NetSDK (Intelligent Traffic)

Programming Manual

V1.0.3

Foreword

Purpose

Welcome to use NetSDK (hereinafter referred to as "SDK") programming manual (hereinafter referred to as "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 Intelligent Traffic Camera (ITC), Intelligent Traffic System (ITSE), and IPMECK. For more function modules and data structures, refer to *NetSDK Development Manual*.

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

Readers

- SDK software development engineers
- Project managers
- Product managers

Safety Instruction

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

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

Revision History

Version	Revision Content	Release Time
V1.0.3	Change the callback functions of login and device searching.	March 2020
V1.0.2	Added "2.3 Parking Lot", "3.3 Parking Lot", "4.8 fTransFileCallBack", "4.9 pfAudioDataCallBack."	September 2019
V1.0.1	Deleted some library files in "Table 1-1. "	January 2019
V1.0.0	First release.	December, 2017

Privacy Protection Notice

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

About the Manual

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

Glossary

Term	Definition
ITC	Intelligent Traffic Camera, which is featured by capturing pictures of vehicles and automatically analyzing the traffic events.
ITSE	Intelligent Traffic System, also named as intelligent box, is connected with ITC to provide store the pictures and analyzed data.
IPMECK	Controls the opening and closing of barrier.
Login Handle	A kind of handle that connects with ITC, ITSE and IPMECK. If the connection is successful, the handle is not null (32-bit 4 bytes, 64-bit 8 bytes). This handle is used in most of function modules and will not be null till logged out.
Video Channel	The video of ITC or ITSE is expressed by channel ID. The single lens ITC has only one channel, and multi-lens ITC and ITSE have multiple channels.
Query Handle	A kind of handle that sends query request to ITSE. If the request is successful, the handle is not null (32-bit 4 bytes, 64-bit 8 bytes). It is used to query a particular function module and will not be null until logged out.
Media File	The picture captured by ITC and will be identified and analyzed automatically.
Intelligent Capture	The user needs to capture some scenarios manually. The device analyzes the captured pictures and sends the results to the user.
Intelligent Traffic Event	When the vehicle is passing the traffic junction or the capturing range, the ITC will capture and analyze send the pictures, and then send the results to the user.
Traffic Junction	The traffic junction where the device capture each passing vehicle. The device will analyze and identify the captured pictures and send the results to the user.
Open Barrier Gate	On the traffic junction installed with IPMECK and barrier gate, open the barrier gate to let the vehicle go through the control of IPMECK.
Close Barrier Gate	On the traffic junction installed with IPMECK and barrier gate, close the barrier gate to let the vehicle go through the control of IPMECK.

Table of Contents

1 Overview

1.1 Introduction

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

The following are the main functions:

SDK initialization, device login, real-time monitoring, download of intelligent images, manual capture, report of intelligent traffic event, vehicle flow statistics, and barrier control.

The development kit might be different dependent on the environment.

Table 1-1 Files included in Windows development kit

Library type	Library file name	Library file description
Function library	dhnetsdk.h	Header file
	dhnetsdk.lib	Lib file
	dhnetsdk.dll	Library file
	avnetsdk.dll	Library file
Configuration library	avglobal.h	Header file
	dhconfigsdk.h	Configuration Header file
	dhconfigsdk.lib	Lib file
	dhconfigsdk.dll	Library file
Auxiliary library of playing (coding and decoding)	dhplay.dll	Playing library
	fisheye.dll	Fisheye correction library
Dependent library of "avnetsdk.dll"	Infra.dll	Infrastructure library
	json.dll	JSON library
	NetFramework.dll	Network infrastructure library
	Stream.dll	Media transmission structure package library
	StreamSvr.dll	Streaming service
Auxiliary library of "dhnetsdk.dll"	IvsDrawer.dll	Image display library

Table 1-2 files included in Linux development kit

Library type	Library file name	Library file description
Function library	dhnetsdk.h	Header file
	libdhnetsdk.so	Library file
	libavnetsdk.so	Library file
Configuration library	avglobal.h	Header file
	dhconfigsdk.h	Configuration Header file
	libdhconfigsdk.so	Configuration library
Auxiliary library of "libavnetsdk.so"	libInfra.so	Infrastructure library
	libNetFramework.so	Network infrastructure library
	libStream.so	Media transmission structure package library
	libStreamSvr.so	Streaming service

- The function library and configuration library are necessary libraries.
- The function library is the main body of SDK, which is used for communication interaction between client and products, remotely controls device, queries device data, configures device data information, as well as gets and handles the streams.
- The configuration library packs and parses the structures of configuration functions.
- It is recommended to use auxiliary library of playing (coding and decoding) to parse and play the streams.
- The auxiliary library decodes the audio and video streams for the functions such as monitoring and voice talk, and collects the local audio.
- If the function library includes avnetsdk.dll or libavnetsdk.so, the corresponding dependent library is necessary.

