
DH_RType_Multiplay	Multi-picture preview
DH_RType_Realplay_0	Real-time monitoring—main stream, equivalent to DH_RType_Realplay
DH_RType_Realplay_1	Real-time monitoring—sub stream 1
DH_RType_Realplay_2	Real-time monitoring—sub stream 2
DH_RType_Realplay_3	Real-time monitoring—sub stream 3
DH_RType_Multiplay_1	Multi-picture preview—1 picture
DH_RType_Multiplay_4	Multi-picture preview—4 pictures
DH_RType_Multiplay_8	Multi-picture preview—8 pictures
DH_RType_Multiplay_9	Multi-picture preview—9 pictures
DH_RType_Multiplay_16	Multi-picture preview—16 pictures

Preview type	Meaning
DH_RType_Multiplay_6	Multi-picture preview—6 pictures
DH_RType_Multiplay_12	Multi-picture preview—12 pictures
DH_RType_Multiplay_25	Multi-picture preview—25 pictures
DH_RType_Multiplay_36	Multi-picture preview—36 pictures

3.4.2 CLIENT_StopRealPlayEx

Table 3-17 Stop the real-time monitoring

Item	Description	
Name	Stop the real-time monitoring.	
Function	<pre> BOOL CLIENT_StopRealPlayEx(LONG IRealHandle); </pre>	
Parameter	[in] IRealHandle	Return value of CLIENT_RealPlayEx.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.4.3 CLIENT_SaveRealData

Table 3-18 Save the real-time monitoring data as file

Item	Description	
Name	Save the real-time monitoring data as file.	
Function	<pre> BOOL CLIENT_SaveRealData(LONG IRealHandle, const char *pchFileName); </pre>	
Parameter	[in] IRealHandle	Return value of CLIENT_RealPlayEx.
	[in] pchFileName	Save path.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.4.4 CLIENT_StopSaveRealData

Table 3-19 Stop saving the real-time monitoring data as file

Item	Description	
Name	Stop saving the real-time monitoring data as file.	
Function	<pre> BOOL CLIENT_StopSaveRealData(LONG IRealHandle); </pre>	
Parameter	[in] IRealHandle	Return value of CLIENT_RealPlayEx.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.4.5 CLIENT_SetRealDataCallbackEx2

Table 3-20 Set the callback of real-time monitoring data

Item	Description
Name	Set the callback of real-time monitoring data.
Function	<pre> BOOL CLIENT_SetRealDataCallBackEx(LONGLONG IRealHandle, fRealDataCallBackEx2 cbRealData, LDWORD dwUser, DWORD dwFlag); </pre>
Parameter	[in] IRealHandle Return value of CLIENT_RealPlayEx.
	[in] cbRealData Callback of monitoring data flow.
	[in] dwUser Parameter of callback for monitoring data flow.
	[in] dwFlag Type of monitoring data in callback. The type is EM_REALDATA_FLAG and supports OR operation.
Return value	<ul style="list-style-type: none"> ● Success: TRUE. ● Failure: FALSE.
Note	None.

Table 3-21 dwFlag type and parameter

dwFlag	Meaning
REALDATA_FLAG_RAW_DATA	Initial data labels.
REALDATA_FLAG_DATA_WITH_FRAME_INFO	Data labels with frame information.
REALDATA_FLAG_YUV_DATA	YUV data labels.
REALDATA_FLAG_PCM_AUDIO_DATA	PCM audio data labels.

3.5 Video Snapshot

3.5.1 CLIENT SnapPictureToFile

Table 3-22 Device snapshot

Item	Description	
Name	Device snapshot.	
Function	BOOL CLIENT_SnapPictureToFile(

Item	Description
Note	<ul style="list-style-type: none"> Synchronous interface. The device captures snapshot and sends to the user through internet. The device is required to support this function.

3.5.2 CLIENT_CapturePictureEx

Table 3-23 Snapshot

Item	Description
Name	Snapshot.
Function	<pre> BOOL CLIENT_CapturePictureEx(LONG hPlayHandle, const char *pchPicFileName, NET_CAPTURE_FORMATS eFormat); </pre>
Parameter	[in] hPlayHandle Return value of CLIENT_RealPlayEx.
	[in] pchPicFileName Save path.
	[in] eFormat Picture format.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE.
Note	<ul style="list-style-type: none"> Synchronous interface. Directly write the picture data as file. Capture the pictures from the real-time monitoring data stream from device.

3.6 PTZ Control

3.6.1 CLIENT_DHPTZControlEx2

Table 3-24 Control PTZ

Item	Description
Name	PTZ control.
Function	<pre> BOOL CLIENT_DHPTZControlEx2(LONG ILoginID, int nChannelID, DWORD dwPTZCommand, LONG IParam1, LONG IParam2, LONG IParam3, BOOL dwStop , void* param4); </pre>
Parameter	[in] ILoginID Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nChannelID Video channel number that is a round number starting from 0.
	[in] dwPTZCommand Control command type.
	[in] IParam1 Parameter 1.

