

NetSDK_JAVA

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.

Reader

- SDK software development engineers
- Project managers
- Product managers

Signals

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

Signal Words	Meaning
 CAUTION	Indicates a potential risk which, if not avoided, could result in property damage, data loss, lower performance, or unpredictable result.
 TIPS	Provides methods to help you solve a problem or save you time.
 NOTE	Provides additional information as the emphasis and supplement to the text.

Revision History

Version	Revision Content	Release Time
V1.0.0	First release.	February 2020

About the Manual

- The manual is for reference only. If there is inconsistency between the manual and the actual product, the actual product shall govern.

- 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 between the actual value of some data and the value provided, if there is any doubt or dispute, please refer to our final explanation.
- Please contact the supplier or customer service if there is any problem occurred when using the device.
- We are not liable for any loss caused by the operations that do not comply with the manual.
- All trademarks, registered trademarks and the company names in the manual are the properties of their respective owners.
- Please visit our website or contact your local service engineer for more information. If there is any uncertainty or controversy, please refer to our final explanation.

Glossary

This chapter provides the definitions to some of the terms 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.

Table of Contents

Foreword	I
Glossary	III
1 Overview	1
1.1 General	1
1.2 Applicability	2
1.3 Solutions of Library Loading Error	2
1.3.1 Using Java Project in Windows	3
1.3.2 Using Java Project in Linux	4
1.3.3 Using Project as a jar Package	5
1.4 Upgrade early jna to latest version	7
2 Function Modules	8
2.1 SDK Initialization	8
2.1.1 Introduction	8
2.1.2 Interface Overview	8
2.1.3 Process	9
2.1.4 Example Code	10
2.2 Device Login	12
2.2.1 Introduction	12
2.2.2 Interface Overview	12
2.2.3 Process	13
2.2.4 Example Code	14
2.3 Real-time Monitoring	16
2.3.1 Introduction	16
2.3.2 Interface Overview	16
2.3.3 Process	16
2.3.4 Example Code	20
2.4 Video Snapshot	22
2.4.1 Introduction	22
2.4.2 Interface Overview	23
2.4.3 Process	23
2.4.4 Example Code	26
2.5 PTZ Control	29
2.5.1 Introduction	29
2.5.2 Interface Overview	29
2.5.3 Process	29
2.5.4 Example Code	31
2.6 Voice Talk	32
2.6.1 Introduction	32
2.6.2 Interface Overview	32
2.6.3 Process	32
2.6.4 Example Code	34
2.7 Alarm Listening	37
2.7.1 Introduction	37

2.7.2 Interface Overview	37
2.7.3 Process.....	38
2.7.4 Example Code	39
2.8 Intelligent Event	40
2.8.1 Introduction	40
2.8.2 Interface Overview	40
2.8.3 Process.....	41
2.8.4 Example Code	42
2.9 Record Playback	44
2.9.1 Introduction	44
2.9.2 Interface Overview	44
2.9.3 Process.....	44
2.9.4 Example Code	46
2.10 Record Download.....	49
2.10.1 Introduction	49
2.10.2 Interface Overview.....	49
2.10.3 Process.....	49
2.10.4 Example Code.....	51
3 Interface Definition	54
3.1 SDK Initialization	54
3.1.1 SDK CLIENT_Init	54
3.1.2 CLIENT_Cleanup	54
3.1.3 CLIENT_SetAutoReconnect.....	54
3.1.4 CLIENT_SetNetworkParam.....	55
3.2 Device Login	55
3.2.1 CLIENT_LoginWithHighLevelSecurity	55
3.2.2 CLIENT_Logout.....	55
3.3 Real-time Monitoring.....	56
3.3.1 CLIENT_RealPlayEx	56
3.3.2 CLIENT_StopRealPlayEx	57
3.3.3 CLIENT_SaveRealData	57
3.3.4 CLIENT_StopSaveRealData	57
3.3.5 CLIENT_SetRealDataCallBackEx	57
3.4 Video Snapshot	58
3.4.1 CLIENT_SnapPictureToFile.....	58
3.4.2 CLIENT_CapturePictureEx.....	59
3.4.3 CLIENT_CapturePictureEx.....	59
3.4.4 Setting Asynchronous Snapshot Callback	59
3.5 PTZ Control.....	60
3.5.1 CLIENT_DHPTZControlEx	60
3.6 Voice Talk.....	64
3.6.1 CLIENT_StartTalkEx.....	64
3.6.2 CLIENT_StopTalkEx.....	64
3.6.3 CLIENT_TalkSendData	65
3.6.4 CLIENT_AudioDecEx	65
3.7 Alarm Listening	65
3.7.1 CLIENT_StartListenEx	65

3.7.2 CLIENT_StopListen	66
3.7.3 CLIENT_SetDVRMessCallBack	66
3.8 Intelligent Event	66
3.8.1 CLIENT_RealLoadPictureEx	66
3.8.2 CLIENT_StopLoadPic	68
3.9 Record Playback	68
3.9.1 CLIENT_PlayBackByTimeEx	68
3.9.2 CLIENT_SetDeviceMode	69
3.9.3 CLIENT_StopPlayBack	69
3.9.4 CLIENT_PausePlayBack	70
3.10 Record Download	70
3.10.1 CLIENT_QueryRecordFile	70
3.10.2 CLIENT_DownloadByTimeEx	71
3.10.3 CLIENT_StopDownload	72
4 Callback Definition	73
4.1 fDisconnect	73
4.2 fHaveReConnect	73
4.3 fRealDataCallBackEx	74
4.4 pfAudioDataCallBack	74
4.5 fAnalyzerDataCallBack	75
4.6 fTimeDownLoadPosCallBack	75
4.7 fMessCallBack	76
4.8 Asynchronous Snapshot	77
Appendix 1 Cybersecurity Recommendations	78

1 Overview

1.1 General

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

The following are the main functions:

SDK initialization, device login, real-time monitoring, PTZ control, voice talk, alarm listening, smart subscription, record playback, record download and so on.

The development kit might be different dependent on the environment.

Table 1-1 Files of window 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	Header file
	dhconfigSDK.lib	Lib file
	dhconfigSDK.dll	Library file
Auxiliary library of playing (coding and decoding)	dhplay.dll	Playing library
	adpcmdecode.dll	Auxiliary library of playing
	postproc.dll	Auxiliary library of playing
	mpeg4enc.dll	Auxiliary library of playing
	mpeg4dec.dll	Auxiliary library of playing
	mp3dec.dll	Auxiliary library of playing
	mjpegdec.dll	Auxiliary library of playing
	h264dec.dll	Auxiliary library of playing
	amrdec.dll	Auxiliary library of playing
	aacdec.dll	Auxiliary library of playing
	DhDecode.dll	Auxiliary library of playing
	lvsDrawer.dll	Auxiliary library of playing
Auxiliary library of "avnetSDK.dll"	Infra.dll	Functional auxiliary library
	json.dll	Functional auxiliary library
	NetFramework.dll	Functional auxiliary library
	Stream.dll	Functional auxiliary library
	StreamSvr.dll	Functional auxiliary 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

Library type	Library file name	Library file description
Configuration library	avglobal.h	Header file
	dhconfigsdk.h	Header file
	libdhconfigsdk.so	Library file
Dependent library of "libavnet sdk.so"	libInfra.so	Functional auxiliary library
	libJson.so	Functional auxiliary library
	libNetFramework.so	Functional auxiliary library
	libStream.so	Functional auxiliary library



- The function library and configuration library are necessary libraries.
- The function library is the main body of SDK, which is used for interaction between client and products, remotely controls device, queries device data, configures device data information, and gets and handles the streams.
- 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 and collects the local audio for the functions such as monitoring, playback and voice talk.
- If the function library includes avnet sdk.dll or libavnet sdk.so, the corresponding dependent library is necessary.

1.2 Applicability

- Recommended memory: No less than 512 M
- Jdk version: jdk1.6; jdk1.8
- 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 and SUSE

1.3 Solutions of Library Loading Error

There are three kinds of dynamic libraries, Windows(.dll), and Linux(.so). Windows and Linux have 64-bit version and 32-bit version. And there are two main methods to call the call C ++ dynamic library, which are using the Java project directly and running the Java project as a jar package for other projects. During the loading of the library, the error of "Unable to find the dynamic library" will appear.

The root cause of "Unable to find the dynamic library" is the mismatch between the code path and the physical path. Compared with the Windows version, the dynamic library name of the Linux has a lib prefix. Therefore, when loading a dynamic library in the Linux environment, you need to pay attention to the lib prefix and add the lib prefix when stitching the dynamic library path. When using java.io.tmpdir to implement path mapping, note that this method has a lower

priority.

1.3.1 Using Java Project in Windows

Possible error information:

```
[Load dhnetsdk path : ./wrongpath/libs/win64/dhnetsdk]
Exception in thread "AWT-EventQueue-0" java.lang.UnsatisfiedLinkError: Unable to load
library './wrongpath/libs/win64/dhnetsdk': Unable to find specified module.
at com.sun.jna.NativeLibrary.loadLibrary(NativeLibrary.java:169)
at com.sun.jna.NativeLibrary.getInstance(NativeLibrary.java:242)
at com.sun.jna.Library$Handler.<init>(Library.java:140)
```

When the above error information appears, you can locate the error according to the following code and then modify the path of the dynamic library.

```
public interface NetSDKLib extends Library {
    NetSDKLib NETSDK_INSTANCE =
    (NetSDKLib)Native.loadLibrary(Utils.getLoadLibrary("dhnetsdk"), NetSDKLib.class);
    ~~~~ Utils.getLoadLibrarycode is as follows
    // Get load library
    public static String getLoadLibrary(String library) {
        if (isChecking()) {
            return null;
        }
        String loadLibrary = "";
        String osPrefix = getOsPrefix();
        if(osPrefix.toLowerCase().startsWith("win32-x86")) {
            loadLibrary = "./libs/win32/";
        } else if(osPrefix.toLowerCase().startsWith("win32-amd64")) {
            loadLibrary = "./libs/win64/";
        } else if(osPrefix.toLowerCase().startsWith("linux-i386")) {
            loadLibrary = "";
        } else if(osPrefix.toLowerCase().startsWith("linux-amd64")) {
            loadLibrary = "";
        }

        System.out.printf(" Load library[%s]\n", loadLibrary + library);
        return loadLibrary + library;
    }
}
```

Solution

Step 1 Create a wrongpath folder as the parent folder of libs.

Step 2 Change if else in the code to:

```
else if(osPrefix.toLowerCase().startsWith("win32-amd64")) {  
    loadLibrary = "./libs/win64/";  
}
```

Note

Windows platform is less likely to have errors, and here is just an example. Note that the platform must correspond to the dynamic library, and the dynamic library suffix for Windows is .dll.

1.3.2 Using Java Project in Linux

Running the Example Code by Script Code in Linux (the script code path is run.sh in the root directory)

```
[user@localhost ~]$ cd JNADemo/  
[user@localhost JNADemo]$ ls  
AnalyzerPicture  bin    DemoSrc  device.xlsx  doc    jna_src  libs    PlateNumber  
README.txt      run.sh  sdk_log  src  
[user@localhost JNADemo]$ ./run.sh -example Preview  
--> linux 64 System.  
--> ClassPath: ../libs/jna.jar:../libs/examples.jar  
--> Bin ../bin  
com/netsdk/demo/example/TrafficEvent.java:40: warning: Unsafe is internal proprietary API  
and may be removed in a future release  
import sun.misc.Unsafe;                                ^  
Note: Some input files use or override a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
Note: Some input files use unchecked or unsafe operations.  
Note: Recompile with -Xlint:unchecked for details.  
4 warnings  
--> path: pwd  
--> example name: com/netsdk/demo/example/Preview  
Load library./home/user/JNADemo/libs/linux64/libdhnetsdk.so]  
Load library./home/user/JNADemo/libs/linux64/libdhconfigsdk.so]  
NetSDK Version [35041138]  
RealPlay Window Closing  
Stop Tasks!  
Stop Tasks!  
/home/user/JNADemo/src
```

```
[user@localhost JNADemo]$
```

Part of Run.sh Script Code

```
## Specify the path of the library    add dynamic link library
if [ $(getconf LONG_BIT) = '64' ]; then
    echo "--> linux 64 System."
    export LD_LIBRARY_PATH=./libs/linux64
else
    echo "--> linux 32 System."
    export LD_LIBRARY_PATH=./libs/linux32
fi
```

Note

Similar to windows. However, if the error is not found, check whether the PATH path introduced in the script is consistent with the system path when the dynamic library is correct.

Solution

Select one of the methods to solve.

- Method 1: Enter "export LD_LIBRARY_PATH = \$ LD_LIBRARY_PATH: / XXX" in the terminal, and then the current terminal takes effect
- Method 2: Temporary folder loading method—java.io.tmpdir, this method is suitable for multiple platforms. It should be noted that loading dynamic libraries in this way has a lower priority. You must change the LoadLibrary parameter in public static String getLoadLibrary (String library) to "".
 - ◇ Copy the required dynamic library to a folder, such as D: / win64 /.
 - ◇ Use static statement block to call: (path should be consistent)

```
static{
    System.setProperty("java.io.tmpdir", "D:/win64");
}
```

1.3.3 Using Project as a jar Package

The advantage of having the dynamic library built into the jar package is that as long as the jar package is exported, the jar can be executed directly regardless of the differences between the platforms.

Here is a way to make the build-in jar package. You need to write the dynamic library of jar package into the local temporary folder (java.io.tmpdir), read the dynamic library from the local machine, and then load it to memory. The following is the example code.

```
public interface NetSDKLib extends Library {

NetSDKLib                                NETSDK_INSTANCE                                =
(NetSDKLib)NetSDKLib.LoadHelper.loadDll("dhnetsdk",NetSDKLib.class);
```

```

NetSDKLib                                CONFIG_INSTANCE                                =
(NetSDKLib)NetSDKLib.LoadHelper.loadDll("dhconfigsdk",NetSDKLib.class);

    public static class LoadHelper{
        public synchronized static Object loadDll(String libName, Class<?> className){
            String libExtension=".dll",systemType="win";

            switch(Platform.getOSType()){
                case Platform.LINUX:
                    libExtension=".so";
                    systemType="linux";
            }
            String libFullName = libName + libExtension;
            String nativeTempDir = System.getProperty("java.io.tmpdir");
// Get the temporary directory cached by the operating system

            InputStream in = null;
            BufferedInputStream reader = null;
            FileOutputStream writer = null;
            File extractedLibFile = new File(nativeTempDir + File.separator + libFullName);
//Temporary files

            if (!extractedLibFile.exists()) {
                try {
                    systemType+=System.getProperty("sun.arch.data.model");
                    in      =      className.getResourceAsStream("/"+systemType+"/"+
libFullName);
                    if (in == null)
                        in = className.getResourceAsStream(libFullName);
                    reader = new BufferedInputStream(in);
                    writer = new FileOutputStream(extractedLibFile);
                    byte[] buffer = new byte[1024];
                    while (reader.read(buffer) > 0) {
                        writer.write(buffer);    // Write dll/so/dylibto the temporary file
                        buffer = new byte[1024];
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (in != null)

```

```

        try {
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        if (writer != null)
            try {
                writer.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
}

String temp=extractedLibFile.toString();
String dllName=temp.substring(0,temp.indexOf("."));
System.out.println("[Load sdk Path : "+temp+"]");
return Native.loadLibrary(dllName, className);
}
}

```

1.4 Upgrade early jna to latest version

Step 1 Upgrade jna package in the libs/ path to the latest version.