1.2 Applicability

- Recommended memory: No less than 512 M.
- System supported by SDK:
  - ◇ Windows
    Windows 10/Windows 8.1/Windows 7 and Windows Server 2008/2003
  - ◇ Linux
    The common Linux systems such as Red Hat/SUSE
- Access ANPR cameras and other traffic devices:
  ITSE1604-GN5A-D Series, ITSE0400-GN5A-B Series, ITSE0804-GN5B-D Series
- Devices in parking lots:
  Access ANPR camera: ITC215-PW4I Series, ITC215-PW5H Series
  Access ANPR kit: IPMECS-2201D Series, IPMECS-2001B Series
  Parking space detection camera: ITCXX4-PH Series

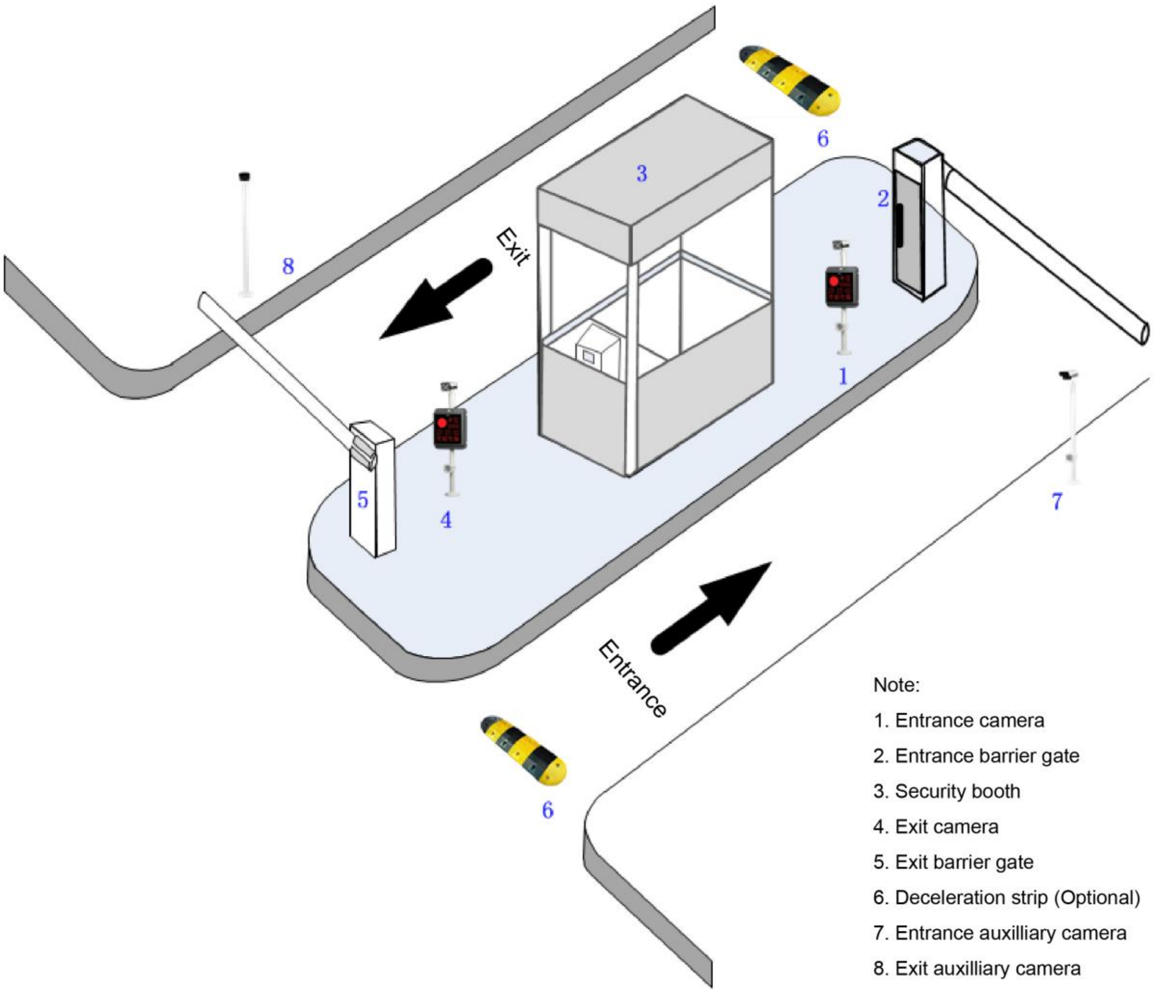
1.3 Application

- ITC and ITSET are installed at the traffic junction, to capture the traffic violations and count the vehicle flow.

Figure 1-1 Application (1)

- ITC, ITSE and IPMECK are installed at the access of parking lot, to control the entrance and exit of the vehicles and monitor the availability of parking space.