Item	Description	
	[in] lParam2	Parameter 2.
	[in] lParam3	Parameter 3.
	[in] dwStop	Stop mark, which is valid for operations of eight directions. When performing other operations, enter FALSE for this parameter.
	[in] param4	Support the following extension command: DH_EXTPTZ_MOVE_ABSOLUTELY DH_EXTPTZ_MOVE_CONTINUOUSLY DH_EXTPTZ_GOTOPRESET DH_EXTPTZ_SET_VIEW_RANGE DH_EXTPTZ_FOCUS_ABSOLUTELY DH_EXTPTZ_HORSECTORSCAN DH_EXTPTZ_VERSECTORSCAN DH_EXTPTZ_SET_FISHEYE_EPTZ
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	For the relationship between dwPTZCommand and Param1, Param2 and Param3, see the following table.	

Table 3-25 relationship between dwPTZCommand and Param1, Param2 and Param3

dwPTZCommand macro definition	Function	param1	param2	param3
DH_PTZ_UP_CONTROL	Up	None	Vertical speed (1–8)	None
DH_PTZ_DOWN_CONTROL	Down	None	Vertical speed (1–8)	None
DH_PTZ_LEFT_CONTROL	Left	None	Horizontal speed (1–8)	None
DH_PTZ_RIGHT_CONTROL	Right	None	Horizontal speed (1–8)	None
DH_PTZ_ZOOM_ADD_CONTROL	Zoom+	None	Multi-speed	None
DH_PTZ_ZOOM_DEC_CONTROL	Zoom-	None	Multi-speed	None
DH_PTZ_FOCUS_ADD_CONTROL	Focus+	None	Multi-speed	None
DH_PTZ_FOCUS_DEC_CONTROL	Focus-	None	Multi-speed	None
DH_PTZ_APERTURE_ADD_CONTROL	Aperture+	None	Multi-speed	None
DH_PTZ_APERTURE_DEC_CONTROL	Aperture-	None	Multi-speed	None
DH_PTZ_POINT_MOVE_CONTROL	Move to preset point	None	Value of preset point	None
DH_PTZ_POINT_SET_CONTROL	Set	None	Value of preset point	None

dwPTZCommand macro definition	Function	param1	param2	param3
DH_PTZ_POINT_DEL_CONTROL	Delete	None	Value of preset point	None
DH_PTZ_POINT_LOOP_CONTROL	Cruise among points	Cruise route	None	76: Start 99: Automatic 96: Stop
DH_PTZ_LAMP_CONTROL	Lamp wiper	0x01: Start x00: Stop	None	None
DH_EXTPTZ_LEFTTOP	Left top	Vertical speed (1–8)	Horizontal speed (1–8)	None
DH_EXTPTZ_RIGHTTOP	Right top	Vertical speed (1–8)	Horizontal speed (1–8)	None
DH_EXTPTZ_LEFTDOWN	Left bottom	Vertical speed (1–8)	Horizontal speed (1–8)	None
DH_EXTPTZ_RIGHTDOWN	Right bottom	Vertical speed (1–8)	Horizontal speed (1–8)	None
DH_EXTPTZ_ADDTOLOOP	Add preset point to tour	Tour route	Value of preset point	None
DH_EXTPTZ_DELFROMLOOP	Delete preset point in cruise	Cruise route	Value of preset point	None
DH_EXTPTZ_CLOSELOOP	Delete cruise	Cruise route	None	None
DH_EXTPTZ_STARTPANCRUISE	Start horizontal rotation	None	None	None
DH_EXTPTZ_STOPPANCRUISE	Stop horizontal rotation	None	None	None
DH_EXTPTZ_SETLEFTBORDER	Set left border	None	None	None
DH_EXTPTZ_RIGHTBORDER	Set right border	None	None	None
DH_EXTPTZ_STARTLINE SCAN	Start line scan	None	None	None
DH_EXTPTZ_CLOSELINE SCAN	Stop line scan	None	None	None
DH_EXTPTZ_SETMODESTART	Set mode start	Mode route	None	None
DH_EXTPTZ_SETMODESTOP	Set mode stop	Mode route	None	None
DH_EXTPTZ_RUNMODE	Running mode	Mode route	None	None
DH_EXTPTZ_STOPMODE	Stop mode	Mode route	None	None
DH_EXTPTZ_DELETEMODE	Delete mode	Mode route	None	None
DH_EXTPTZ_REVERSECOMMAND	Reverse command	None	None	None