Step 2 Configure the latest version of jna to make it suitable for the current environment.

Step 3 Change the library loading method of NetSDKLib encapsulation class to the following.

```

NetSDKLib NETSDK_INSTANCE = (NetSDKLib)Native.load(Utils.getLoadLibrary("dhnetsdk"),
NetSDKLib.class);

```

Step 4 In the NativeString class, change the parameters in the Pointer.getString method and Pointer.setString method to pointer.getString (0) and pointer.setString (0, string).



After upgrading the jna, if there is a callback function in the running demo and StdCallCallback is inherited, you need to comment out StdCallCallback; otherwise it will crash when running under the Linux environment.

2 Function Modules

There are 10 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 cleanup interface to release SDK resource.

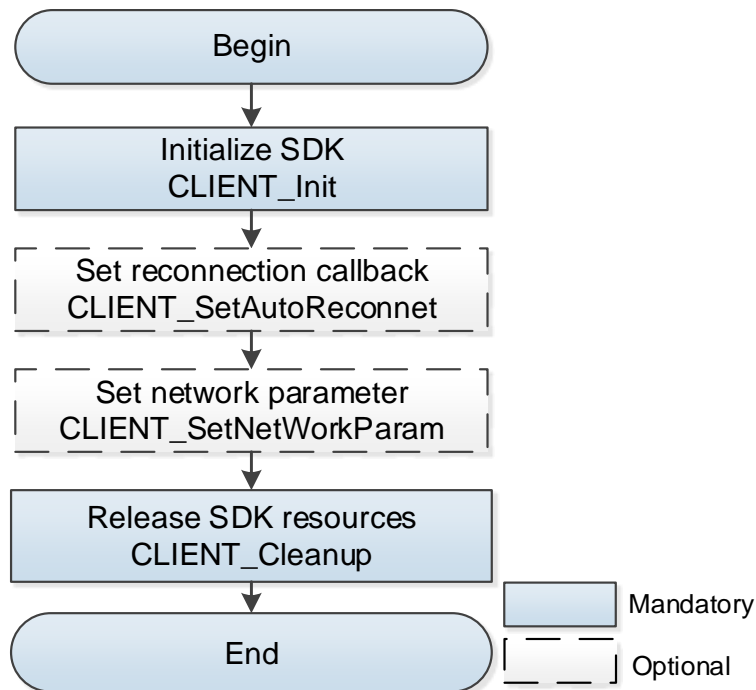
2.1.2 Interface Overview

Table 2-1 Interfaces of 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 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 resume after reconnection is successful.
- Dynamic library loading: If there is an error "Unable to load library './wrongpath/libs/win64/dhnetSDK': The specified module cannot be found" when loading the dynamic library, usually the path does not match. You need to adjust the path of the dynamic library or modify the code based on the error codes. This problem is more

common when packaging the whole project and providing the jar package to other projects. Because this problem is related to the use of the platform and the project, it cannot be generalized and requires specific analysis.

For example, if you want to use the project directly on the Linux platform, you can load the dynamic library path into the dynamic library search path by the following ways:

- ◇ Enter “export LD_LIBRARY_PATH = \$ LD_LIBRARY_PATH: / XXX” in the terminal. The current terminal takes effect.
- ◇ Modify “~ / .bashrc” or “~ / .bash_profile”, and add “export LD_LIBRARY_PATH = \$ LD_LIBRARY_PATH: / XXX” in the last line. After saving, you can use “source.bashrc” to execute the file. The current user takes effect.
- ◇ Modify “/ etc / profile”, and then add “export LD_LIBRARY_PATH = \$ LD_LIBRARY_PATH: / XXX”. After saving, you can use “source” to execute the file. Effective for all users.

2.1.4 Example Code

```
import java.io.File;

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

import com.sun.jna.ptr.IntByReference;

/**
 * Implement login interface
 * Mainly are initialization, login and logout functions.
 */
public class LoginModule {

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

    // Login handle
    public static LLong m_hLoginHandle = new LLong(0);

    private static boolean blnit        = false;
    private static boolean bLogopen     = false;

    //Initialize
    public static boolean init(NetSDKLib.fDisconnect disconnect, NetSDKLib.fHaveReConnect
haveReConnect) {
        blnit = netsdk.CLIENT_Init(disconnect, null);
```

```

if(!bInit) {
    System.out.println("Initialize SDK failed");
    return false;
}

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

// Set the callback of reconnection after disconnection. After setting, the SDK will automatically
reconnect when device disconnects.

// This operation is optional but recommended.
netsdk.CLIENT_SetAutoReconnect(haveReConnect, null);

// (Optional) Set login timeout and login times
int waitTime = 5000; //Set the timeout of request response as 5 seconds
int tryTimes = 1;    // Try to establish a link once during login
netsdk.CLIENT_SetConnectTime(waitTime, tryTimes);

// Set other network parameters, such as nWaittime of NET_PARAM, member of
nConnectTryNum and CLIENT_SetConnectTime.

// (Optional) Set the login timeout of device and login times having same meaning.
NetSDKLib.NET_PARAM netParam = new NetSDKLib.NET_PARAM();
netParam.nConnectTime = 10000;    // Timeout of trying to establish a link during login
netParam.nGetConnInfoTime = 3000; // Timeout of setting subconnection
netsdk.CLIENT_SetNetworkParam(netParam);

```

```

        return true;
    }

    // Clean up environment
    public static void cleanup() {
        if(bLogopen) {
            netsdk.CLIENT_LogClose();
        }

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

```

2.2 Device Login

2.2.1 Introduction

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

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

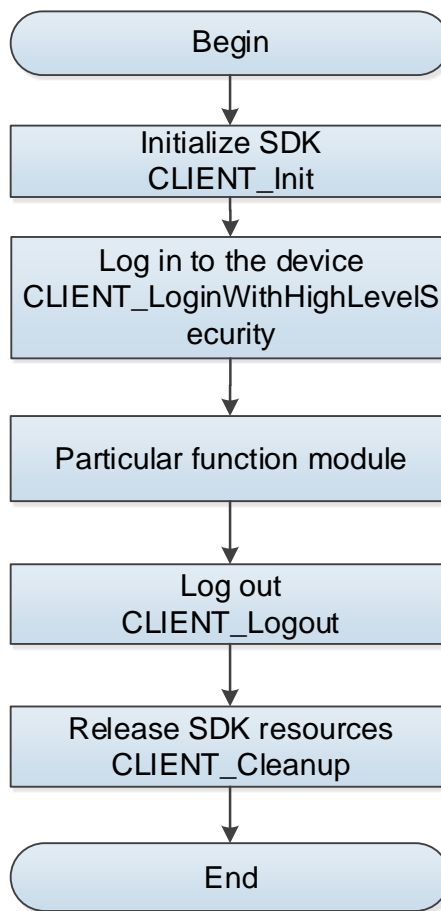
2.2.2 Interface Overview

Table 2-2 Interfaces of device login

Interface	Implication
CLIENT_LoginWithHighLevelSecurity	High-security login.
CLIENT_Logout	Logout.

2.2.3 Process

Figure 2-2 Process of login



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 After successful login, you can realize the required function module.
- Step 4 After using the function module, call **CLIENT_Logout** to log out of 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 log in for 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.
- Duplicate handles: It is normal that the login handle is the same as the existed handle. For example, log in to device A and get handle loginIDA. However, if you log out of loginIDA and then log in, you may get LoginIDA again. But the duplicate handles do not occur throughout the lifetime of the handle.
- Logout: The interface will release the opened functions internally, but it is not suggested to rely on the cleaning up function. 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. For the common error code, see Table 2-3.
- Multi-device login: After the SDK is initialized, you can log in to multiple devices, but the corresponding login handle and login information need to be adjusted.

Table 2-3 Error code and meaning

Error code	Meaning
1	Password is wrong
2	User name does not exist
3	Login timeout Evasion example code is as follows: NET_PARAM stuNetParam = {0}; stuNetParam.nWaittime = 8000; // unit ms stunetParam.nGetConnInfoTime = 3000; // Set connection timeout CLIENT_SetNetworkParam (&stuNetParam);
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

2.2.4 Example Code

```
import java.io.File;

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

import com.sun.jna.ptr.IntByReference;

public class LoginModule {

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

```

//Initialize SDK and skip SDK cleanup

// Device information
public static NetSDKLib.NET_DEVICEINFO_Ex m_stDeviceInfo = new
NetSDKLib.NET_DEVICEINFO_Ex();

// Login handle
public static LLong m_hLoginHandle = new LLong(0);

// Log in to device
public static boolean login(String m_strIp, int m_nPort, String m_strUser, String m_strPassword) {

    IntByReference nError = new IntByReference(0);
    m_hLoginHandle = netsdk.CLIENT_LoginEx2(m_strIp, m_nPort, m_strUser, m_strPassword, 0,
null, m_stDeviceInfo, nError);
    if(m_hLoginHandle.longValue() == 0) {
        System.err.printf("Login Device[%s] Port[%d]Failed. %s\n", m_strIp, m_nPort,
ToolKits.getErrorCodePrint());
    } else {
        System.out.println("Login Success [ " + m_strIp + " ]");
    }

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

// Log out of device
public static boolean logout() {
    if(m_hLoginHandle.longValue() == 0) {
        return false;
    }

    boolean bRet = netsdk.CLIENT_Logout(m_hLoginHandle);
    if(bRet) {
        m_hLoginHandle.setValue(0);
    }

    return bRet;
}
}

```

2.3 Real-time Monitoring

2.3.1 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.

2.3.2 Interface Overview

Table 2-4 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_SetRealDataCallBackEx	Set real-time monitoring data callback function extension interface.

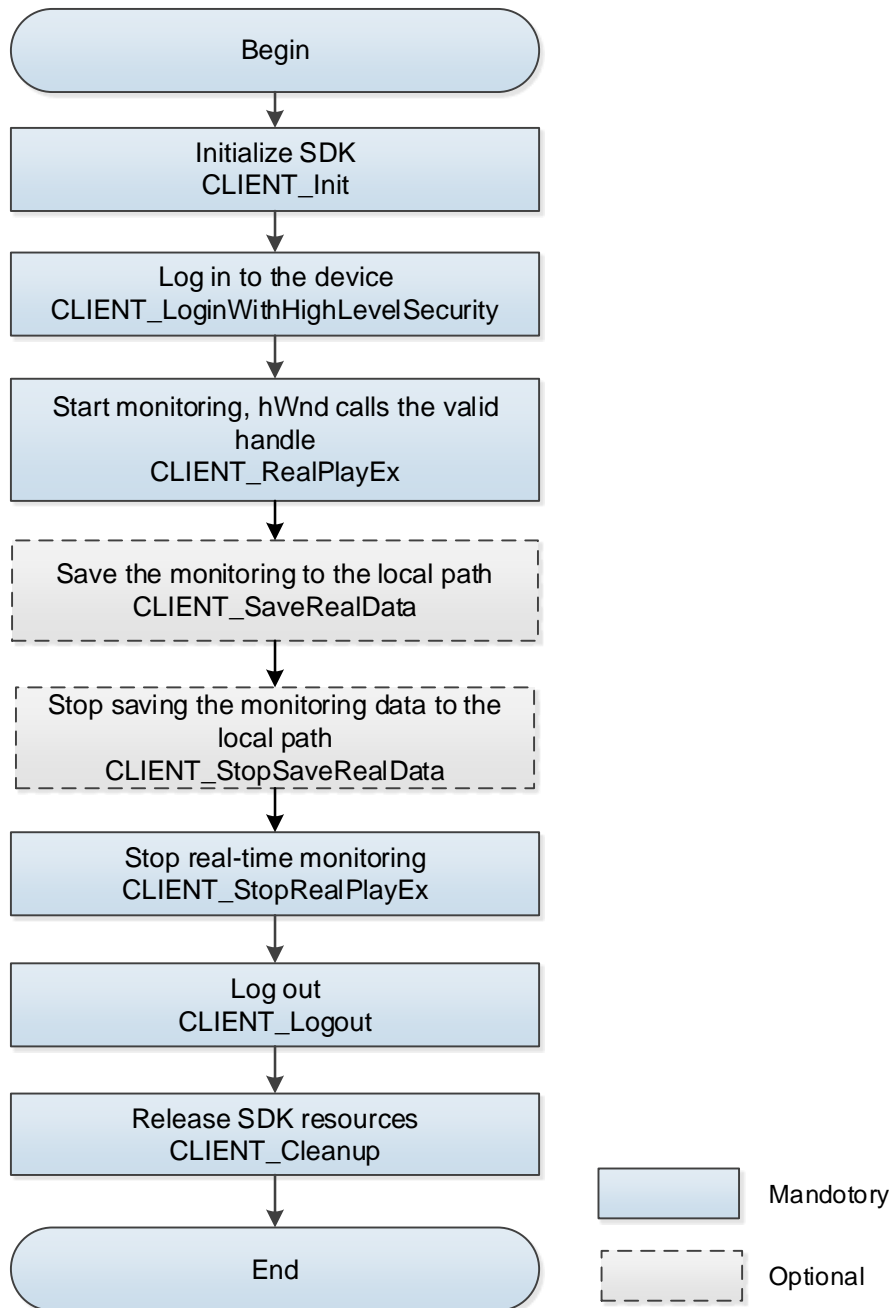
2.3.3 Process

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

2.3.3.1 SDK Decoding Play

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

Figure 2-3 Process of playing by SDK decoding library



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to 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 After using the real-time function, call **CLIENT_StopRealPlayEx** to stop real-time monitoring.
- Step 7 After using the function module, call **CLIENT_Logout** to log out of the device.
- Step 8 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**.

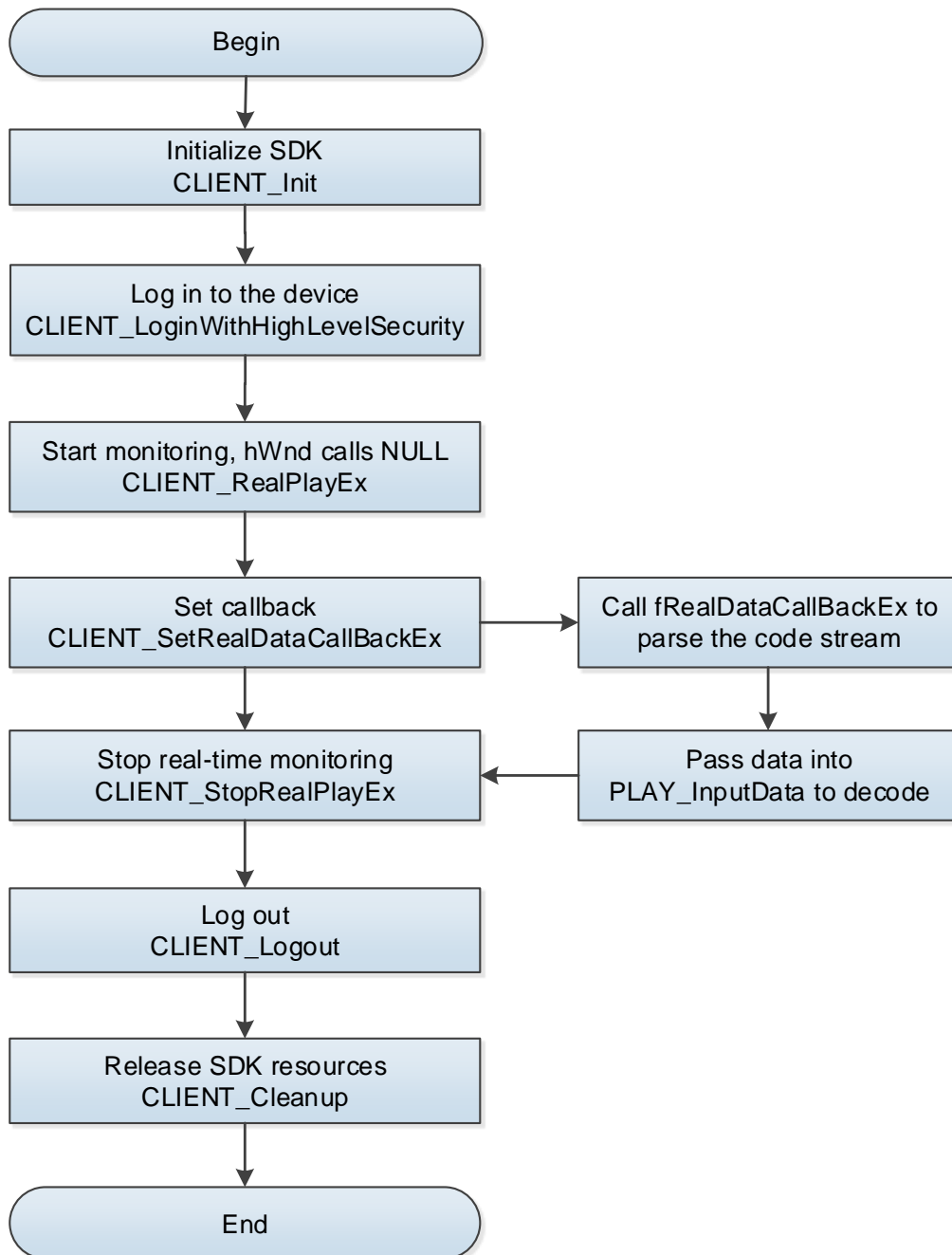
```
NET_PARAM stuNetParam = new NET_PARAM();  
stuNetParam.nGetConnInfoTime = 5000; 0, default is 1000 ms.  
CLIENT_SetNetworkParam (stuNetParam);
```

- 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 "2.3.3.2 Call Third Party Play Library."

2.3.3.2 Call Third Party Play Library

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

Figure 2-4 Process of calling third party play library



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 After successful login, call **CLIENT_RealPlayEx** to enable real-time monitoring. The parameter hWnd is NULL.
- Step 4 Call **CLIENT_SetRealDataCallBackEx** 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 log out of 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 buffer 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 move to the next video data only after returning from you. It is not recommended for you to consume time for the unnecessary operations; otherwise the performance could be affected.

2.3.4 Example Code

2.3.4.1 SDK Decoding Play