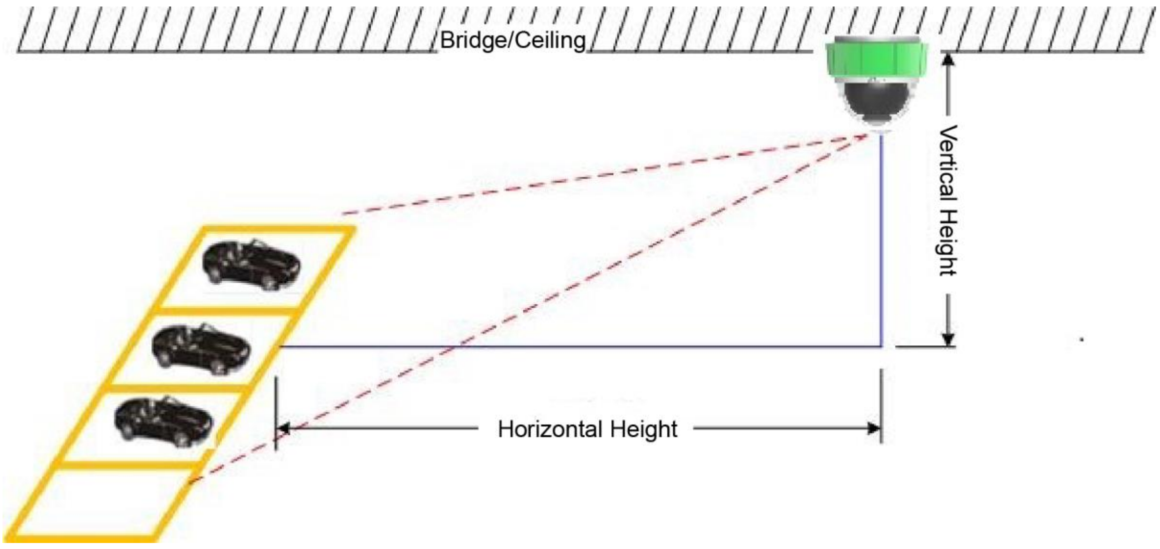
Figure 1-2 Application (2)



The access cameras with the functions of ITC and IPMECK are used to take snapshot and control barrier gates.

- ITC, ITSE and IPMECK are installed in parking lot, to capture and monitor the vehicles, and display the current status of parking spaces.

Figure 1-3 Application (3)



.

2 Function Modules

2.1 General

2.1.1 SDK Initialization

2.1.1.1 Introduction

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

- Initialization occupies some memory.
- Only the first initialization is valid within one process.
- After using this function, call **CLIENT_Cleanup** to release SDK resource.

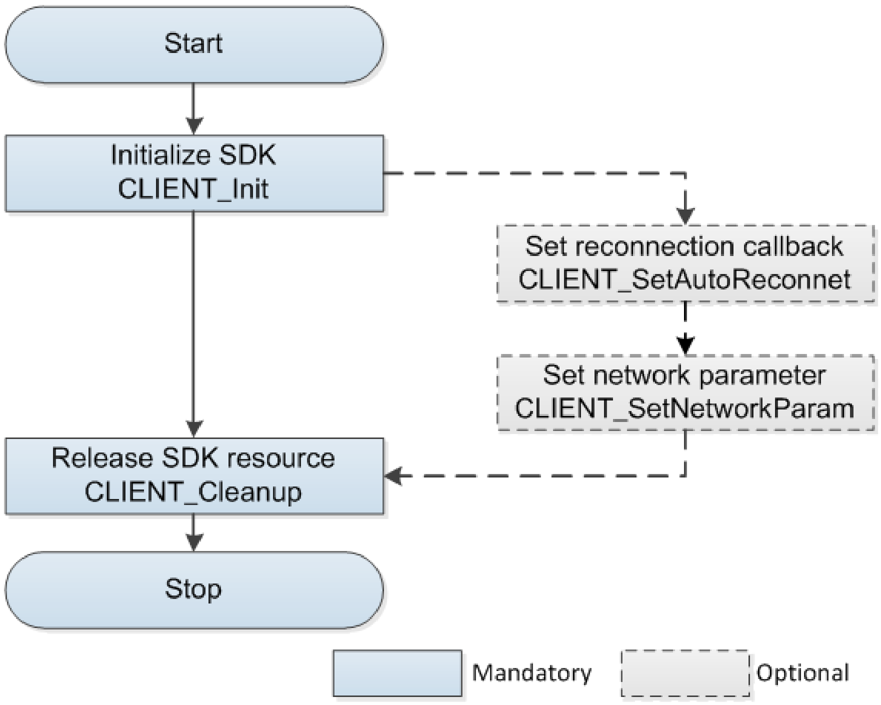
2.1.1.2 Interface Overview

Table 2-1 SDK initialization interfaces

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

2.1.1.3 Process

Figure 2-1 SDK initialization



Process Description

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

Step 2 (Optional) Call **CLIENT_SetAutoReconnect** to set reconnection callback to allow the auto reconnecting after disconnection.

Step 3 (Optional) Call **CLIENT_SetNetworkParam** to set network login parameter that includes connection timeout and connection attempts.