dwPTZCommand macro definition	Function	param1	param2	param3
DH_EXTPTZ_FASTGOTO	Fast positioning	Horizontal coordinate (0–8192)	Vertical coordinate (0–8192)	Zoom (4)
DH_EXTPTZ_AUXIOPEN	Open auxiliary switch	Auxiliary point	None	None
DH_EXTPTZ_AUXICLOSE	Close auxiliary switch	Auxiliary point	None	None
DH_EXTPTZ_OPENMENU	Open SD menu	None	None	None
DH_EXTPTZ_CLOSEMENU	Close menu	None	None	None
DH_EXTPTZ_MENUOK	Menu confirm	None	None	None
DH_EXTPTZ_MENUCANCEL	Menu cancel	None	None	None
DH_EXTPTZ_MENUUP	Menu up	None	None	None
DH_EXTPTZ_MENUDOWN	Menu down	None	None	None
DH_EXTPTZ_MENULEFT	Menu left	None	None	None
DH_EXTPTZ_MENURIGHT	Menu right	None	None	None
DH_EXTPTZ_ALARMHANDLE	Alarm action with PTZ	Alarm input channel	Alarm action type: <ul style="list-style-type: none"> • Preset point • Line scan • Cruise 	Linkage value, such as preset point number
DH_EXTPTZ_MATRIXSWITCH	Matrix switch	Monitor device number (video output number)	Video input number	Matrix number
DH_EXTPTZ_LIGHTCONTROL	Light controller	Refer to DH_PTZ_LAMP_CONTROL	None	None
DH_EXTPTZ_EXACTGOTO	3D positioning	Horizontal angle (0–3600)	Vertical coordinate (0–900)	Zoom (1–128)
DH_EXTPTZ_RESETZERO	Reset to zero	None	None	None
DH_EXTPTZ_UP_TELE	UP +TELE	Speed (1–8)	None	None
DH_EXTPTZ_DOWN_TELE	DOWN +TELE	Speed (1–8)	None	None
DH_EXTPTZ_LEFT_TELE	LEFT +TELE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHT_TELE	RIGHT +TELE	Speed (1–8)	None	None
DH_EXTPTZ_LEFTUP_TELE	LEFTUP +TELE	Speed (1–8)	None	None

dwPTZCommand macro definition	Function	param1	param2	param3
DH_EXTPTZ_LEFTDOWN_TELE	LEFTDOWN +TELE	Speed (1–8)	None	None
DH_EXTPTZ_TIGHTUP_T ELE	TIGHTUP +TELE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHTDOWN N_TELE	RIGHTDOWN +TELE	Speed (1–8)	None	None
DH_EXTPTZ_UP_WIDE	UP +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_DOWN_WI DE	DOWN +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_LEFT_WIDE	LEFT +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHT_WID E	RIGHT +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_LEFTUP_WI DE	LEFTUP +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_LEFTDOWN _WIDE	LEFTDOWN +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHTUP_ WIDE	RIGHTUP +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHTDOWN N_WIDE	RIGHTDOWN +WIDE	Speed (1–8)	None	None

3.7 Voice Talk

3.7.1 CLIENT_StartTalkEx

Table 3-26 Start voice talk

Item	Description	
Name	Start voice talk.	
Function	<pre> LLONG CLIENT_StartTalkEx(LLONG ILoginID, pfAudioDataCallBack pfcB, LDWORD dwUser); </pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pfcB	Audio data callback.
	[in] dwUser	Parameter of audio data callback.
Return value	<ul style="list-style-type: none"> • Success: Not 0. • Failure: 0. 	
Note	None.	

3.7.2 CLIENT_StopTalkEx

Table 3-27 Stop voice talk

Item	Description	
Name	Stop voice talk.	
Function	BOOL CLIENT_StopTalkEx(LLONG ITalkHandle);	
Parameter	[in] ITalkHandle	Return value of CLIENT_StartTalkEx.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	None.	

3.7.3 CLIENT_RecordStartEx

Table 3-28 Start recording

Item	Description	
Name	Start local recording.	
Function	BOOL CLIENT_RecordStartEx(LLONG ILoginID);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	Valid only in Windows system.	

3.7.4 CLIENT_RecordStopEx

Table 3-29 Stop recording

Item	Description	
Name	Stop local recording.	
Function	BOOL CLIENT_RecordStopEx(LLONG ILoginID);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	Valid only in Windows system.	

3.7.5 CLIENT_TalkSendData

Table 3-30 Send audio data

Item	Description	
Name	Send audio data to device.	
Function	LONG CLIENT_TalkSendData(LLONG ITalkHandle,	

Item	Description	
	[in] nWaitTime	Timeout. The unit is millisecond.
Return value	<ul style="list-style-type: none"> Success: Not 0. Failure: 0. 	
Note	None.	