```
import java.awt.Panel;

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

import com.sun.jna.Native;

/**
 * Implement live interface
 * Mainly are streaming starting function and streaming stopping function.
 */
public class RealPlayModule {
    // Start live view
    public static LLong startRealPlay(int channel, int stream, Panel realPlayWindow) {
        LLong m_hPlayHandle =
LoginModule.netsdk.CLIENT_RealPlayEx(LoginModule.m_hLoginHandle, channel,
Native.getComponentPointer(realPlayWindow), stream);

        if(m_hPlayHandle.longValue() == 0) {
            System.err.println("failed to real-time monitoring, and the error code " +
ToolKits.getErrorCodePrint());
        } else {
            System.out.println("Success to start realplay");
        }
    }

    // Customize stream to save file. Do this operation when you need to save the video.
    String outFile="example/outputfile";
```

```

LoginModule.netsdk.CLIENT_SaveRealData(m_hPlayHandle,outFile);
    }
    return m_hPlayHandle;
}

// Stop live view
public static void stopRealPlay(LLong m_hPlayHandle) {
    if(m_hPlayHandle.longValue() == 0) {
        return;
    }
}
// Close file saving
LoginModule.netsdk.CLIENT_StopSaveRealData(m_hPlayHandle);
    boolean bRet = LoginModule.netsdk.CLIENT_StopRealPlayEx(m_hPlayHandle);
    if(bRet) {
        m_hPlayHandle.setValue(0);
    }
}
}

```

2.3.4.2 Call Play Library

```

public class RealPlayModule {
    class DataCallBackEx implements NetSDKLib.fRealDataCallBackEx{
        @Override
        public void invoke(LLong lRealHandle, int dwDataType, Pointer pBuffer,
            int dwBufSize, int param, Pointer dwUser) {
            // TODO
            // Call PlaySDK interface to get stream data from device. For more details, see demo
            source code of SDK monitoring.

        }
    }

    private DataCallBackEx m_DataCallBackEx = new DataCallBackEx();
    public LLong startRealPlay(int channel, int stream, Panel realPlayWindow) {
        LLong m_hPlayHandle =
LoginModule.netsdk.CLIENT_RealPlayEx(LoginModule.m_hLoginHandle, channel,
Native.getComponentPointer(realPlayWindow), stream);

        LoginModule.netsdk.CLIENT_SetRealDataCallBackEx(m_hPlayHandle,m_DataCallBackEx,
null, 0x00000001);
    }
}

```

```

        if(m_hPlayHandle.longValue() == 0) {
            System.err.println("failed to real-time monitoring, and the error code" +
ToolKits.getErrorCodePrint());
        } else {
            System.out.println("Success to start realplay");
        }

        return m_hPlayHandle;
    }

    public void stopRealPlay(LLong m_hPlayHandle) {
        if(m_hPlayHandle.longValue() == 0) {
            return;
        }
        boolean bRet = LoginModule.netsdk.CLIENT_StopRealPlayEx(m_hPlayHandle);
        if(bRet) {
            m_hPlayHandle.setValue(0);
        }
    }
}

```

2.4 Video Snapshot

2.4.1 Introduction

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

- Synchronous snapshot: Call the SDK interface which sends the snapshot command to the device. The device will capture the current image and send to SDK through network, and then SDK returns the image data to you
- Asynchronous snapshot: Call the SDK interface and set snapshot callback so that the captured image data shows in callback function. At the same time, call asynchronous snapshot interface to snapshot.
- Local snapshot: When the monitoring is opened, you can save the monitoring data in the picture format which is the frame information that does not have network interaction with the device.

2.4.2 Interface Overview

Table 2-5 Interfaces of video snapshot

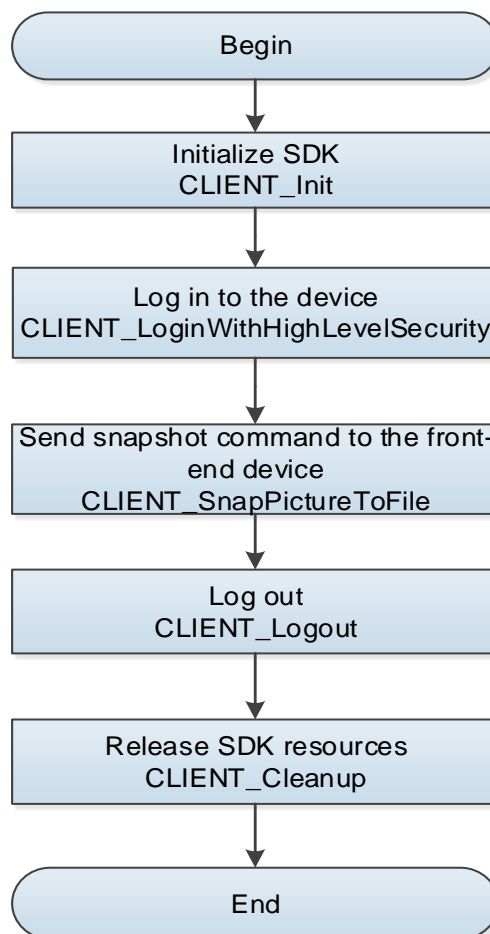
Interface	Implication
CLIENT_SnapPictureToFile	Snap picture and send to the user.
CLIENT_CapturePictureEx	Local snap and the parameter could be the handle of monitoring or playback.
CLIENT_SetSnapRevCallBack	Set snapshot callback to implement fSnapRev interface.
CLIENT_SnapPictureEx	Asynchronous snapshot which is suitable for non-intelligent traffic devices and parking lot devices, such as IPC and speed dome.

2.4.3 Process

Video snapshot is consisted of synchronous snapshot, asynchronous snapshot and local snapshot.

2.4.3.1 Synchronous Snapshot

Figure 2-5 Process of synchronous snapshot



Process Description

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

Notes for Process

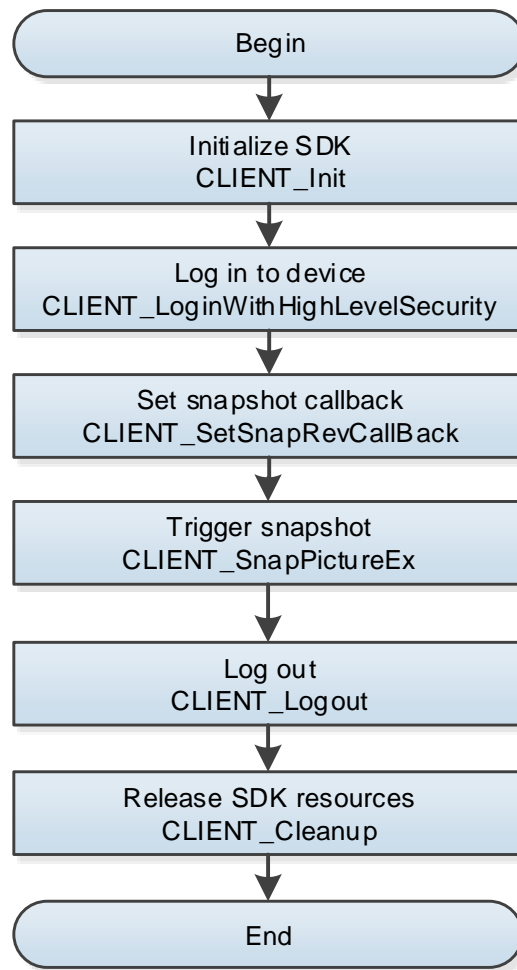
- Picture size limit: SDK allocates the fixed memory to receive the picture data returned from the device. If the picture is larger than the fixed memory, SDK will return the truncated data.
- SDK provides the interface to modify the default memory. If the picture (for example, the high definition picture) 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 = new NET_PARAM();  
stuNetParam. nPicBufSize = 4000*1024*1024; nPicBufSize is 2M by default  
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 network snapshot such as quality and definition. However, if you are satisfied with the default configurations, do not modify them.
- 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).

2.4.3.2 Asynchronous Snapshot

Figure 2-6 Process of asynchronous snapshot

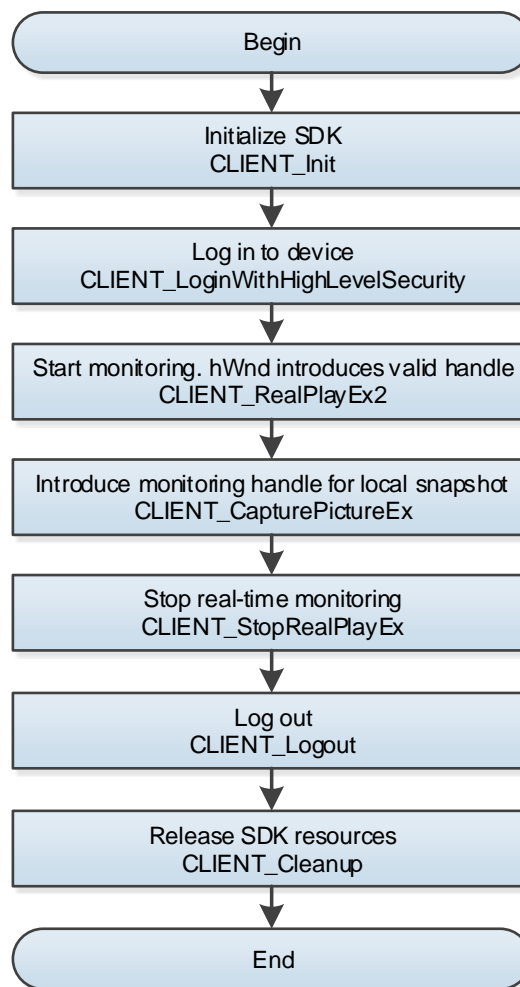


Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 Call **CLIENT_SetSnapRevCallBack** to set snapshot callback.
- Step 4 Call **CLIENT_SnapPictureEx** to trigger snapshot and then analyze captured image data in the callback function.
- Step 5 Call **CLIENT_Logout** to log out of the device.
- Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

2.4.3.3 Local Snapshot

Figure 2-7 Process of local snapshot



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to 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 log out of the device.
- Step 7 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

2.4.4 Example Code

```
/*  
 * Example of synchronous snapshot  
 */  
NetSDKLib.NET_IN_SNAP_PIC_TO_FILE_PARAM snapParamIn = new  
NetSDKLib.NET_IN_SNAP_PIC_TO_FILE_PARAM();
```