Step 4 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Call **CLIENT_Init** and **CLIENT_Cleanup** in pairs. It supports 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 until succeeded. Only the real-time monitoring, alarm and snapshot subscription can be resumed after reconnection is successful.

2.1.1.4 Example Code

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

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

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

2.1.2 Device Initialization

2.1.2.1 Introduction

The device is uninitialized by default. Initialize the device before using it.

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

2.1.2.2 Interface Overview

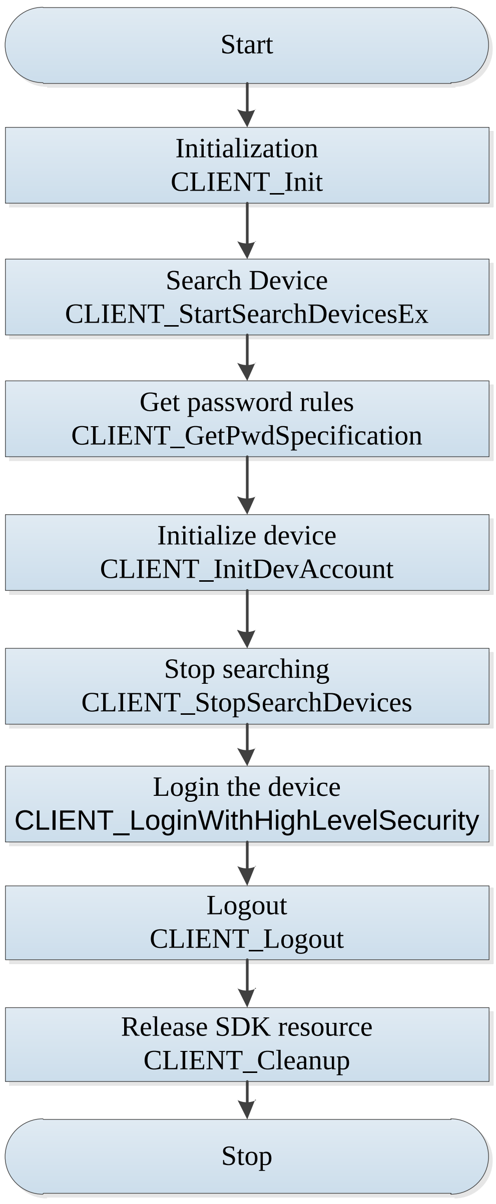
Table 2-2 Device initialization interfaces

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

2.1.2.3 Process

2.1.2.3.1 Device Initialization

Figure 2-2 Device initialization



Process Description

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

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

Multi-thread calling is not supported.

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

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

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

Step 6 Call **CLIENT_LoginWithHighLevelSecurity** and login the admin account with the configured password.

Step 7 After using the function module, call **CLIENT_Logout** to logout the device.

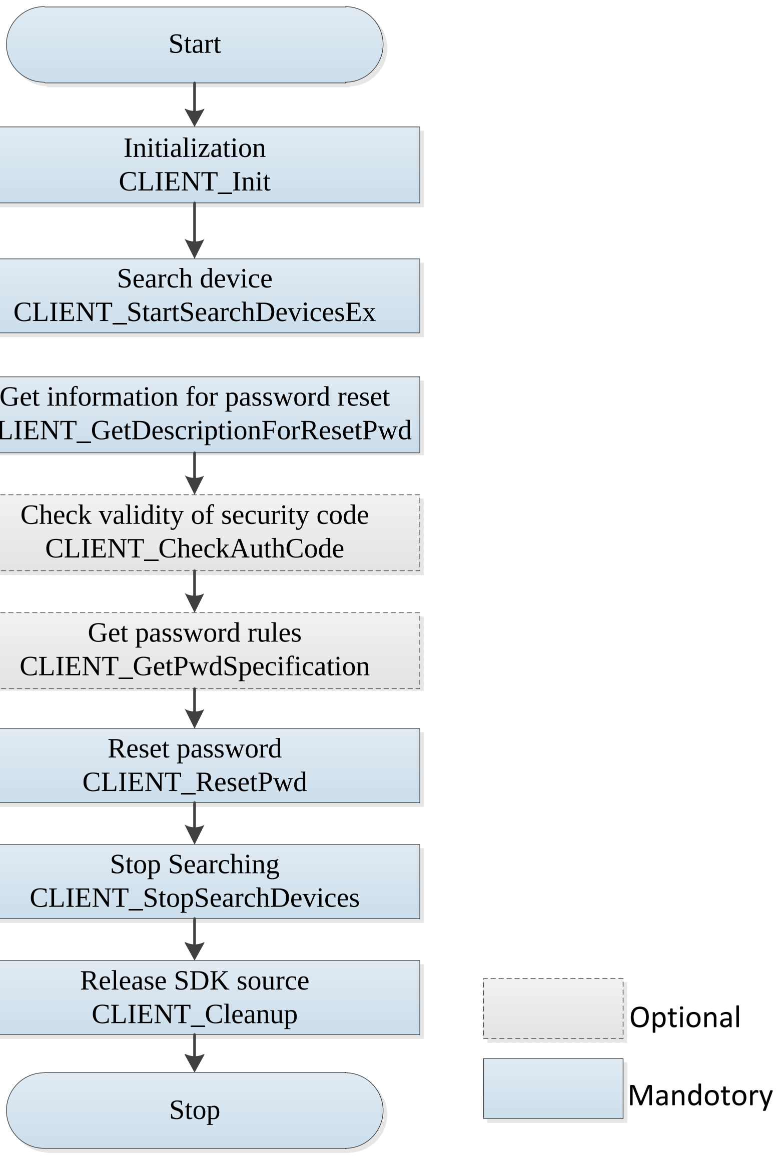
Step 8 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