3.8.2 CLIENT_RadiometryDetach

Table 3-33 Stop subscribing heat map data

Item	Description	
Name	Stop subscribing heat map data.	
Function	<pre> BOOL CLIENT_RadiometryDetach(LLONG IAttachHandle); </pre>	
Parameter	[in] IAttachHandle	Return value of CLIENT_RadiometryAttach.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.8.3 CLIENT_RadiometryFetch

Table 3-34 Capture heat map data

Item	Description	
Name	Capture heat map data.	
Function	<pre> BOOL CLIENT_RadiometryFetch(LLONG ILoginID, const NET_IN_RADIOMETRY_FETCH* pInParam, NET_OUT_RADIOMETRY_FETCH* pOutParam, int nWaitTime); </pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam	Input parameter.
	[in] pOutParam	Output parameter.
	[in] nWaitTime	Timeout. The unit is millisecond.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.8.4 CLIENT_RadiometryDataParse

Table 3-35 Parse heat map data

Item	Description
Name	Parse heat map data.

Item	Description	
Function	<pre> BOOL CLIENT_RadiometryDataParse(const NET_RADIOMETRY_DATA* pBuf, unsigned short* plmg, float* pTemp); </pre>	
Parameter	[in] pBuf	Heat map data.
	[in out] plmg	Unzipped data is a gray map. Introducing null pointer indicates this data is not needed.
	[in out] pTemp	Temperature data of each pixel. Introducing null pointer indicates this data is not needed.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.9 Heat Map (Activity)

3.9.1 CLIENT_QueryDevState

Table 3-36 Get the connection state of remote device

Item	Description	
Name	Get the connection state of remote device.	
Function	<pre> BOOL CLIENT_QueryDevState(LLONG ILoginID, int nType, char *pBuf, int nBufLen, int *pRetLen, int waittime=1000); </pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity
	[in] nType	Query information type.
	[out] pBuf	Buffer used to receive the data returned by the query. The data structure of returned data is different according to the type of query.
	[in] nBufLen	Buffer length. The unit is byte.
	[out] pRetLen	The data length returned actually. The unit is byte.
	[in] waittime	Waiting time for query state. The default waiting time is 1000ms. It can be set according to the needs.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

Table 3-37 Relationship between query information and structure

Query Name	nType	pBuf
Get the statistical activity information	DH_DEVSTATE_GET_HEAT_MAP	NET_QUERY_HEAT_MAP

3.10 Thermal Firepoint Control

3.10.1 CLIENT_SetConfig

Table 3-38 Set firpoint detection and firepoint master configuration

Item	Description	
Name	Set firpoint detection and firepoint master configuration.	
Function	<pre> BOOL CLIENT_SetConfig (LLONG ILoginID NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szInBuffer DWORD dwInBufferSize int waittime=3000 int * restart=NULL void * reserve=NULL); </pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emCfgOpType	Set configuration types: <ul style="list-style-type: none"> Firpoint detection mode: NET_EM_CFG_FIRE_WARNINGMODE Firepoint master configuration: NET_EM_CFG_FIRE_WARNING
	[in] nChannelID	Channel number.
	[in] szInBuffer	Configure buffer address.
	[in] dwInBufferSize	Buffer address size.
	[in] waittime	Waiting time.
	[out] restart	Whether to restart.
	[out] reserve	Reserve parameters.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.10.2 CLIENT_GetConfig

Table 3-39 Get firpoint detection and firepoint master configuration

Item	Description	
Name	Get firpoint detection and firepoint master configuration.	
Function	<pre> BOOL CLIENT_GetConfig (LLONG ILoginID </pre>	

Item	Description	
	<pre> NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szOutBuffer DWORD dwOutBufferSize int waittime=3000 void * reserve=NULL); </pre>	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emCfgOpType	Set configuration type: <ul style="list-style-type: none"> Firpoint detection mode: NET_EM_CFG_FIRE_WARNINGMODE Firepoint master configuration: NET_EM_CFG_FIRE_WARNING
	[in] nChannelID	Channel number.
	[out] szOutBuffer	Configure buffer address.
	[in] dwOutBufferSize	Buffer address size.
	[in] waittime	Waiting time.
	[out] reserve	Actual configuration size.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.10.3 CLIENT_PacketData

Table 3-40 Packet data of device configuration

Item	Description	
Name	Pack firepoint snapshot configuration, device installation GPS coordinate configuration, and thermal calibration configuration.	
Function	<pre> BOOL CLIENT_PacketData(char* szCommand, LPVOID lpInBuffer, DWORD dwInBufferSize, char* szOutBuffer, DWORD dwOutBufferSize); </pre>	
Parameter	[in] szCommand	Command Parameters <ul style="list-style-type: none"> Firepoint snapshot configuration: CFG_CMD_FIRE_WARNING_EXT Device installation GPS coordinate configuration: CFG_CMD_DEVLOCATION Thermal calibration configuration: CFG_CMD_LOCATION_CALIBRATE
	[in] lpInBuffer	Input buffer.
	[in] dwInBufferSize	Input buffer size.
	[out]szOutBuffer	Output buffer.