```

NetSDKLib.NET_OUT_SNAP_PIC_TO_FILE_PARAM snapParamOut = new
NetSDKLib.NET_OUT_SNAP_PIC_TO_FILE_PARAM(1024 * 1024);
snapParamIn.stuParam.Channel = 0;
snapParamIn.stuParam.Quality = 3;
snapParamIn.stuParam.ImageSize = 1; // 0: QCIF,1: CIF,2: D1
snapParamIn.stuParam.mode = 0;
// -1: Stop snapshot, 0: Require for 1 frame, 1: Send request on time, 1: Require continuously
snapParamIn.stuParam.InterSnap = 5;
snapParamIn.stuParam.CmdSerial = serialNum;
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyMMddHHmmss");
final String fileName = "SyncSnapPicture_" + dateFormat.format(new Date()) + "_" + serialNum + ".jpg";
System.arraycopy(fileName.getBytes(), 0, snapParamIn.szFilePath, 0, fileName.getBytes().length);
final int timeOut = 5000; // 5 second
if (!netsdkApi.CLIENT_SnapPictureToFile(loginHandle, snapParamIn, snapParamOut, timeOut)) {
    System.err.printf("CLIENT_SnapPictureEx Failed ! Last Error[%x]\n",
        netsdkApi.CLIENT_GetLastError());
    return;
}
System.out.println("CLIENT_SnapPictureToFile Success. " + new File(fileName).getAbsolutePath());

/*
*Example of local snapshot
*/
//Live view
int playType = NetSDKLib.NET_RealPlayType.NET_RType_Realplay; // Live view
m_hRealPlayHandle = netsdkApi.CLIENT_RealPlayEx(m_hLoginHandle, channel,
Native.getComponentPointer(realplayPanel), playType);
if (m_hRealPlayHandle.longValue() == 0) {
    System.err.println("Failed to start real-time monitoring , and error code " +
ToolKits.getErrorCode());
    return false;
} else {
    System.out.println("Success to start realplay");
}
// Local snapshot
if (!LoginModule.netsdk.CLIENT_CapturePictureEx(hPlayHandle, picFileName,
NetSDKLib.NET_CAPTURE_FORMATS.NET_CAPTURE_JPEG))
{
    System.err.printf("CLIENT_CapturePicture Failed!" + ToolKits.getErrorCodePrint());
} else {

```

```

        System.out.println("CLIENT_CapturePicture success");
    }

// Stop live view
if(m_hRealPlayHandle.longValue() != 0) {
    netsdkApi.CLIENT_StopRealPlayEx(m_hRealPlayHandle);
}

/*
 * Example of asynchronous snapshot
 */
    /// Set snapshot callback: Pictures are mainly returned from SnapCallback.getInstance()
    invoke.
    netsdkApi.CLIENT_SetSnapRevCallBack(SnapCallback.getInstance(), null);

    NetSDKLib.SNAP_PARAMS snapParam = new NetSDKLib.SNAP_PARAMS();
    snapParam.Channel = 0; // Snapshot channel
    snapParam.mode = 0; // Require for 1 frame
    snapParam.CmdSerial = serialNum ++; // Require for serial number, and the valid rage from 0
    through 635535. The serial number out of the rage will be truncated.
    /// Trigger snapshot
    if (!netsdkApi.CLIENT_SnapPictureEx(loginHandle, snapParam , null)) {
        System.err.printf("CLIENT_SnapPictureEx Failed ! Last Error[%x]\n",
            netsdkApi.CLIENT_GetLastError());
        return;
    }
    // Ensure the generation of image data
    try {
        synchronized (SnapCallback.class) {
            SnapCallback.class.wait(3000L); // Wait for 3 seconds by default, to prevent that the
            callback is not triggered and then device freezes when the device is disconnected and then.
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    System.out.println("--> " + Thread.currentThread().getName() + " CLIENT_SnapPictureEx Success." +
        System.currentTimeMillis());

```

2.5 PTZ Control

2.5.1 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 to 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.

2.5.2 Interface Overview

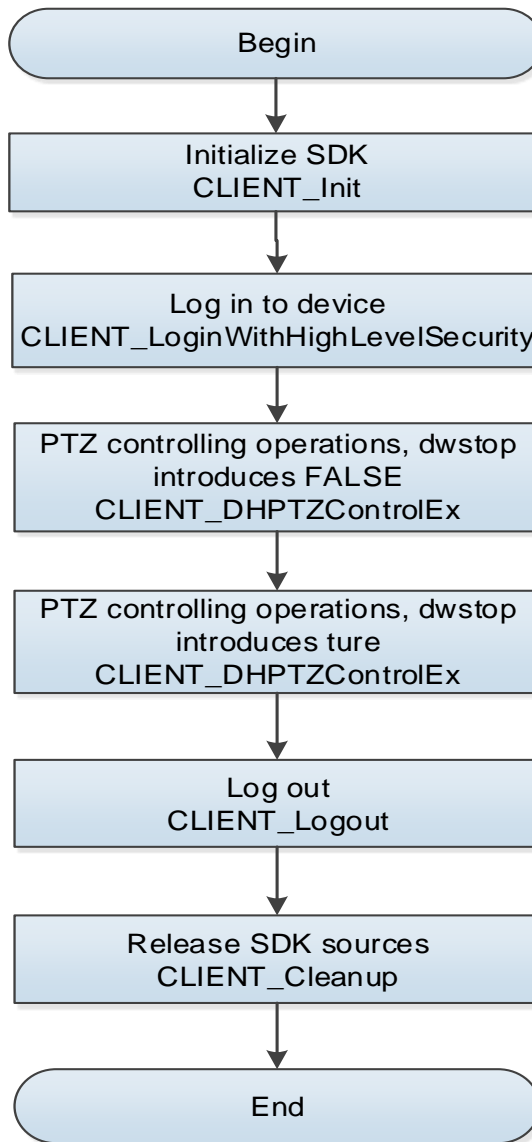
Table 2-6 Interface of PTZ control

Interface	Implication
CLIENT_DHPTZControlEx	PTZ control extension interface

2.5.3 Process

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

Figure 2-8 Process of PTZ control



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 Call **CLIENT_DHPTZControlEx** to operate the PTZ according to the situation. Different PTZ commands 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 "2.5.4 Example Code."
- Step 4 After using the function module, call **CLIENT_Logout** to log out of 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 (NetSDK_Chn_java\src\main\java\com\netsdk\demo\frame\PTZControl.java).

2.5.4 Example Code

```
/**
 * Implement interface of PTZ control
 * Mainly are direction control, focus, zoom and iris.
 */
public class PtzControlModule {
    /**
     * Up
     */
    public static boolean ptzControlUpStart(int nChannelID, int IParam1, int IParam2) {
        return LoginModule.netsdk.CLIENT_DHPTZControlEx(LoginModule.m_hLoginHandle,
nChannelID,

        NetSDKLib.NET_PTZ_ControlType.NET_PTZ_UP_CONTROL,
                                IParam1, IParam2, 0, 0);
    }
    public static boolean ptzControlUpEnd(int nChannelID) {
        return LoginModule.netsdk.CLIENT_DHPTZControlEx(LoginModule.m_hLoginHandle,
nChannelID,

NetSDKLib.NET_PTZ_ControlType.NET_PTZ_UP_CONTROL,
                                0, 0, 0, 1);
    }
    .....
    // Call CLIENT_DHPTZControlEx to implement other functions which are the same as Up's. However,
    the input type parameter of NetSDKLib.NET_PTZ_ControlType is different.
}
```

2.6 Voice Talk

2.6.1 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.

2.6.2 Interface Overview

Table 2-7 Interfaces of voice talk

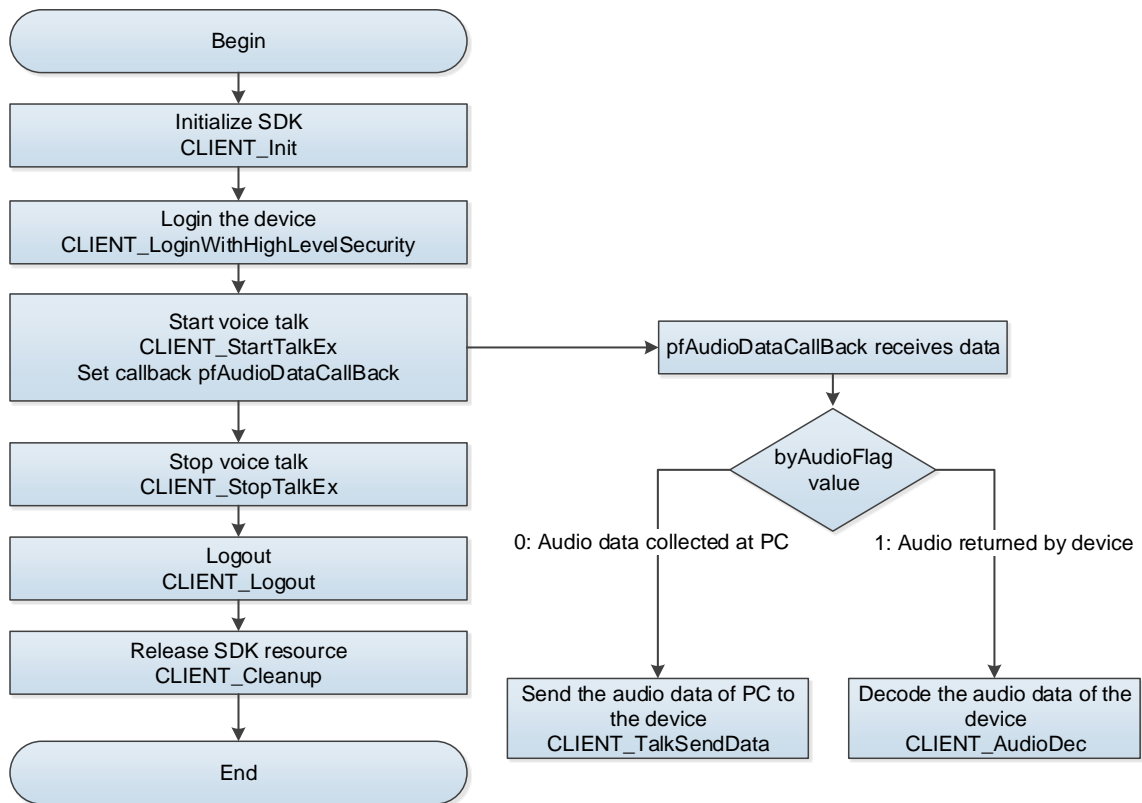
Interface	Implication
CLIENT_StartTalkEx	Start voice talk
CLIENT_StopTalkEx	Stop voice talk
CLIENT_TalkSendData	Send voice data to the device
CLIENT_AudioDecEx	Decode audio data (valid only in Windows system)

2.6.3 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.

Figure 2-9 Process of voice talk



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to 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_StopTalkEx** to stop voice talk.
- Step 6 After using the function module, call **CLIENT_Logout** to log out of the device.
- Step 7 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 (NetSDK_Chn_java\src\main\java\com\netsdk\demo\frame\Talk.java). 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.

2.6.4 Example Code

```
/**
 * Implement voice talk
 * Implement start, stop and data callback of voice talk.
 */
public class TalkModule {
    public static LLong m_hTalkHandle = new LLong(0); // Voice talk handle
    private static boolean m_bRecordStatus = false; // Is recording?

    /**
     * Start voice talk
     */
    public static boolean startTalk(int transferType, int chn) {
        // Set the encoding format of voice talk
        NetSDKLib.NETDEV_TALKDECODE_INFO talkEncode = new
NetSDKLib.NETDEV_TALKDECODE_INFO();
        talkEncode.encodeType = NetSDKLib.NET_TALK_CODING_TYPE.NET_TALK_PCM;
        talkEncode.dwSampleRate = 8000;
        talkEncode.nAudioBit = 16;
        talkEncode.nPacketPeriod = 25;
        talkEncode.write();

        if(LoginModule.netsdk.CLIENT_SetDeviceMode(LoginModule.m_hLoginHandle,
NetSDKLib.EM_USEDEV_MODE.NET_TALK_ENCODE_TYPE, talkEncode.getPointer())) {
            System.out.println("Set Talk Encode Type Succeed!");
        } else {
            System.err.println("Set Talk Encode Type Failed!" + ToolKits.getErrorCodePrint());
            return false;
        }

        // Set the speak parameter of voice talk
        NetSDKLib.NET_SPEAK_PARAM speak = new NetSDKLib.NET_SPEAK_PARAM();
        speak.nMode = 0;
        speak.bEnableWait = false;
        speak.nSpeakerChannel = 0;
        speak.write();

        if (LoginModule.netsdk.CLIENT_SetDeviceMode(LoginModule.m_hLoginHandle,
NetSDKLib.EM_USEDEV_MODE.NET_TALK_SPEAK_PARAM, speak.getPointer())) {
            System.out.println("Set Talk Speak Mode Succeed!");
        } else {
            System.err.println("Set Talk Speak Mode Failed!" + ToolKits.getErrorCodePrint());
        }
    }
}
```

```

        return false;
    }

    // Set the voice talk in transfer mode
    NetSDKLib.NET_TALK_TRANSFER_PARAM talkTransfer = new
NetSDKLib.NET_TALK_TRANSFER_PARAM();
    talkTransfer.bTransfer = transferType;
    talkTransfer.write();

    if(LoginModule.netsdk.CLIENT_SetDeviceMode(LoginModule.m_hLoginHandle,
NetSDKLib.EM_USEDEV_MODE.NET_TALK_TRANSFER_MODE, talkTransfer.getPointer())) {
        System.out.println("Set Talk Transfer Mode Succeed!");
    } else {
        System.err.println("Set Talk Transfer Mode Failed!" + ToolKits.getErrorCodePrint());
        return false;
    }

    if (talkTransfer.bTransfer == 1) { // Set transfer channel for transfer mode
        IntByReference nChn = new IntByReference(chn);
        if(LoginModule.netsdk.CLIENT_SetDeviceMode(LoginModule.m_hLoginHandle,
NetSDKLib.EM_USEDEV_MODE.NET_TALK_TALK_CHANNEL, nChn.getPointer())) {
            System.out.println("Set Talk Channel Succeed!");
        } else {
            System.err.println("Set Talk Channel Failed!" + ToolKits.getErrorCodePrint());
            return false;
        }
    }

    m_hTalkHandle = LoginModule.netsdk.CLIENT_StartTalkEx(LoginModule.m_hLoginHandle,
AudioDataCB.getInstance(), null);
    if(m_hTalkHandle.longValue() == 0) {
        System.err.println("Start Talk Failed!" + ToolKits.getErrorCodePrint());
        return false;
    } else {
        System.out.println("Start Talk Success");
        if(LoginModule.netsdk.CLIENT_RecordStart()){
            System.out.println("Start Record Success");
            m_bRecordStatus = true;
        } else {
            System.err.println("Start Local Record Failed!" + ToolKits.getErrorCodePrint());
            stopTalk();
            return false;
        }
    }
}

```