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

2.1.2.3.2 Password Resetting

Figure 2-3 Password resetting



Process Description

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

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

Multi-thread calling is not supported.

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

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

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

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

Step 7 Call CLIENT_StopSearchDevices to stop searching.

Step 8 Call **CLIENT_LoginWithHighLevelSecurity** and log in to the admin account with the configured password.

Step 9 After using the function module, call **CLIENT_Logout** to log out of the device.

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

Notes for Process

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

2.1.2.4 Example Code

2.1.2.4.1 Device Initialization

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

//Device initialization
NET_IN_INIT_DEVICE_ACCOUNT sInitAccountIn = {sizeof(sInitAccountIn)};
NET_OUT_INIT_DEVICE_ACCOUNT sInitAccountOut = {sizeof(sInitAccountOut)};
sInitAccountIn.byPwdResetWay = 1;//1 stands for password reset by mobile phone number, and 2
stands for password reset by email
strncpy(sInitAccountIn.szMac, szMac, sizeof(sInitAccountIn.szMac) - 1);//Set mac value
strncpy(sInitAccountIn.szUserName, szUserName, sizeof(sInitAccountIn.szUserName) - 1);//Set user
name
strncpy(sInitAccountIn.szPwd, szPwd, sizeof(sInitAccountIn.szPwd) - 1);//Set password
strncpy(sInitAccountIn.szCellPhone, szRig, sizeof(sInitAccountIn.szCellPhone) - 1);//If the byPwdResetWay is set as 1, please set szCellPhone field; if the byPwdResetWay is set as 2, please set sInitAccountIn.szMail field.
CLIENT_InitDevAccount(&sInitAccountIn, &sInitAccountOut, 5000, NULL);
```

2.1.2.4.2 Password Reset