Item	Description	
	[in] dwOutBufferSize	Output buffer size.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.10.4 CLIENT_SetNewDevConfig

Table 3-41 Set device configuration

Item	Description	
Name	Set parameter of firepoint detection mode configuration.	
Function	<pre> BOOL CLIENT_SetNewDevConfig(LLONG ILoginID, char* szCommand, int nChannelID, char* szInBuffer, DWORD dwInBufferSize, int *error, int *restart, int waittime=500); </pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] szCommand	Command Parameters <ul style="list-style-type: none"> • Firepoint snapshot configuration: CFG_CMD_FIRE_WARNING_EXT • Device installation GPS coordinate configuration: CFG_CMD_DEVLOCATION • Thermal calibration configuration: CFG_CMD_LOCATION_CALIBRATE
	[in] nChannelID	Channel number.
	[in] szInBuffer	Input buffer, json information composed of and user storage and used for setting configuration.
	[in] dwInBufferSize	Buffer address size.
	[out] error	Error code address.
	[in] restart	Restart mark address.
	[in] waittime	Waiting time for configuration.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.10.5 CLIENT_GetNewDevConfig

Table 3-42 Get device configuration

Item	Description
Name	Set firpoint detection parameters.
Function	BOOL CLIENT_GetNewDevConfig(

Item	Description	
	LLONG ILoginID, char* szCommand, int nChannelID, char* szOutBuffer, DWORD dwOutBufferSize, int *error, int waittime=500);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] szCommand	Command Parameters <ul style="list-style-type: none"> Firepoint snapshot configuration: CFG_CMD_FIRE_WARNING_EXT Device installation GPS coordinate configuration: CFG_CMD_DEVLOCATION Thermal calibration configuration: CFG_CMD_LOCATION_CALIBRATE
	[in] nChannelID	Channel number.
	[out] szOutBuffer	Output buffer, json information gotten from the device by user storage.
	[in] dwOutBufferSize	Error code address.
	[out] error	Restart mark address.
	[in] waittime	Waiting time for configuration.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.10.6 CLIENT_ParseData

Table 3-43 Parse configuration data

Item	Description	
Name	Parse firepoint snapshot configuration, device installation GPS coordinate configuration, and thermal calibration configuration.	
Function	BOOL CLIENT_ParseData(char* szCommand, char* szInBuffer, LPVOID lpOutBuffer, DWORD dwOutBufferSize, void* pReserved);	
Parameter	[in] szCommand	Command Parameters <ul style="list-style-type: none"> Firepoint snapshot configuration: CFG_CMD_FIRE_WARNING_EXT Device installation GPS coordinate configuration: CFG_CMD_DEVLOCATION Thermal calibration configuration: CFG_CMD_LOCATION_CALIBRATE

Item	Description	
	[in] szInBuffer	Input buffer, string configuration buffer.
	[in] lpOutBuffer	Output buffer.
	[out]dwOutBufferSize	Output buffer size.
	[in] pReserved	Reserve parameters.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.10.7 CLIENT_RealLoadPictureEx

Table 3-44 Subscribe firepoint alarm event

Item	Description	
Name	Subscribe firepoint alarm event.	
Function	LLONG CLIENT_RealLoadPictureEx(LLONG ILoginID, int nChannelID, DWORD dwAlarmType, BOOL bNeedPicFile, fAnalyzerDataCallBack cbAnalyzerData, LDWORD dwUser, void* Reserved);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nChannelID	Channel of subscribing firepoint alarm event.
	[in] dwAlarmType	Expected alarm type Firepoint alarm event: EVENT_IVS_FIREWARNING.
	[in] bNeedPicFile	Whether to subscribe pictures and files.
	[in] cbAnalyzerData	Callback function of intelligent picture alarm.
	[in] dwUser	User data, SDK returns the data to user by calling back intelligent picture alarm function fAnalyzerDataCallBack.
	[out]Reserved	Reserve parameters.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.10.8 CLIENT_StopLoadPic

Table 3-45 Stop subscribing firepoint alarm event

Item	Description	
Name	Stop subscribing firepoint alarm event.	
Function	BOOL CLIENT_StopLoadPic(LLONG IAnalyzerHandle);	
Parameter	[in] IAnalyzerHandle	Subscribing ID of firepoint alarm event.
Return value	<ul style="list-style-type: none"> • Success: TRUE. 	

Item	Description
	<ul style="list-style-type: none"> Failure: FALSE.
Note	None.

3.10.9 CLIENT_SetDVRMessCallBack

Table 3-46 Set firepoint information callback function interface

Item	Description
Name	Set firepoint information callback function interface.
Function	void CLIENT_SetDVRMessCallBack(fMessCallBack cbMessage, LDWORD dwUser);
Parameter	[in] cbMessage Alarm callback function.
	[in] dwUser User data.
Return value	None
Note	Call CLIENT_SetDVRMessCallBack interface before subscribing alarm, and the received callback function does not include events with pictures.

3.10.10 CLIENT_StartListenEx

Table 3-47 Subscribe firepoint information interface

Item	Description
Name	Subscribe firepoint information interface.
Function	BOOL CLIENT_StartListenEx(LLONG ILoginID);
Parameter	[in] ILoginID Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE.
Note	Alarm events of all device give feedback to user by the callback function set by CLIENT_SetDVRMessCallBack interface.