```

        return true;
    }
    /**
     * Stop voice talk
     */
    public static void stopTalk() {
        if(m_hTalkHandle.longValue() == 0) {
            return;
        }

        if (m_bRecordStatus){
            LoginModule.netsdk.CLIENT_RecordStop();
            m_bRecordStatus = false;
        }
        if(LoginModule.netsdk.CLIENT_StopTalkEx(m_hTalkHandle)) {
            m_hTalkHandle.setValue(0);
        }else {
            System.err.println("Stop Talk Failed!" + ToolKits.getErrorCodePrint());
        }
    }
    /**
     * Data callback of voice talk
     */
    private static class AudioDataCB implements NetSDKLib.pfAudioDataCallBack {

        private AudioDataCB() {}
        private static AudioDataCB audioCallBack = new AudioDataCB();
        public static AudioDataCB getInstance() {
            return audioCallBack;
        }

        public void invoke(LLong lTalkHandle, Pointer pDataBuf, int dwBufSize, byte byAudioFlag, Pointer
dwUser){
            if(lTalkHandle.longValue() != m_hTalkHandle.longValue()) {
                return;
            }
            if (byAudioFlag == 0) { // Send the sound card data which is detected by the local PC to the
device

                LLong lSendSize = LoginModule.netsdk.CLIENT_TalkSendData(m_hTalkHandle,
pDataBuf, dwBufSize);

```

```

        if(!SendSize.longValue() != (long)dwBufSize) {
            System.err.println("send incomplete" + !SendSize.longValue() + ":" + dwBufSize);
        }
    }else if (byAudioFlag == 1) { // Send the voice talk data which is sent by the device to SDK ,
to decode and playback.
        LoginModule.netsdk.CLIENT_AudioDecEx(m_hTalkHandle, pDataBuf, dwBufSize);
    }
}
}
}

```

2.7 Alarm Listening

2.7.1 Introduction

Alarm listening is the function to analyze real-time stream by smart devices. When the set event occurs, alarm triggers.

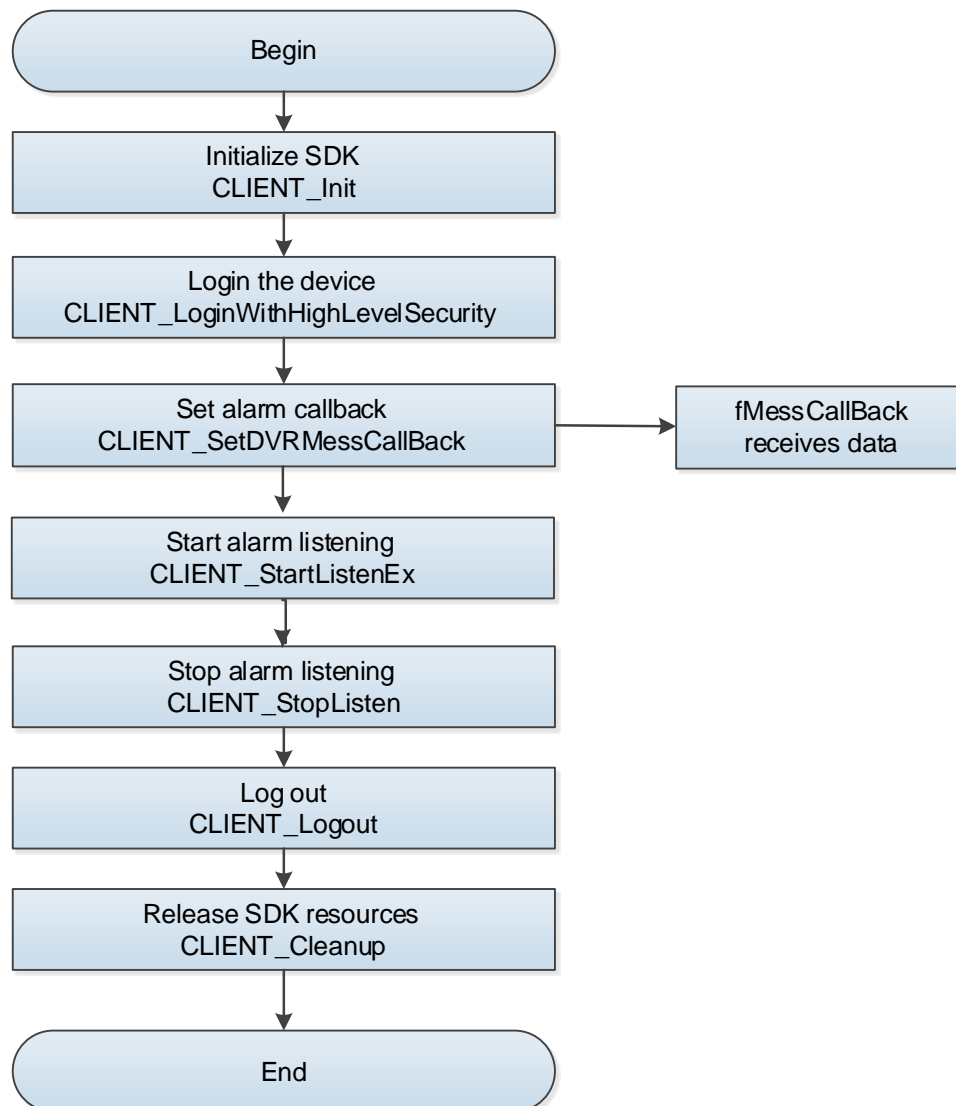
2.7.2 Interface Overview

Table 2-8 Interfaces of alarm listening

Interface	Implication
CLIENT_StartListenEx	Subscribe alarm from device
CLIENT_StopListen	Stop subscribing alarm
CLIENT_SetDVRMessCallBack	Set alarm listening

2.7.3 Process

Figure 2-10 Process of alarm listening



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 Set alarm callback function and the way to call **CLIENT_SetDVRMessCallBack**.
- Step 4 Call **CLIENT_StartListenEx** to start alarm listening.
- Step 5 After alarm listening, the **fAnalyzerDataCallBack** callback gets the alarm events uploaded by devices and then notifies users.
- Step 6 Call **CLIENT_StopListen** to stop alarm listening.
- Step 7 After using the function module, call **CLIENT_Logout** to log out of the device.
- Step 8 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

No uploaded data: Only when you call **CLIENT_SetDVRMessCallBack**, will alarm data be

acquired.

2.7.4 Example Code

```
// Set alarm callback
netSdk.CLIENT_SetDVRMessCallBack(callback, null);
// Start alarm listening
if (listening) {
    return true;
}
listening = netSdk.CLIENT_StartListenEx(loginHandle);
if (!listening) {
    System.err.println("Start Listen Failed!" + ToolKits.getErrorCode());
} else {
    System.out.println("Start Listen Success.");
}
}
//fmessCallback
public class MessCallBack implements NetSDKLib.fMessCallBack {
    private MessCallBack() {}
    private static class CallbackHolder {
        private static final MessCallBack cb = new MessCallBack();
    }

    public static final MessCallBack getInstance() {
        return CallbackHolder.cb;
    }
    @Override
    public boolean invoke(int ICommand, LLong ILoginID, Pointer pStuEvent,
        int dwBufLen, String strDeviceIP, NativeLong nDevicePort,
        Pointer dwUser) {
        switch (ICommand) {
            case NetSDKLib.NET_ALARM_ACCESS_CTL_EVENT : // Access control event
            {
                ALARM_ACCESS_CTL_EVENT_INFO msg = new
ALARM_ACCESS_CTL_EVENT_INFO();
                ToolKits.GetPointerData(pStuEvent, msg);
                System.out.println("【Access control event】 " + msg);
                break;
            }
        }
    }
}
```

```
// Stop alarm listening
if (listening) {
    netSdk.CLIENT_StopListen(loginHandle);
    listening = false;
}
```

2.8 Intelligent Event

2.8.1 Introduction

Intelligent event is the function to analyze real-time stream by smart devices. When the set event occurs, the alarm events will be sent to users, such as traffic violation and parking space. SDK connects to the device and subscribes intelligent event function. When the device gets the intelligent events, they will be sent to SDK.

For the supported intelligent events, see the constants starting with `EVENT_IVS_` in `NetSDKLib.java`, which include events such as regular traffic violation.

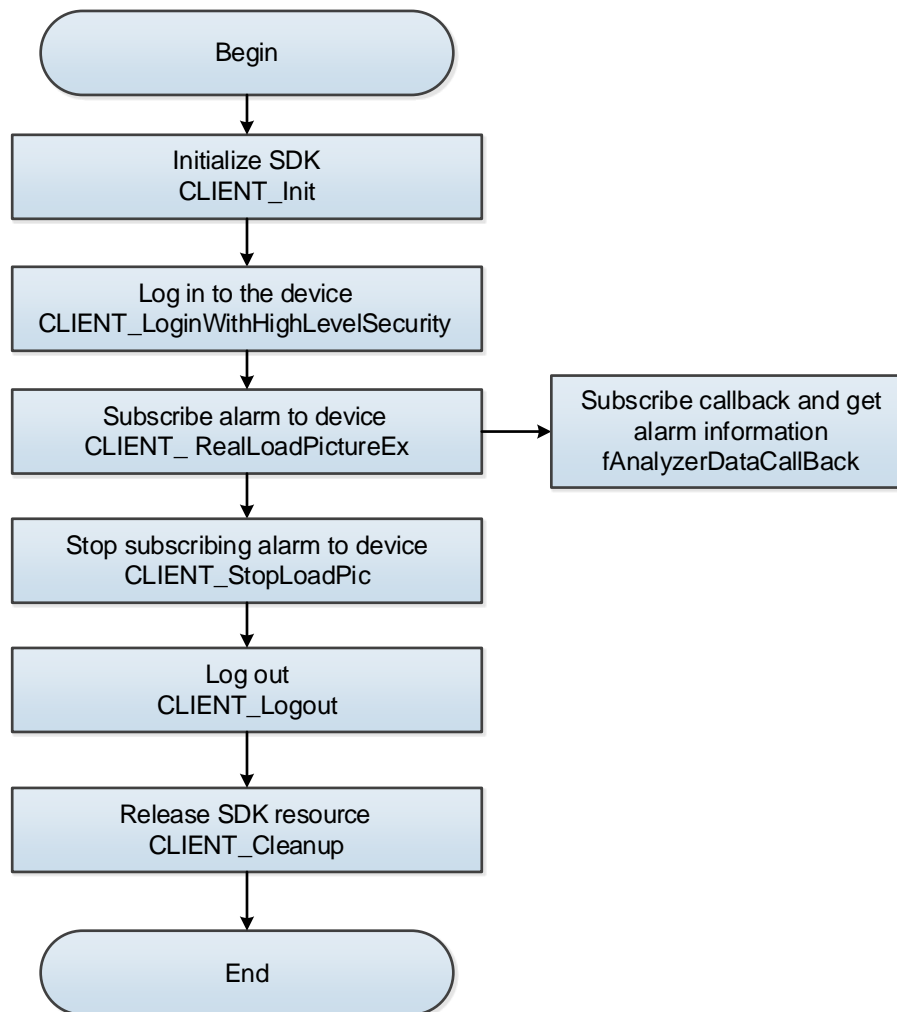
2.8.2 Interface Overview

Table 2-9 Interfaces of intelligent event

Interface	Implication
<code>CLIENT_RealLoadPictureEx</code>	Subscribe alarm events.
<code>CLIENT_StopLoadPic</code>	Stop subscribing intelligent events.
<code>fAnalyzerDataCallBack</code>	Get intelligent event information from callback.

2.8.3 Process

Figure 2-11 Process of intelligent event upload



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 Call **CLIENT_RealLoadPictureEx** to subscribe to smart traffic event to device.
- Step 4 After subscribing, the **fAnalyzerDataCallBack** callback gets the alarm events uploaded by devices and then notifies users.
- Step 5 After uploading, call **CLIENT_StopLoadPic** to stop subscription.
- Step 6 After using the function module, call **CLIENT_Logout** to log out of the device.
- Step 7 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Event type: Subscribe to all intelligent events (EVENT_IVS_ALL) if different intelligent events need to be uploaded. Also support to subscribe to a single intelligent event.
- Image receiving or not: The network environment of some devices is 3G or 4G. When the SDK is connected to the device, set bNeedPicFile parameter in the CLIENT_RealLoadPictureEx to false if images are not needed. Then only receive

information about smart traffic event, without images.

2.8.4 Example Code

```
// Take access control event as an example.
// Skip steps of SDK initialization and login.
// Subscribe handle
public static LLong m_hAttachHandle = new LLong(0);
private AnalyzerDataCB analyzerCallback = new AnalyzerDataCB();
private boolean isAttach = false;
// Listening
private void setOnClickListener() {
    // Subscribe to the upload of intelligent events.
    attachBtn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            m_hAttachHandle = GateModule.realLoadPic(chnComboBox.getSelectedIndex(),
analyzerCallback);
            if(m_hAttachHandle.longValue() != 0) {
                isAttach = true;
                attachBtn.setEnabled(false);
                detachBtn.setEnabled(true);
            } else {
                JOptionPane.showMessageDialog(null, ToolKits.getErrorCodeShow(),
Res.string().getErrorMessage(), JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    // cancel subscription
    detachBtn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            GateModule.stopRealLoadPic(m_hAttachHandle);
            synchronized (this) {
                isAttach = false;
            }
            attachBtn.setEnabled(true);
            detachBtn.setEnabled(false);
        }
    });
}
```

```

        clearPanel();
    }
});
}
// Callback of intelligent event of access control system which comes from fAnalyzerDataCallBack. And
// then the callback implements the logic itself.
private class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack {
    private BufferedImage gateBufferedImage = null;
    @Override
    public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
        Pointer pAlarmInfo, Pointer pBuffer, int dwBufSize,
        Pointer dwUser, int nSequence, Pointer reserved)
    {
        if (lAnalyzerHandle.longValue() == 0 || pAlarmInfo == null) {
            return -1;
        }

        File path = new File("./GateSnapPicture/");
        if (!path.exists()) {
            path.mkdir();
        }

        ///< Access control event
        if(dwAlarmType == NetSDKLib.EVENT_IVS_ACCESS_CTL) {
            DEV_EVENT_ACCESS_CTL_INFO msg = new DEV_EVENT_ACCESS_CTL_INFO();
            ToolKits.GetPointerData(pAlarmInfo, msg);

            // Save image and get image buffer
            String snapPicPath = path + "\\" + System.currentTimeMillis() + "GateSnapPicture.jpg"; //
Image save path
            byte[] buffer = pBuffer.getByteArray(0, dwBufSize);
            ByteArrayInputStream byteArrInputGlobal = new ByteArrayInputStream(buffer);

            try {
                gateBufferedImage = ImageIO.read(byteArrInputGlobal);
                if(gateBufferedImage != null) {
                    ImageIO.write(gateBufferedImage, "jpg", new File(snapPicPath));
                }
            } catch (IOException e2) {
                e2.printStackTrace();
            }
        }
    }
}

```