```
//Firstly, call CLIENT_StartSearchDevicesEx to get the device information.
//Get the information for password reset
NET_IN_DESCRIPTION_FOR_RESET_PWD stIn = {sizeof(stIn)};
strncpy(stIn.szMac, szMac, sizeof(stIn.szMac) - 1); //Set mac value
strncpy(stIn.szUserName, szUserName, sizeof(stIn.szUserName) - 1);//Set user name
stIn.byInitStatus = bStstus; //bStstus is the value of return field byInitStatus of device search interface (Callback of CLIENT_SearchDevices and CLIENT_StartSearchDevice and CLIENT_StartSearchDevicesEx, and CLIENT_SearchDevicesByIPs)
NET_OUT_DESCRIPTION_FOR_RESET_PWD stOut = {sizeof(stOut)};
char szTemp[360];
stOut.pQrCode = szTemp;
CLIENT_GetDescriptionForResetPwd(&stIn, &stOut, 3000, NULL);//In the case of single network card, the last parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter. After successful connection, stout will output a QR code with address of stOut.pQrCode. Scan this QR code to get the security code for password reset. This security code will be sent to the reserved mobile phone or email box.
//(Optional) Check the security code
NET_IN_CHECK_AUTHCODE stIn1 = {sizeof(stIn1)};
strncpy(stIn1.szMac, szMac, sizeof(stIn1.szMac) - 1); //Set mac value
strncpy(stIn1.szSecurity, szSecu, sizeof(stIn1.szSecurity) - 1); // szSecu is the security code sent to the reserved mobile phone or email box
NET_OUT_CHECK_AUTHCODE stOut1 = {sizeof(stOut1)};
bRet = CLIENT_CheckAuthCode(&stIn1, &stOut1, 3000, NULL); //In the case of single network card, the last parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter
//Get password rules
NET_IN_PWD_SPECI stIn2 = {sizeof(stIn2)};
strncpy(stIn2.szMac, szMac, sizeof(stIn2.szMac) - 1); //Set mac value
NET_OUT_PWD_SPECI stOut2 = {sizeof(stOut2)};
CLIENT_GetPwdSpecification(&stIn2, &stOut2, 3000, NULL);// In the case of single network card, the last parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter. Set the password according to the rules which are used for preventing user from setting the passwords that are not supported by the device
//Reset password
NET_IN_RESET_PWD stIn3 = {sizeof(stIn3)};
strncpy(stIn3.szMac, szMac, sizeof(stIn3.szMac) - 1); //Set mac value
strncpy(stIn3.szUserName, szUserName, sizeof(stIn3.szUserName) - 1); //Set user name
strncpy(stIn3.szPwd, szPassWd, sizeof(stIn3.szPwd) - 1); //szPassWd is the password reset according to the rules
strncpy(stIn3.szSecurity, szSecu, sizeof(stIn1.szSecurity) - 1); //szSecu is the security code sent to the reserved mobile phone or email box
stIn3.byInitStaus = bStstus; //bStstus is the value of return field byInitStatus of device search interface (Callback of CLIENT_SearchDevices and CLIENT_StartSearchDevice, and CLIENT_SearchDevicesByIPs)
stIn3.byPwdResetWay = bPwdResetWay; // bPwdResetWay is the value of return field byPwdResetWay of device search interface (Callback of CLIENT_SearchDevices and CLIENT_StartSearchDevice, and CLIENT_SearchDevicesByIPs)
NET_OUT_RESET_PWD stOut3 = {sizeof(stOut3)};
CLIENT_ResetPwd(&stIn3, &stOut3, 3000, NULL);//In the case of single network card, the last parameter can be left unfilled; in the case of multiple network card, enter the host PC IP for the last parameter.
```

2.1.3 Device Login

2.1.3.1 Introduction

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

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

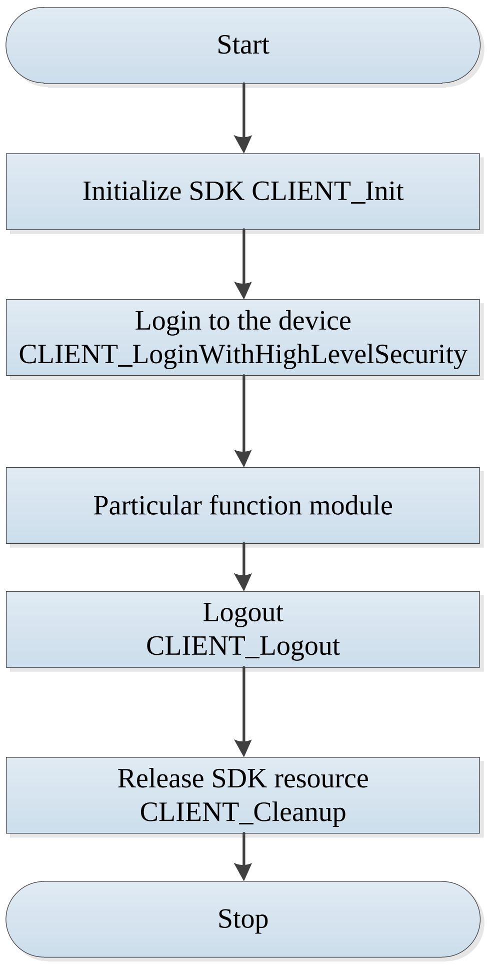
2.1.3.2 Interface Overview

Table 2-3 Device login interfaces

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

2.1.3.3 Process

Figure 2-4 Device login


Start
Initialize SDK CLIENT_Init
Login to the device
CLIENT_LoginWithHighLevelSecurity
Particular function module
Logout
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
Stop

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

- Login failure: It is suggested to check the failure through the error parameter of the login interface.

Table 2-4 Common error code

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

The example code to avoid error code 3 is as follows.

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

For more information about error codes, see "CLIENT_LoginWithHighLevelSecurity interface" in *Network SDK Development Manual.chm*.

2.1.3.4 Example Code

```
NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY stInparam;
memset(&stInparam, 0, sizeof(stInparam));
stInparam.dwSize = sizeof(stInparam);
strncpy(stInparam.szIP, “192.168.1.108”, sizeof(stInparam.szIP) - 1);
strncpy(stInparam.szPassword, “123456”, sizeof(stInparam.szPassword) - 1);
strncpy(stInparam.szUserName, “admin”, sizeof(stInparam.szUserName) - 1);
stInparam.nPort = 37777;
stInparam.emSpecCap = EM_LOGIN_SPEC_CAP_TCP;

NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY stOutparam;
memset(&stOutparam, 0, sizeof(stOutparam));
stOutparam.dwSize = sizeof(stOutparam);
LLONG lLoginID = CLIENT_LoginWithHighLevelSecurity(&stInparam, &stOutparam);
```

2.1.4 Real-time Monitoring

2.1.4.1 Introduction

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

SDK can get the main stream and sub stream from the device once it logged.

- Supports calling the window handle for SDK to directly decode and play the stream (Windows system only).
- Supports calling the real-time stream for 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.

2.1.4.2 Interface Overview

Table 2-5 Real-time monitoring interfaces

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

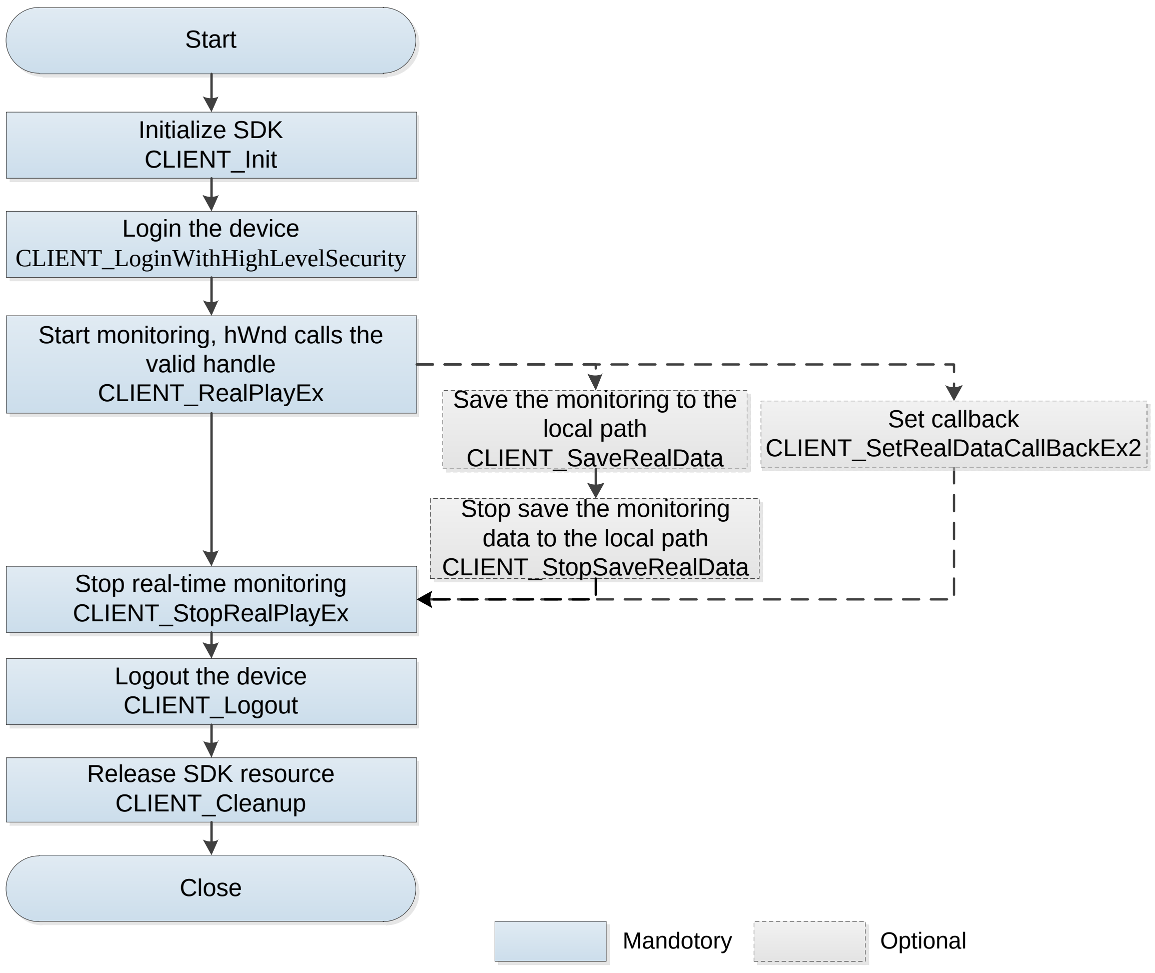
2.1.4.3 Process

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

2.1.4.3.1 SDK Decoding Play

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

Figure 2-5 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 (Optional) If you call **CLIENT_SetRealDataCallBackEx2**, you can choose to save or forward the video file. If save the video file, see the step 4 and step 5.

Step 7 After using the real-time function, call **CLIENT_StopRealPlayEx** to stop real-time monitoring**.**

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

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

Notes for Process

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

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

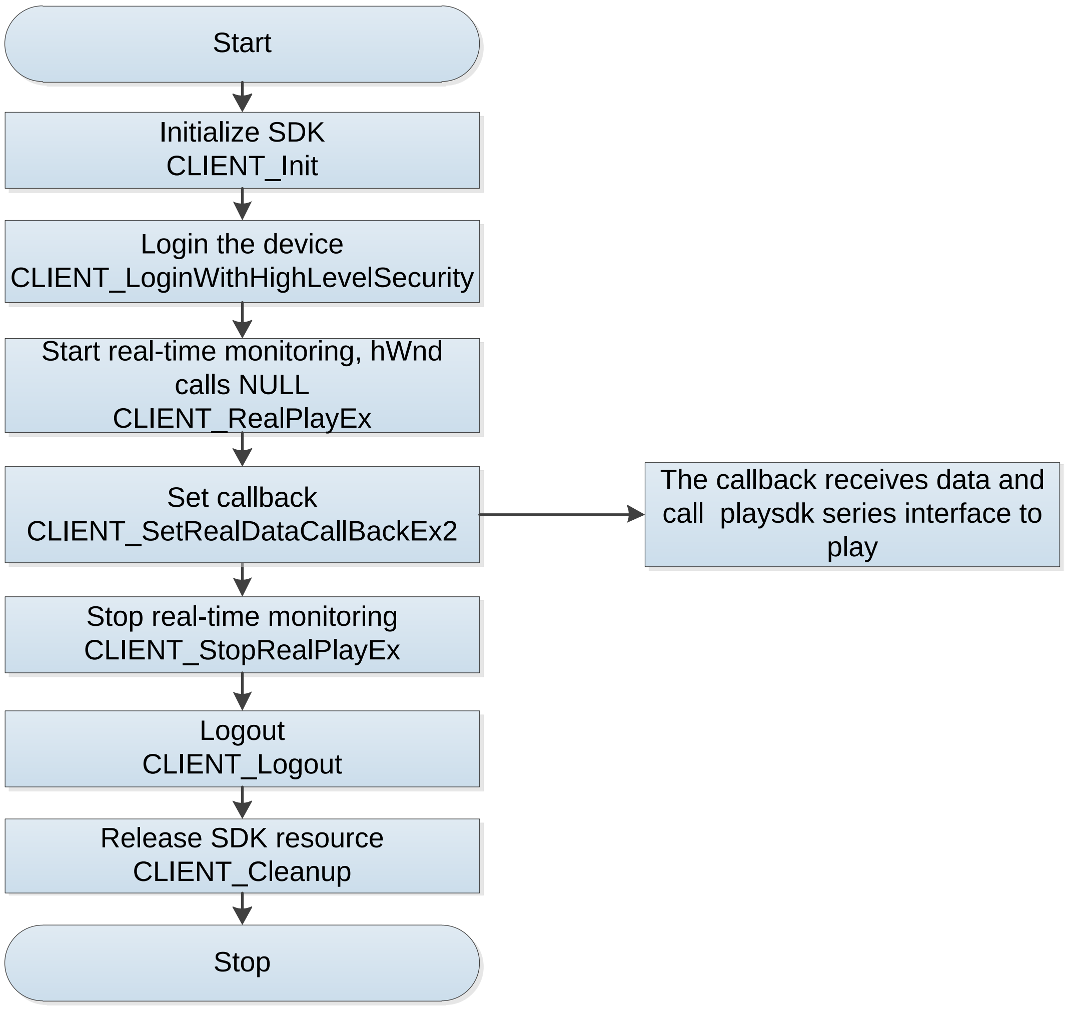
```
NET_PARAM stuNetParam = {0};
stuNetParam. nGetConnInfoTime = 5000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- CLIENT_SetNetworkParam (&stuNetParam);CLIENT_SetNetworkParam (&stuNetParam);
- Failed to repeat opening: For some models, the same channel cannot be opened for multiple times during a login. 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 have 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_FAILD_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 "2.1.4.3.2 Call Third Party Library."