3.10.11 CLIENT_StopListen

Table 3-48 Stop subscribing firepoint information interface

Item	Description
Name	Stop subscribing firepoint information interface.
Function	BOOL CLIENT_StopListen(LLONG ILoginID);
Parameter	[in] ILoginID Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE.
Note	None.

3.10.12 CLIENT_OperateCalibrateInfo

Table 3-49 Operate thermal device calibration information

Item	Description	
Name	Operate thermal device calibration information.	
Function	BOOL CLIENT_OperateCalibrateInfo (LLONG ILoginID, EM_CALIBRATEINFO_OPERATE_TYPE emType, const void* pStuInParam, void* pStuOutParam, int nWaitTime);	
Parameter	[in] ILoginID	Command Parameters parse configuration type.
	[in] emType	Operating types: <ul style="list-style-type: none"> • Get calibration information of certain point: EM_CALIBRATEINFO_OPERATE_GET • Set calibration information of certain point: EM_CALIBRATEINFO_OPERATE_SET • Delete calibration information of certain point: EM_CALIBRATEINFO_OPERATE_DELETE • Get all calibration information: EM_CALIBRATEINFO_OPERATE_GETALL
	[in] pStuInParam	Input buffer.
	[out] pStuOutParam	Output buffer.
	[in] nWaitTime	Waiting time.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.11 Thermal PTZ Control

3.11.1 CLIENT_SetConfig

Table 3-50 Set firpoint detection and firepoint master configuration

Item	Description
Name	Set firpoint detection and firepoint master configuration.
Function	BOOL CLIENT_SetConfig (LLONG ILoginID NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szInBuffer DWORD dwInBufferSize int waittime=3000 int * restart=NULL void * reserve=NULL

Item	Description	
);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emCfgOpType	Set configuration type: Pan group limit mode configuration: NET_EM_CFG_PTZ_HORIZONTAL_ROTATION_GROUP_SCAN
	[in] nChannelID	Channel number.
	[in] szInBuffer	Configured buffer address.
	[in] dwInBufferSize	Buffer address size.
	[in] waittime	Waiting time.
	[out] restart	Whether to restart.
	[out] reserve	Reserve parameters.
Return Value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.11.2 CLIENT_GetConfig

Table 3-51 Get firpoint detection and firepoint master configuration

Item	Description	
Name	Get firpoint detection and firepoint master configuration.	
Function	<pre> BOOL CLIENT_GetConfig (LLONG ILoginID NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szOutBuffer DWORD dwOutBufferSize int waittime=3000 void * reserve=NULL); </pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emCfgOpType	Set configuration type: Pan group limit mode configuration: NET_EM_CFG_PTZ_HORIZONTAL_ROTATION_GROUP_SCAN
	[in] nChannelID	Channel number.
	[out] szOutBuffer	Got buffer address.
	[in] dwOutBufferSize	Buffer address size.
	[in] waittime	Waiting time.
	[out] reserve	Actual configuration size.
Return Value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE. 	
Note	None.	

3.11.3 CLIENT_PTZSetPanGroup

Table 3-52 Set pan interface

Item	Description	
Name	Set pan interface.	
Function	BOOL CLIENT_SetConfig (LLONG ILoginID, const NET_IN_SET_PAN_GROUP_PARAM* pInParam, NET_OUT_SET_PAN_GROUP_PARAM* pOutParam, int nWaitTime);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam	Input parameters buffer address.
	[out]pOutParam	Output parameters buffer address.
	[in]nWaitTime	Waiting time.
Return Value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	None.	

3.11.4 CLIENT_PTZGetPanGroup

Table 3-53 Get pan interface

Item	Description	
Name	Get pan interface.	
Function	BOOL CLIENT_GetConfig (LLONG ILoginID, const NET_IN_GET_PAN_GROUP_PARAM* pInParam, NET_OUT_GET_PAN_GROUP_PARAM* pOutParam, int nWaitTime);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam	Input parameters buffer address.
	[out]pOutParam	Output parameters buffer address.
	[in]nWaitTime	Waiting time.
Return Value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	None.	