```

    }

    // Display images and interface of access control information.
    EventQueue eventQueue = Toolkit.getDefaultToolkit().getSystemEventQueue();
    if (eventQueue != null) {
        eventQueue.postEvent( new AccessEvent(target,
                                                gateBufferedImage,
                                                msg));
    }
}

return 0;
}
}

```

2.9 Record Playback

2.9.1 Introduction

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

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

2.9.2 Interface Overview

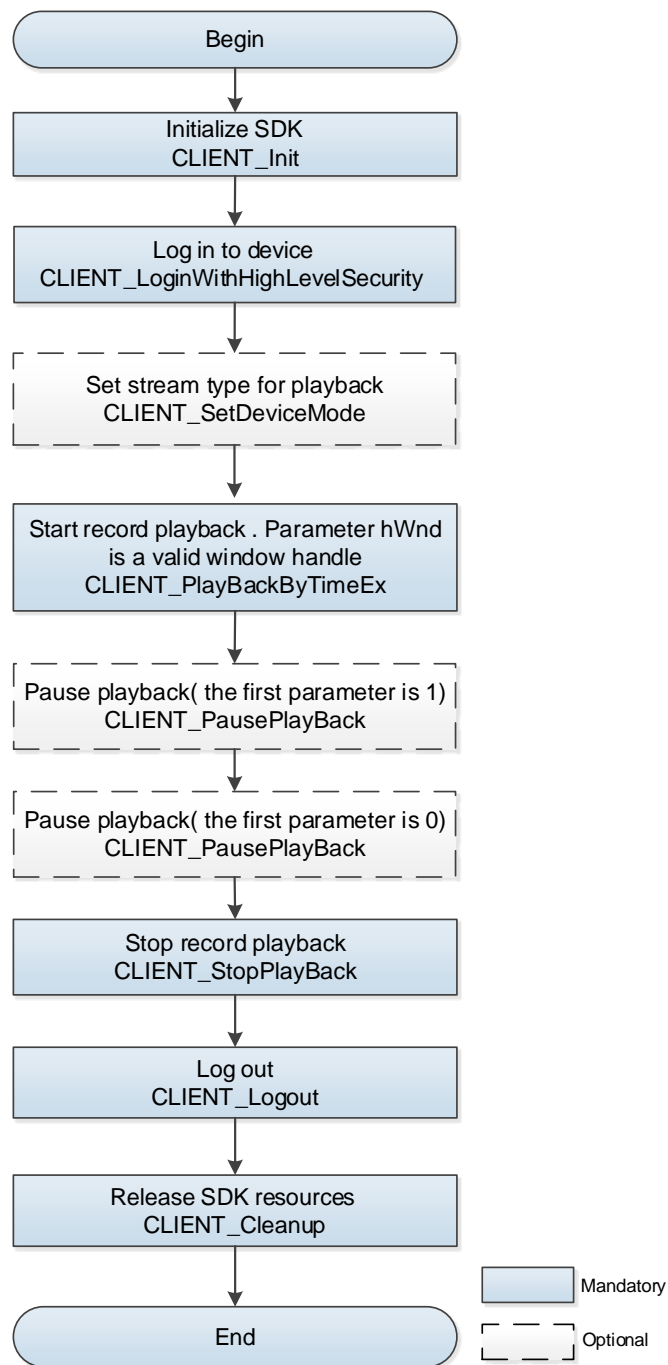
Table 2-10 Interfaces of record playback

Interface	Implication
CLIENT_PlayBackByTimeEx	Playback by time.
CLIENT_SetDeviceMode	Set the work mode such as voice talk, playback, and authority.
CLIENT_StopPlayBack	Stop record playback.
CLIENT_PausePlayBack	Pause or resume playback.

2.9.3 Process

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

Figure 2-12 Process of record playback



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 (Optional) Call **CLIENT_SetDeviceMode** twice and set the stream type parameter emType as DH_RECORD_STREAM_TYPE and the record type parameter emType as DH_RECORD_TYPE.
- Step 4 Call **CLIENT_PlayBackByTimeEx** to start playback. The parameter hWnd is a valid window handle value.
- Step 5 (Optional) Call **CLIENT_PausePlayBack**. The playback will pause when the second parameter is 1.

Step 6 (Optional) Call **CLIENT_PausePlayBack**. The playback will resume when the second parameter is 0.

Step 7 Call **CLIENT_StopPlayBack** to stop playback.

Step 8 After using the function module, call **CLIENT_Logout** to log out of the device.

Step 9 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

2.9.4 Example Code

```
// Start playback
private void StartPlayBack() {
    if (m_hLoginHandle.longValue() == 0) {
        System.err.printf("Please Login First");
        return;
    }

    // Playback and download cannot be performed simultaneously with the same login handle.
    if (m_hDownloadHandle.longValue() != 0) {
        JOptionPane.showMessageDialog(playFrame, "Please stop downloading");
        return;
    }

    updatePlayBackParams(); // Update parameters

    // set stream type of playback
    IntByReference steamType = new IntByReference(m_streamType); // 0-Main and sub stream,
    1-Mian stream, 2-Sub stream
    int emType = NetSDKLib.EM_USEDEV_MODE.NET_RECORD_STREAM_TYPE;

    boolean bret = NetSdk.CLIENT_SetDeviceMode(m_hLoginHandle, emType,
    steamType.getPointer());
    if (!bret) {
        System.err.printf("Set Stream Type Failed, Get last error [0x%x]\n",
        NetSdk.CLIENT_GetLastError());
    }

    // Set video type of recorded playback
    IntByReference emFileType = new IntByReference(m_recordType); // All recorded videos
    NET_RECORD_TYPE
    emType = NetSDKLib.EM_USEDEV_MODE.NET_RECORD_TYPE;
    bret = NetSdk.CLIENT_SetDeviceMode(m_hLoginHandle, emType, emFileType.getPointer());
    if (!bret) {
```

```

        System.err.printf("Set Record Type Failed, Get last error [0x%x]\n",
NetSdk.CLIENT_GetLastError());
    }

    m_hPlayHandle = NetSdk.CLIENT_PlayBackByTimeEx(m_hLoginHandle,
m_channel.intValue(), m_startTime, m_stopTime,
        playWindow.getHWNDOFFrame(), m_PlayBackDownLoadPos, null, m_dataCallBack,
null);

    if (m_hPlayHandle.longValue() == 0) {
        int error = NetSdk.CLIENT_GetLastError();
        System.err.printf("PlayBackByTimeEx Failed, Get last error [0x%x]\n", error);
        switch(error) {
            case LastError.NET_NO_RECORD_FOUND:
                JOptionPane.showMessageDialog(playFrame, "No recorded video");

                break;
            default:
                JOptionPane.showMessageDialog(playFrame, "Failed to start, and error code" +
String.format("0x%x", error));
                break;
        }
    }
    else {
        System.out.println("PlayBackByTimeEx Succeeded");
        m_playFlag = true; // Enable the play flag
        playButton.setText("Stop playback");
        panelPlayBack.repaint();
        panelPlayBack.setVisible(true);
    }
}

// Stop playback
private void StopPlayBack() {
    if (m_hPlayHandle.longValue() == 0) {
        System.err.println("Please make sure the PlayBack Handle is valid");
        return;
    }

    if (!NetSdk.CLIENT_StopPlayBack(m_hPlayHandle)) {
        System.err.println("StopPlayBack Failed");
    }
}

```

```

        return;
    }

    m_hPlayHandle.setValue(0);
    m_playFlag = false;
    m_pauseFlag = true;
    playPos = 0;
    playButton.setText("Start playback");
    pauseButton.setText("Pause");
    panelPlayBack.repaint();
}

/**
 * Pause and play
 * @param pause true - Pause; false - Play
 */
private void PausePlayBack(boolean pause) {
    if (m_hPlayHandle.longValue() == 0) {
        System.err.println("Please make sure the PlayBack Handle is valid");
        return;
    }

    NetSdk.CLIENT_PausePlayBack(m_hPlayHandle, pause ? 1 : 0); // 1 – Pause 0 - Resume
    pauseButton.setText(pause ? "Play":"Pause");
}

// Play at normal speed
private void NormalPlayBack() {
    if (m_hPlayHandle.longValue() == 0) {
        System.err.println("Please make sure the PlayBack Handle is valid");
        return;
    }

    NetSdk.CLIENT_NormalPlayBack(m_hPlayHandle);
}

// Fast play
private void FastPlayBack() {
    if (m_hPlayHandle.longValue() == 0) {
        System.err.println("Please make sure the PlayBack Handle is valid");
    }
}

```

```

        return;
    }

    NetSdk.CLIENT_FastPlayBack(m_hPlayHandle);
}

// Slow play
private void SlowPlayBack() {
    if (m_hPlayHandle.longValue() == 0) {
        System.err.println("Please make sure the PlayBack Handle is valid");
        return;
    }

    NetSdk.CLIENT_SlowPlayBack(m_hPlayHandle);
}

```

2.10 Record Download

2.10.1 Introduction

Video surveillance system widely applies to safe city, airport, metro, bank and factory. When any event occurs, you need to download the video records and report to the leaders, public security bureau, or mass media. Therefore, record download is an important function.

The record download function helps you obtain the records saved on the device through SDK and save into the local. It allows you to download from the selected channels and export to the local disk or external USB flash drive.

This function is available for some select models.

2.10.2 Interface Overview

Table 2-11 Interfaces of record download

Interface	Implication
CLIENT_QueryRecordFile	Query for all the record files within a period.
CLIENT_DownloadByTimeEx	Download the record by time.
CLIENT_StopDownload	Stop the record download.

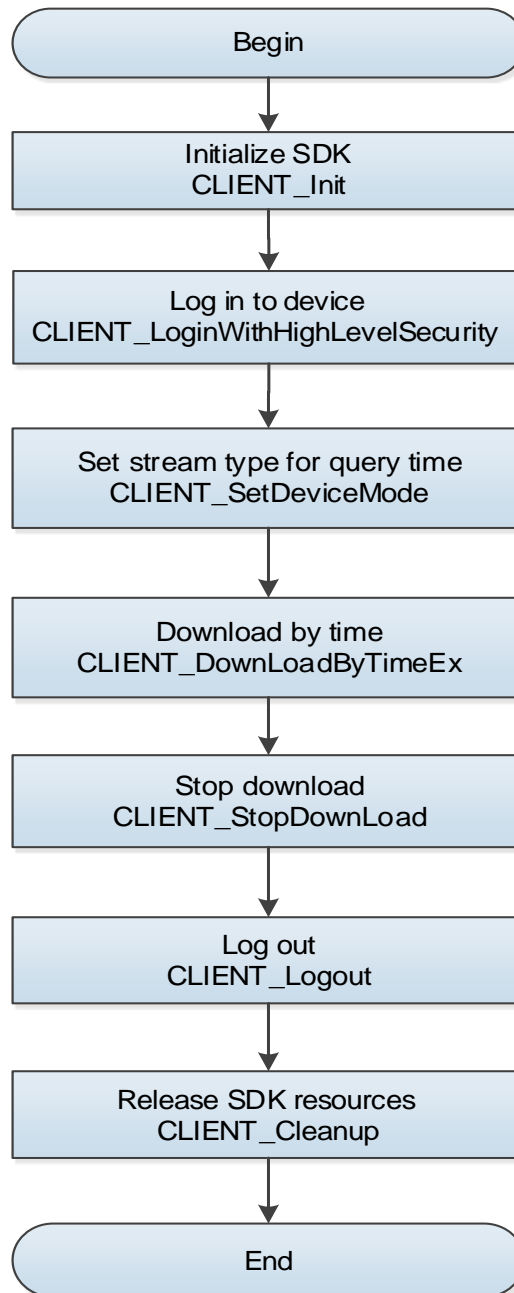
2.10.3 Process

You can import the start time and end time of download. SDK can download the specified

record file and save to the required place.

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

Figure 2-13 Process of record download



Process Description

- Step 1 Call **CLIENT_Init** to initialize SDK.
- Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
- Step 3 Call **CLIENT_SetDeviceMode** to set the stream type for query time, and the parameter emType should be DH_RECORD_STREAM_TYPE.
- Step 4 Call **CLIENT_DownloadByTimeEx** to start downloading by time. Either sSavedFileName or fDownloadDataCallBack is valid. You can decide whether to use cbDownloadPos; if not, set it as NULL.
- Step 5 Call **CLIENT_StopDownload** to stop download. You can close the download process

after it is completed or it is just partially completed.

Step 6 After using the function module, call **CLIENT_Logout** to log out of the device.

Step 7 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

2.10.4 Example Code

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

import com.sun.jna.ptr.IntByReference;

/**
 * Implement record download
 * Mainly are record query, record download and setting of stream type.
 */
public class DownloadRecordModule {
    // Download handle
    public static LLong m_hDownloadHandle = new LLong(0);

    // Query for record file
    public static boolean queryRecordFile(int nChannelId,
                                           NetSDKLib.NET_TIME stTimeStart,
                                           NetSDKLib.NET_TIME stTimeEnd,
                                           NetSDKLib.NET_RECORDFILE_INFO[] stFileInfo,
                                           IntByReference nFindCount) {
        // RecordFileType 0: All recorded videos, 1: External alarm, 2: Dynamic monitoring alarm, 3: All
        // alarms, 4: Card ID query, 5: combined condition query
        // 6: Record location and deviation length, 8: Image query by card ID (currently only supported
        // by select models of HB-U and NVS), 9: Image query (currently only supported by select models of HB-U
        // and NVS)
        // 10: Query by field, 15: Return network data structure (Jinqiao Internet bar), 16: Query for all
        // recoding files of transparent string data
        int nRecordFileType = 0;
        boolean bRet = LoginModule.netsdk.CLIENT_QueryRecordFile(LoginModule.m_hLoginHandle,
                                                                    nChannelId, nRecordFileType, stTimeStart, stTimeEnd, null, stFileInfo, stFileInfo.length *
                                                                    stFileInfo[0].size(), nFindCount, 5000, false);

        if(bRet) {
            System.out.println("QueryRecordFile Succeed! \n" + "The number of queried videos: " +
                                nFindCount.getValue());
        }
    }
}
```

```

    } else {
        System.err.println("QueryRecordFile Failed!" + ToolKits.getErrorCodePrint());
        return false;
    }
    return true;
}

/**
 * Set stream type of playback
 * @param m_streamType
 */
public static void setStreamType(int m_streamType) {

    IntByReference steamType = new IntByReference(m_streamType);// 0-Main and sub stream,
1-Mian stream, 2-Sub stream
    int emType = NetSDKLib.EM_USEDEV_MODE.NET_RECORD_STREAM_TYPE;

    boolean bret = LoginModule.netsdk.CLIENT_SetDeviceMode(LoginModule.m_hLoginHandle,
emType, steamType.getPointer());
    if (!bret) {
        System.err.println("Set Stream Type Failed, Get last error." +
ToolKits.getErrorCodePrint());
    } else {
        System.out.println("Set Stream Type Succeed!");
    }
}

// Download record
public static LLong downloadRecordFile(int nChannelId,
                                        int nRecordFileType,
                                        NetSDKLib.NET_TIME stTimeStart,
                                        NetSDKLib.NET_TIME stTimeEnd,
                                        String SavedFileName,
                                        NetSDKLib.fTimeDownLoadPosCallBack
cbTimeDownLoadPos) {

    m_hDownLoadHandle =
LoginModule.netsdk.CLIENT_DownloadByTimeEx(LoginModule.m_hLoginHandle, nChannelId,
nRecordFileType, stTimeStart, stTimeEnd, SavedFileName, cbTimeDownLoadPos, null, null, null, null);
    if(m_hDownLoadHandle.longValue() != 0) {
        System.out.println("Downloading RecordFile!");
    } else {

```

```
        System.err.println("Download RecordFile Failed!" + ToolKits.getErrorCodePrint());
    }
    return m_hDownloadHandle;
}

public static void stopDownloadRecordFile(LLong m_hDownloadHandle) {
    if (m_hDownloadHandle.longValue() == 0) {
        return;
    }
    LoginModule.netsdk.CLIENT_StopDownload(m_hDownloadHandle);
}
}
```

3 Interface Definition

3.1 SDK Initialization

3.1.1 SDK CLIENT_Init

Table 3-1 Initialize SDK

Item	Description	
Name	Initialize SDK.	
Function	public boolean CLIENT_Init(Callback cbDisconnect, Pointer 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	public 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	public void CLIENT_SetAutoReconnect(Callback cbAutoConnect, Pointer dwUser);	
Parameter	[in]cbAutoConnect	Reconnection callback.
	[in]dwUser	User parameter of disconnection callback.
Return value	None.	
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	public void CLIENT_SetNetworkParam(NET_PARAM pNetParam);	
Parameter	[in]pNetParam	Parameters such as network delay, reconnection times, and buffer size.
Return value	None.	
Note	Adjust the parameters according to the actual network environment.	

3.2 Device Login

3.2.1 CLIENT_LoginWithHighLevelSecurity

Table 3-5 Log in

Item	Description	
Name	Log in to the device.	
Function	public LLong CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY pstOutParam);	
Parameter	[in] pstInParam	Input parameter.
	[out] pstOutParam	Output parameter.
Return value	<ul style="list-style-type: none">• Success: not 0.• Failure: 0.	
Note	This method is encapsulated in the NetSDKLib interface and is usually called by: m_hLoginHandle = netsdk.CLIENT_CLIENT_LoginWithHighLevelSecurity(pstInParam, pstOutParam);	

3.2.2 CLIENT_Logout

Table 3-6 Log out

Item	Description	
Name	User logout the device.	
Function	public boolean CLIENT_Logout(LLong lLoginID);	
Parameter	[in]lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	

Item	Description
Note	This method is encapsulated in the NetSDKLib interface and is usually called by: netsdk.CLIENT_Logout(m_hLoginHandle);

3.3 Real-time Monitoring

3.3.1 CLIENT_RealPlayEx

Table 3-7 Start monitoring

Item	Description
Name	Open the real-time monitoring.
Function	public LLong CLIENT_RealPlayEx(LLong ILoginID, int nChannelID, Pointer hWnd, int rType);
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 Live type.
Return value	<ul style="list-style-type: none"> Success: not 0 Failure: 0
Note	Windows system: <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 3-8 Live type and meaning

Live type	Meaning
DH_RType_Realplay	Real-time live
DH_RType_Multiplay	Multi-picture live
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 live—1 picture
DH_RType_Multiplay_4	Multi-picture live—4 pictures
DH_RType_Multiplay_8	Multi-picture live—8 pictures
DH_RType_Multiplay_9	Multi-picture live—9 pictures
DH_RType_Multiplay_16	Multi-picture live—16 pictures
DH_RType_Multiplay_6	Multi-picture live—6 pictures
DH_RType_Multiplay_12	Multi-picture live—12 pictures
DH_RType_Multiplay_25	Multi-picture live—25 pictures

Live type	Meaning
DH_RType_Multiplay_36	Multi-picture live—36 pictures

3.3.2 CLIENT_StopRealPlayEx

Table 3-9 Stop monitoring

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

3.3.3 CLIENT_SaveRealData

Table 3-10 Save monitoring data

Item	Description
Name	Save the real-time monitoring data as file.
Function	public boolean CLIENT_SaveRealData(LLong IRealHandle, String pchFileName);
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.

3.3.4 CLIENT_StopSaveRealData

Table 3-11 Stop saving monitoring data

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

3.3.5 CLIENT_SetRealDataCallbackEx

Table 3-12 Set the callback of real-time monitoring data

Item	Description
Name	Set the callback of real-time monitoring data.

Item	Description	
Function	public boolean CLIENT_SetRealDataCallBackEx(LLong IRealHandle, StdCallCallback cbRealData, Pointer dwUser, int dwFlag);	
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.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 	
Note	None.	

Table 3-13 Type and meaning of dwFlag

dwFlag	Meaning
0x00000001	Initial data of device.
0x00000004	Data converted to YUV format.

3.4 Video Snapshot

3.4.1 CLIENT_SnapPictureToFile

Table 3-14 Synchronous snapshot

Item	Description	
Name	Synchronous snapshot.	
Function	public boolean CLIENT_SnapPictureToFile(LLong ILoginID, NET_IN_SNAP_PIC_TO_FILE_PARAM pInParam, NET_OUT_SNAP_PIC_TO_FILE_PARAM pOutParam, int nWaitTime)	
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	<ul style="list-style-type: none"> • Synchronous interface. The device captures snapshot and sends to the user through internet. • This function is available for some select models. 	

3.4.2 CLIENT_CapturePictureEx

Table 3-15 Asynchronous snapshot

Item	Description	
Name	Asynchronous snapshot.	
Function	public boolean CLIENT_SnapPictureEx(LLong ILoginID, SNAP_PARAMS stParam, IntByReference reserved);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] stParam	Snapshot parameters structure.
	[in] reserved	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.	

3.4.3 CLIENT_CapturePictureEx

Table 3-16 Local snapshot

Item	Description	
Name	Local snapshot.	
Function	public boolean CLIENT_CapturePictureEx(LLong hPlayHandle, String pchPicFileName, int eFormat);	
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.	

3.4.4 Setting Asynchronous Snapshot Callback

Table 3-17 Set asynchronous snapshot callback

Item	Description	
Name	Callback of asynchronous snapshot.	
Function	public void CLIENT_SetSnapRevCallBack(Callback OnSnapRevMessage, Pointer dwUser);	
Parameter	[out] OnSnapRevMessage	Function prototype of snapshot callback.
	[out] dwUser	User parameters of callback.
Return value	None.	

Item	Description
Note	None.

3.5 PTZ Control

3.5.1 CLIENT_DHPTZControlEx

Table 3-18 Control PTZ

Item	Description
Name	PTZ control.
Function	public boolean CLIENT_DHPTZControlEx(LLong ILoginID, int nChannelID, int dwPTZCommand, int IParam1, int IParam2, int IParam3, int dwStop);
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.
	[in] IParam2 Parameter 2.
	[in] IParam3 Parameter 3.
	[in] dwStop Stop mark, which is valid for operations of eight directions. When performing other operations, enter FALSE for this parameter.
Return value	<ul style="list-style-type: none"> Success: TRUE. Failure: FALSE.
Note	For the relationship between dwPTZCommand and Param1, Param2 and Param3, see Table 3-19.

Table 3-19 Relationship between command and parameters

dwPTZCommand macro definition	Function	param1	param2	param3
DH_PTZ_UP_CONTR OL	Up	None	Vertical speed (1–8)	None
DH_PTZ_DOWN_CON TROL	Down	None	Vertical speed (1–8)	None
DH_PTZ_LEFT_CONT ROL	Left	None	Horizontal speed (1–8)	None
DH_PTZ_RIGHT_CON TROL	Right	None	Horizontal speed (1–8)	None
DH_PTZ_ZOOM_ADD _CONTROL	Zoom+	None	Multi-speed	None

dwPTZCommand macro definition	Function	param1	param2	param3
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
DH_PTZ_POINT_DELETE_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_DELETEFROMLOOP	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_SETRIGHTBORDER	Set right border	None	None	None

dwPTZCommand macro definition	Function	param1	param2	param3
DH_EXTPTZ_STARTLINESCAN	Start line scan	None	None	None
DH_EXTPTZ_CLOSELINESCAN	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_RUNNINGMODE	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
DH_EXTPTZ_FASTGOTO	Fast positioning	Horizontal coordinate (0–8192)	Vertical coordinate (0–8192)	Zoom (4)
DH_EXTPTZ_AUXILIARYOPEN	Open auxiliary switch	Auxiliary point	None	None
DH_EXTPTZ_AUXILIARYCLOSE	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

dwPTZCommand macro definition	Function	param1	param2	param3
DH_EXTPTZ_ALARM HANDLE	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_MATRIX SWITCH	Matrix switch	Monitor device number (video output number)	Video input number	Matrix number
DH_EXTPTZ_LIGHTC ONTROL	Light controller	Refer to DH_PTZ_LA MP_CONTRO L	None	None
DH_EXTPTZ_EXACT GOTO	3D positioning	Horizontal angle (0– 3600)	Vertical coordinate (0– 900)	Zoom (1– 128)
DH_EXTPTZ_RESE TZERO	Reset to zero	None	None	None
DH_EXTPTZ_UP_T ELE	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_LEFT UP _TELE	Leftup +TELE	Speed (1–8)	None	None
DH_EXTPTZ_LEFT DOWN _TELE	Leftdown +TELE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHT UP _TELE	Rightup+TELE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHT DOWN _TELE	Rightdown +TELE	Speed (1–8)	None	None
DH_EXTPTZ_UP_W IDE	Up +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_DOWN _WIDE	Down+WIDE	Speed (1–8)	None	None
DH_EXTPTZ_LEFT _WIDE	Left +WIDE	Speed (1–8)	None	None
DH_EXTPTZ_RIGHT _WIDE	Right+WIDE	Speed (1–8)	None	None

dwPTZCommand macro definition	Function	param1	param2	param3
DH_EXTPTZ_LEFTUP_WIDE	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_WIDE	Rightdown +WIDE	Speed (1–8)	None	None

3.6 Voice Talk

3.6.1 CLIENT_StartTalkEx

Table 3-20 Start voice talk

Item	Description	
Name	Start voice talk.	
Function	public LLong CLIENT_StartTalkEx(LLong ILoginID, Callback pfcB, Pointer dwUser);	
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.	

3.6.2 CLIENT_StopTalkEx

Table 3-21 Stop voice talk

Item	Description	
Name	Stop voice talk.	
Function	public boolean 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.	

3.6.3 CLIENT_TalkSendData

Table 3-22 Send voice talk data

Item	Description	
Name	Send audio data to device.	
Function	public LLong CLIENT_TalkSendData(LLong ITalkHandle, Pointer pSendBuf, int dwBufSize);	
Parameter	[in] ITalkHandle	Return value of CLIENT_StartTalkEx.
	[in] pSendBuf	Pointer of audio data block that needs to be sent.
	[in] dwBufSize	Length of audio data black that needs to be sent. The unit is byte.
Return value	<ul style="list-style-type: none">• Success: Length of audio data block.• Failure: -1.	
Note	None.	

3.6.4 CLIENT_AudioDecEx

Table 3-23 Decode audio data

Item	Description	
Name	Decode audio data.	
Function	public boolean CLIENT_AudioDecEx(LLong ITalkHandle, Pointer pAudioDataBuf, int dwBufSize);	
Parameter	[in] ITalkHandle	Return value of CLIENT_StartTalkEx.
	[in] pAudioDataBuf	Pointer of audio data block that needs decoding.
	[in] dwBufSize	Length of audio data black that needs decoding. The unit is byte.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	None.	

3.7 Alarm Listening

3.7.1 CLIENT_StartListenEx

Table 3-24 Start alarm listening

Item	Description	
Name	Start alarm listening.	
Function	public boolean CLIENT_StartListenEx(LLong ILoginID);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.

Item	Description
Return value	<ul style="list-style-type: none"> • Success: Not 0. • Failure: 0.
Note	None.

3.7.2 CLIENT_StopListen

Table 3-25 Stop alarm listening

Item	Description		
Name	Stop alarm listening.		
Function	public boolean CLIENT_StopListen(LLong ILoginID);		
Parameter	<table border="1"> <tr> <td>[in] ILoginID</td><td>Return value of CLIENT_LoginWithHighLevelSecurity.</td></tr> </table>	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.		
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE. 		
Note	None.		

3.7.3 CLIENT_SetDVRMessCallBack

Table 3-26 Set alarm listening

Item	Description	
Name	Set alarm listening.	
Function	public void CLIENT_SetDVRMessCallBack(Callback cbMessage , Pointer dwUser);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] dwUser	Returned user information.
Return value	<ul style="list-style-type: none">● Success: TRUE.● Failure: FALSE.	
Note	None.	

3.8 Intelligent Event

3.8.1 CLIENT_RealLoadPictureEx

Table 3-27 Start subscribing intelligent event

Item	Description
Name	Start subscribing intelligent event.

Item	Description	
Function	public LLong CLIENT_RealLoadPictureEx(LLong ILoginID, int nChannelID, int dwAlarmType, int bNeedPicFile, StdCallCallback cbAnalyzerData, Pointer dwUser, Pointer Reserved);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nChannelID	Device channel number, starting from 0.
	[in] dwAlarmType	Type of subscribed alarm event.
	[in] bNeedPicFile	Subscribe to image file or not?
	[in] cbAnalyzerData	Callback of intelligent event.
	[in] dwUser	Type of customized data.
	[in] Reserved	Reserved handle.
Return value	<ul style="list-style-type: none"> Success: Subscription handle of LLONG type. Failure: FALSE. 	
Note	Get error code by CLIENT_GetLastError when the interface fails to return.	