3.11.5 CLIENT_PTZSetPanGroupLimit

Table 3-54 Set pan group limit

Item	Description
Name	Set pan group limit.
Function	BOOL CLIENT_PTZSetPanGroupLimit (LLONG ILoginID, const NET_IN_PAN_GROUP_LIMIT_INFO* pInParam,

Item	Description	
	NET_OUT_PAN_GROUP_LIMIT_INFO* pOutParam, int nWaitTime);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam	Input parameters buffer address.
	[out]pOutParam	Output parameters buffer address.
	[in]nWaitTime	Waiting time..
Return Value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.11.6 CLIENT_PTZGotoPanPosition

Table 3-55 Go to pan position

Item	Description	
Name	Go to pan position.	
Function	BOOL CLIENT_PTZGotoPanPosition (LLONG ILoginID, const NET_IN_GOTO_PAN_POSITION* pInParam, NET_OUT_GOTO_PAN_POSITION* pOutParam, int nWaitTime);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam	Input parameters buffer address.
	[out]pOutParam	Output parameters buffer address.
	[in]nWaitTime	Waiting time.
Return Value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

3.11.7 CLIENT_PausePtzAction

Table 3-56 Pause the current PTZ action

Item	Description	
Name	Pause the current PTZ action.	
Function	BOOL CLIENT_PausePtzAction (LLONG ILoginID, const NET_IN_PAUSE_PTZ_ACTION_INFO* pInParam, NET_OUT_PAUSE_PTZ_ACTION_INFO* pOutParam, int nWaitTime);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam	Input parameters buffer address.
	[out]pOutParam	Output parameters buffer address.
	[in]nWaitTime	Waiting time.

Item	Description
Return Value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE.
Note	None.

3.11.8 CLIENT_ResumePtzLastTask

Table 3-57 Resume PTZ last task

Item	Description
Name	Resume PTZ last task.
Function	<pre> BOOL CLIENT_ResumePtzLastTask (LLONG ILoginID, const NET_IN_RESUME_PTZ_LASTTASK_INFO* pInParam, NET_OUT_RESUME_PTZ_LASTTASK_INFO* pOutParam, int nWaitTime); </pre>
Parameter	[in] ILoginID Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pInParam Input parameters buffer address.
	[out]pOutParam Output parameters buffer address.
	[in]nWaitTime Waiting time.
Return Value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE.
Note	None.

4 Callback Definition

4.1 fSearchDevicesCB

Table 4-1 Callback of searching devices (1)

Item	Description	
Name	Callback of searching devices.	
Function	typedef void(CALLBACK *fSearchDevicesCB)(DEVICE_NET_INFO_EX * pDevNetInfo, void* pUserData);	
Parameter	[out]pDevNetInfo	The searched device information.
	[out]pUserData	User data.
Return value	None.	
Note	None.	

4.2 fSearchDevicesCBEx

Table 4-2 Callback of searching devices (2)

Item	Description	
Name	Callback of searching devices.	
Function	typedef void(CALLBACK * fSearchDevicesCBEx)(LLONG ISearchHandle, DEVICE_NET_INFO_EX2 *pDevNetInfo, void* pUserData);	
Parameter	[in]ISearchHandle	SearchHandle
	[out]pDevNetInfo	The searched device information.
	[out]pUserData	User data.
Return value	None.	
Note	None.	

4.3 fDisconnect

Table 4-3 Disconnection callback

Item	Description
Name	Disconnection callback.
Function	typedef void (CALLBACK *fDisconnect)(LLONG ILoginID, char *pchDVRIP, LONG nDVRPort,

Item	Description	
	LDWORD dwUser);	
Parameter	[out] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] pchDVRIP	IP of the disconnected device.
	[out] nDVRPort	Port of the disconnected device.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.4 fHaveReConnect

Table 4-4 Reconnection callback

Item	Description	
Name	Reconnection callback.	
Function	typedef void (CALLBACK *fHaveReConnect)(LLONG ILoginID, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
Parameter	[out] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] pchDVRIP	IP of the reconnected device.
	[out] nDVRPort	Port of the reconnected device.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.5 fRealDataCallBackEx2

Table 4-5 Callback of real-time monitoring data

Item	Description	
Name	Callback of real-time monitoring data.	
Function	typedef void (CALLBACK *fRealDataCallBackEx2)(LLONG IRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LLONG param, LDWORD dwUser);	
Parameter	[out] IRealHandle	Return value of CLIENT_RealPlayEx.

Item	Description	
	[out] dwDataType	Data type: <ul style="list-style-type: none"> 0: Initial data. 1: Data with frame information. 2: YUV data. 3: PCM audio data.
	[out] pBuffer	Address of monitoring data block.
	[out] dwBufSize	Length (unit: byte) of the monitoring data block.
	[out] param	Callback parameter structure. Different dwDataType value corresponds to different type. <ul style="list-style-type: none"> The param is blank pointer when dwDataType is 0. The param is the pointer of tagVideoFrameParam structure when dwDataType is 1. The param is the pointer of tagCBYUVDataParam structure when dwDataType is 2. The param is the pointer of tagCBPCMDDataParam structure when dwDataType is 3.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.6 pfAudioDataCallback

Table 4-6 Callback of audio data of voice talk

Item	Description	
Name	Audio data callback of voice talk.	
Function	<pre>typedef void (CALLBACK *pfAudioDataCallBack)(LONG ITalkHandle, char *pDataBuf, DWORD dwBufSize, BYTE byAudioFlag, LDWORD dwUser);</pre>	
Parameter	[out] ITalkHandle	Return value of CLIENT_StartTalkEx.
	[out] pDataBuf	Address of audio data block.
	[out] dwBufSize	Length of the audio data block. The unit is byte.
	[out] byAudioFlag	Data type: <ul style="list-style-type: none"> 0: Local collecting. 1: Sending from device.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.7 fRadiometryAttachCB