Table 3-28 Intelligent event

dwAlarmType macro definition	Value of macro definition	Meaning	Corresponding structure of pAlarmInfo
EVENT_IVS_ALL	0x00000001	All event	None
EVENT_IVS_CROSSFENCE DETECTION	0x0000011F	Cross fence	DEV_EVENT_CROSSFENC EDETECTION_INFO
EVENT_IVS_CROSSLINED ETECTION	0x00000002	Tripwire	DEV_EVENT_CROSSLINE_ INFO
EVENT_IVS_CROSSREGIO NDETECTION	0x00000003	Intrusion	DEV_EVENT_CROSSREGI ON_INFO
EVENT_IVS_LEFTDETECTI ON	0x00000005	Abandoned Object	DEV_EVENT_LEFT_INFO
EVENT_IVS_PRESERVATIO N	0x00000008	Object Protection	DEV_EVENT_PRESERVATI ON_INFO
EVENT_IVS_TAKENAWAYD ETECTION	0x00000115	Missing object	DEV_EVENT_TAKENAWAY DETECTION_INFO
EVENT_IVS_WANDERDET ECTION	0x00000007	Loitering	DEV_EVENT_WANDER_IN FO
EVENT_IVS_VIDEOABNOR MALDETECTION	0x00000013	Video error	DEV_EVENT_VIDEOABNO RMALDETECTION_INFO
EVENT_IVS_AUDIO_ABNO RMALDETECTION	0x00000126	Audio error	DEV_EVENT_IVS_AUDIO_ ABNORMALDETECTION_I NFO
EVENT_IVS_CLIMBDETEC TION	0x00000128	Sticker detection	DEV_EVENT_IVS_CLIMB_I NFO
EVENT_IVS_FIGHTDETECT ION	0x0000000E	Fighting detection	DEV_EVENT_FLOWSTAT_I NFO
EVENT_IVS_LEAVEDETEC TION	0x00000129	AWOL detection	DEV_EVENT_IVS_LEAVE_I NFO

dwAlarmType macro definition	Value of macro definition	Meaning	Corresponding structure of pAlarmInfo
EVENT_IVS_PRISONERRISEDETECTION	0x0000011E	Stand detection	DEV_EVENT_PRISONERRISEDETECTION_INFO
EVENT_IVS_PASTEDETECTION	0x00000004	Sticker detection	DEV_EVENT_PASTE_INFO

3.8.2 CLIENT_StopLoadPic

Table 3-29 Stop subscribing intelligent event

Item	Description
Name	Stop subscribing intelligent event.
Function	public boolean CLIENT_StopLoadPic(LLong IAnalyzerHandle);
Parameter	[in] IAnalyzerHandle Subscription handle of intelligent event.
Return value	BOOL type: <ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE.
Note	Get error code by CLIENT_GetLastError when the interface fails to return.

3.9 Record Playback

3.9.1 CLIENT_PlayBackByTimeEx

Table 3-30 Playback by time

Item	Description
Name	Playback by time.
Function	public LLong CLIENT_PlayBackByTimeEx(LLong ILoginID, int nChannelID, NET_TIME lpStartTime, NET_TIME lpStopTime, Pointer hWnd, Callback cbDownLoadPos, Pointer dwPosUser, Callback fDownLoadDataCallBack, Pointer dwDataUser);
Parameter	[in] ILoginID Login handle.
	[in] nChannelID Device channel number, starting from 0.
	[in] lpStartTime Start time.
	[in] lpStopTime Stop time.
	[in] hWnd Window handle (valid only in Windows system).
	[in] cbDownLoadPos Callback of fDownLoadPosCallBack.
	[out] dwPosUser None.
	[out] fDownLoadDataCallBack Callback of fDataCallBack.
	[in] dwDataUser None.
Return value	<ul style="list-style-type: none"> • Success: Network playback ID. • Failure: 0.

Item	Description
Note	None.

3.9.2 CLIENT_SetDeviceMode

Table 3-31 Set the work mode

Item	Description
Name	Set the work mode.
Function	public boolean CLIENT_SetDeviceMode(LLong ILoginID, int emType, Pointer pValue);
Parameter	[in] ILoginID Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emType Work mode enumeration.
	[in] pValue The corresponding structure of work mode.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE.
Note	None.

The following table shows information about work mode enumeration and structure.

Table 3-32 Work mode enumeration and structure

emType enumeration	Meaning	Structure
DH_RECORD_STREAM_TYPE	Set the stream type of recorded videos to be queried or played back by time. <ul style="list-style-type: none"> • 0: Main and sub stream • 1: Main stream • 2: Sub stream 	None
DH_RECORD_TYPE	Set the record file type to play back and download by time.	NET_RECORD_TYPE

3.9.3 CLIENT_StopPlayBack

Table 3-33 Stop record playback

Item	Description
Name	Stop video playback.
Function	public boolean CLIENT_StopPlayBack(LLong IPlayHandle);
Parameter	[in] IPlayHandle Return value of playback interface.
Return value	<ul style="list-style-type: none"> • Success: TRUE. • Failure: FALSE.
Note	None.

3.9.4 CLIENT_PausePlayBack

Table 3-34 Pause or resume record playback

Item	Description	
Name	Pause or resume playback.	
Function	public boolean CLIENT_PausePlayBack(LLong IPlayHandle, int bPause);	
Parameter	[in] IPlayHandle	Return value of playback interface.
	[out] bPause	Parameters for network playback stops and resumes: 1: Pause 0: Resume
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	
Note	Pause or resume the ongoing playback.	

3.10 Record Download

3.10.1 CLIENT_QueryRecordFile

Table 3-35 Query for all record files within a period

Item	Description	
Name	Query for all record files within a period.	
Function	public boolean CLIENT_QueryRecordFile(LLong ILoginID, int nChannelId, int nRecordFileType, NET_TIME tmStart, NET_TIME tmEnd, String pchCardid, NET_RECORDFILE_INFO[] stFileInfo, int maxlen, IntByReference filecount, int waittime, boolean bTime);	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nChannelId	Device channel number, starting from 0.
	[in] nRecordFileType	Record file type.
	[in] tmStart	Record start time.
	[in] tmEnd	Record end time.
	[in] pchCardid	Card ID.
	[out] nriFileInfo	The returned record file is a LPNET_RECORDFILE_INFO structured data.
	[in] maxlen	The maximum length of nriFileInfo buffer, whose unit is byte and recommended to be between "(100~200) *sizeof(NET_RECORDFILE_INFO)".
	[out] filecount	Check the number of returned files only in the cache.
	[in] waittime	Waiting time.
	[in] bTime	Currently invalid.
Return value	<ul style="list-style-type: none">• Success: TRUE.• Failure: FALSE.	

Item	Description
Note	Before playback, call this interface to query for the records. When the queried records within the defined time are larger than the cache size, it will only return the records that can be stored by cache. Continue with the query if needed.

The following table shows information about record file and card ID.

Table 3-36 Record file and card ID

Value	Record file type	Card ID
0	All recorded videos	NULL
1	External alarm	NULL
2	Alarm by motion detection	NULL
3	All the alarms	NULL
4	Card ID query	Card ID
5	Combined condition query	Card ID && Transaction type && Transaction amount (If you want to skip a field, set as blank)
6	Record location and deviation length	NULL
8	Image query by card ID (Only supported by select models of HB-U and NVS)	Card ID
9	Image query (Only supported by select models of HB-U and NVS)	NULL
10	Query by field	FELD1&&FELD2&&FELD3&& (If you want to skip a field, set as blank)

3.10.2 CLIENT_DownloadByTimeEx

Table 3-37 Download record by file

Item	Description	
Name	Download record by time.	
Function	<pre>public LLong CLIENT_DownloadByTimeEx(LLong ILoginID, int nChannelId, int nRecordFileType, NET_TIME tmStart, NET_TIME tmEnd, String sSavedFileName, StdCallCallback cbTimeDownLoadPos, Pointer dwUserData, StdCallCallback fDownLoadDataCallBack, Pointer dwDataUser, Pointer pReserved);</pre>	
Parameter	[in] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nChannelId	The device channel number starting from 0.

Item	Description	
	[in] nRecordFileType	Query type of file. 0: All recorded videos. 1: External alarm. 2: Records of motion detection. 3: All alarms. 4: Recorded video query by card ID. 5: Combined condition query. 8: Image query by card ID. 9: Image query. 10: Query by field.
	[in] tmStart	Start time of download.
	[in] tmEnd	End time of download.
	[in] sSavedFileName	The record file name and full save path.
	[in] cbTimeDownloadPos	Download progress callback.
	[in] dwUserData	Download progress callback customized data.
	[in] fDownloadDataCallBack	Data callback.
	[in] dwUserData	Download data callback customized data.
	[in] pReserved	Parameter reserved and the default is NULL.
Return value	<ul style="list-style-type: none"> Success: Download ID. Failure: 0. 	
Note	<ul style="list-style-type: none"> For callback declaration of fDownloadPosCallBack and fDataCallBack, see "Chapter 错误!未找到引用源。 错误!未找到引用源。." sSavedFileName is not blank, and the record data is input into the file corresponding with the path. fDownloadDataCallBack is not blank, and the record data is returned through callback. 	

3.10.3 CLIENT_StopDownload

Table 3-38 Stop record download

Item	Description	
Name	Stop record download.	
Function	public boolean CLIENT_StopDownload(LLong IFileHandle);	
Parameter	[in] IFileHandle	Return value of CLIENT_DownloadByTimeEx.
Return value	<ul style="list-style-type: none"> Success: ID of download. Failure: 0. 	
Note	Stop downloading after it is completed or partially completed according to particular situation.	

4 Callback Definition

4.1 fDisconnect

Table 4-1 Disconnection callback

Item	Description	
Name	Disconnection callback.	
Function	<pre>public interface fDisconnect extends StdCallCallback { public void invoke(LLong ILoginID, String pchDVRIP, int nDVRPort, Pointer dwUser); }</pre>	
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.	

4.2 fHaveReConnect

Table 4-2 Reconnection callback

Item	Description	
Name	Reconnection callback.	
Function	<pre>public interface fHaveReConnect extends StdCallCallback { public void invoke(LLong ILoginID, String pchDVRIP, int nDVRPort, Pointer dwUser); }</pre>	
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.	

4.3 fRealDataCallbackEx

Table 4-3 Callback of real-time monitoring data

Item	Description	
Name	Callback of real-time monitoring data.	
Function	<pre>public interface fRealDataCallBackEx extends StdCallCallback { public void invoke(LLong IRealHandle, int dwDataType, Pointer pBuffer, int dwBufSize, int param, Pointer dwUser); }</pre>	
Parameter	[out] IRealHandle	Return value of CLIENT_RealPlayEx.
	[out] dwDataType	Data type: <ul style="list-style-type: none"> 0: Initial data. 2: YUV data.
	[out] pBuffer	Address of monitoring data block.
	[out] dwBufSize	Length of the monitoring data block. The unit is byte.
	[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 tagCBYUVDataParam structure when dwDataType is 2.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.4 pfAudioDataCallback

Table 4-4 Audio data callback

Item	Description	
Name	Audio data callback of voice talk.	
Function	<pre>public interface pfAudioDataCallBack extends StdCallCallback { public void invoke(LLong ITalkHandle, Pointer pDataBuf, int dwBufSize, byte byAudioFlag, Pointer 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.

Item	Description	
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.5 fAnalyzerDataCallBack

Table 4-5 Callback of intelligent event

Item	Description	
Name	Callback of intelligent event.	
Function	<pre>public interface fAnalyzerDataCallBack extends StdCallCallback { public int invoke(LLong IAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo, Pointer pBuffer, int dwBufSize, Pointer dwUser, int nSequence, Pointer reserved); }</pre>	
Parameter	[out] IAnalyzerHandle	Return value of CLIENT_RealLoadPictureEx.
	[out] dwAlarmType	Intelligent event type.
	[out] pAlarmInfo	Event information buffer.
	[out] pBuffer	Image buffer.
	[out] dwBufSize	Image buffer size.
	[out] dwUser	User data.
	[out] nSequence	Information about the uploaded picture. 0 means the first occurrence, 2 means the last occurrence or only one occurrence, and 1 means there are still appearances after this time.
	[out] reserved	Reserved.
Return value	None.	
Note	None.	

4.6 fTimeDownloadPosCallBack

Table 4-6 Callback of download by time

Item	Description	
Name	Callback of download by time.	
Function	<pre>public interface fTimeDownloadPosCallBack extends StdCallCallback { public void invoke(LLong IPlayHandle, int dwTotalSize, int dwDownloadSize, int index, NET_RECORDFILE_INFO.ByValue recordfileinfo, Pointer dwUser); }</pre>	

Item	Description	
Parameter	[out] IPlayHandle	Return value of CLIENT_DownloadByTimeEx.
	[out] dwTotalSize	Total size of playback. The unit is KB.
	[out] dwDownloadSize	The size that has been played. The unit is KB. <ul style="list-style-type: none"> • -1: Current download finished. • -2: Write file failed.
	[out] index	Index.
	[out] recordfileinfo	Record file information.
	[out] dwUser	User data.
Return value	None.	

4.7 fMessCallBack

Table 4-7 Alarm subscription callback

Item	Description	
Name	Callback of real-time monitoring data.	
Function	<pre> public interface fMessCallBack extends StdCallCallback{ public boolean invoke(int ICommand , LLong ILoginID , Pointer pStuEvent , int dwBufLen , String strDeviceIP , NativeLong nDevicePort , Pointer dwUser); } </pre>	
Parameter	[out] ICommand	Concrete alarm event.
	[out] ILoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] pStuEvent	Returned data pointer.
	[out] dwBufLen	Length of the returned pointer.
	[out] strDeviceIP	Returned IP.
	[out] nDevicePort	Returned port number.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.8 Asynchronous Snapshot

Table 4-8 Callback of asynchronous snapshot.

Item	Description	
Name	Callback of asynchronous snapshot.	
Function	public interface fSnapRev extends Callback{ public void invoke(LLong ILoginID , Pointer pBuf, int RevLen, int EncodeType, int CmdSerial, Pointer dwUser); }	
Parameter	[out] ILoginID	Return value of CLIENT_LoginWithHighLevelSerity.
	[out] pBuf	Address of asynchronous snapshot.
	[out] RevLen	Length of asynchronous snapshot.
	[out] EncodeType	Encoding type.
	[out] CmdSerial	Operation serial number.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

Appendix 1 Cybersecurity Recommendations

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

Mandatory actions to be taken for basic equipment network security:

1. Use Strong Passwords

Please refer to the following suggestions to set passwords:

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

2. Update Firmware and Client Software in Time

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

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

1. Physical Protection

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

2. Change Passwords Regularly

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

3. Set and Update Passwords Reset Information Timely

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

4. Enable Account Lock

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

5. Change Default HTTP and Other Service Ports

We suggest you to change default HTTP and other service ports into any set of numbers

between 1024~65535, reducing the risk of outsiders being able to guess which ports you are using.

6. Enable HTTPS

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

7. Enable Whitelist

We suggest you to enable whitelist function to prevent everyone, except those with specified IP addresses, from accessing the system. Therefore, please be sure to add your computer's IP address and the accompanying equipment's IP address to the whitelist.

8. MAC Address Binding

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

9. Assign Accounts and Privileges Reasonably

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

10. 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.

11. 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.

12. Secure Auditing

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

13. Network Log

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

14. Construct a Safe Network Environment

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

- Disable the port mapping function of the router to avoid direct access to the intranet devices from external network.
- The network should be partitioned and isolated according to the actual network needs. If there are no communication requirements between two sub networks, it is suggested to use VLAN, network GAP and other technologies to partition the network,

so as to achieve the network isolation effect.

- Establish the 802.1x access authentication system to reduce the risk of unauthorized access to private networks.
- It is recommended that you enable your device's firewall or blacklist and whitelist feature to reduce the risk that your device might be attacked.