Table 4-7 Callback of temperature distribution data

Item	Description	
Name	Callback of temperature distribution data.	
Function	<pre>typedef void (CALLBACK *fRadiometryAttachCB)(LLONG IAttachHandle, NET_RADIOMETRY_DATA* pBuf, int nBufLen, LDWORD dwUser);</pre>	
Parameter	[out] IAttachHandle	Return value of CLIENT_RadiometryAttach.
	[out] pBuf	Address of data block.
	[out] nBufLen	Length of the data block. The unit is byte.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.8 fAnalyzerDataCallBack

Table 4-8 Firpoint alarm callback function

Item	Description	
Name	Firpoint alarm callback function.	
Function	<pre>typedef int(CALLBACK *fAnalyzerDataCallBack)(LLONG IAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved);</pre>	
Parameter	[out] IAnalyzerHandle	Firepoint alarm subscribing handle.
	[out] dwAlarmType	Alarm type.
	[out] pAlarmInfo	Struct indicator, matching with alarm type.
	[out] pBuffer	Firepoint picture information buffer.
	[out] dwBufSize	Get picture information size.
	[out] dwUser	User data, in accordance with the input user data during setting callback function fSearchDevicesCB.
	[out] nSequence	Fields for recognizing picture repeat: <ul style="list-style-type: none"> 0: The picture appears for the first time, and there are other alarms using the picture. 1: The picture is same with the picture of previous alarm, and there are other alarms using the picture.

Item	Description	
		<ul style="list-style-type: none"> 2: The picture is same with the picture of previous alarm, and it is the last time to appear or the picture only appears once (Most alarms have their own individual pictures, so nSequence vable is 2 generally).
	[out] reserved	<p>The state of current callback function, reserved is the int indicator.</p> <p>*(int *)reserved value description</p> <ul style="list-style-type: none"> 0: The current data is real-time value 1: The current data is offline value 2: Offline data transmission is completed (Data of most intelligent picture alarms are real-time data, so *(int *)reserved is 0 generally).
Return Value	Return value has been abolished, None has special meanings, you just need to return 0.	
Note	<p>It is not recommend to call SDK in this function.</p> <p>Set the function by CLIENT_RealLoadPictureEx/CLIENT_RealLoadPicture. When the device reports firepoint picture even, SDK will call the function. SDK receives 2M buffer by default, so when callback picture information larger than 2M, call CLIENT_SetNetworkParam interface to set receiving buffer again, otherwise SDK will loss the date packet larger than 2M.</p>	

4.9 fMessCallBack

Table 4-9 Firpoint information callback function

Item	Description	
Name	Firpoint information callback function.	
Function	<pre>typedef BOOL (CALLBACK *fMessCallBack)(LONG ICommand, LLONG ILoginID, char *pBuf, DWORD dwBufLen, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser);</pre>	
Parameter	[out] ICommand	<p>Callback alarm event type.</p> <p>Use with pBuf, when ICommand values are different, the directive data types of pBuf are different.</p>
	[out] ILoginID	<p>Login ID.</p> <p>The return value of corresponding login interface CLIENT_LoginEx.</p>
	[out] pBuf	<p>Received alarm data buffer.</p> <p>When the called monitor interface and ICommand values are different, the directive data types of pBuf are</p>

Item	Description	
		different.
	[out] dwBufLen	The size of received alarm data buffer(Unit: Byte).
	[out] pchDVRIP	The IP of the device that reports alarm.
	[out] nDVRPort	The port of the device that reports alarm.
	[out] dwUser	User data, in accordance with the input user data during setting callback function fMessCallBack.
Return Value	None.	
Note	<p>All login devices use a same alarm report callback function.</p> <p>You can define alarm report belongs to which login byLoginl.</p> <p>When the called monitor interface and ICommand values are different, the directive data types of pBuf are different.</p>	

Appendix 1 Cybersecurity Recommendations

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

Mandatory actions to be taken for basic device network security:

1. Use Strong Passwords

Please refer to the following suggestions to set passwords:

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

2. Update Firmware and Client Software in Time

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

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

1. Physical Protection

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

2. Change Passwords Regularly

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

3. Set and Update Passwords Reset Information Timely

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

4. Enable Account Lock

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

5. Change Default HTTP and Other Service Ports

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

6. Enable HTTPS

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

7. MAC Address Binding

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

8. Assign Accounts and Privileges Reasonably

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

9. Disable Unnecessary Services and Choose Secure Modes

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

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

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

10. Audio and Video Encrypted Transmission

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

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

11. Secure Auditing

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

12. Network Log

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

13. Construct a Safe Network Environment

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

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