2.1.4.3.2 Call Third Party Library

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

Figure 2-6 Calling the third party library


Start
Initialize SDK
CLIENT_Init
Login the device
CLIENT_LoginWithHighLevelSecurity
Start real-time monitoring, hWnd calls NULL
CLIENT_RealPlayEx
Set callback
CLIENT_SetRealDataCallBackEx2
The callback receives data and call playsdk series interface to play
Stop real-time monitoring
CLIENT_StopRealPlayEx
Logout
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
Stop

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_SetRealDataCallBackEx2** to set the real-time data callback.

Step 5 In the callback, pass the data to PlaySDK to finish decoding.

Step 6 After completing the real-time monitoring, call **CLIENT_StopRealPlayEx** to stop real-time monitoring.

Step 7 After using the function module, call **CLIENT_Logout** to 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 cache size (the PLAY_OpenStream interface in playsdk) for decoding. If the stream resolution value is big, it is recommended to modify the parameter value smaller such as 3 M.

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

2.1.4.4 Example Code

2.1.4.4.1 SDK Decoding Play

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

2.1.4.4.2 Call Play Library

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

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

```
if (0 != lRealHandle)
{
        CLIENT_StopRealPlayEx(lRealHandle);
}

void CALLBACK RealDataCallBackEx(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer,
DWORD dwBufSize, LLONG param, LDWORD dwUser)
{
    // Call PlaySDK interface to get the stream data from the device. See SDK monitoring demo source
    data for more details.
            printf("receive real data, param: lRealHandle[%p], dwDataType[%d], pBuffer[%p],
dwBufSize[%d]\n", lRealHandle, dwDataType, pBuffer, dwBufSize);
}
```

2.2 Traffic Junction

2.2.1 Download of Media File

2.2.1.1 Introduction

You can get the decoded pictures from ITSE through SDK and saves into the local path for further use.

To download the media files, the SDK connects to the device firstly. It sends query command per the query condition of media file and sends the download command after getting the query result to the device, and then the device will send the media files and decoded data to you.

2.2.1.2 Interface Overview

Table 2-6 Downloading media file interfaces

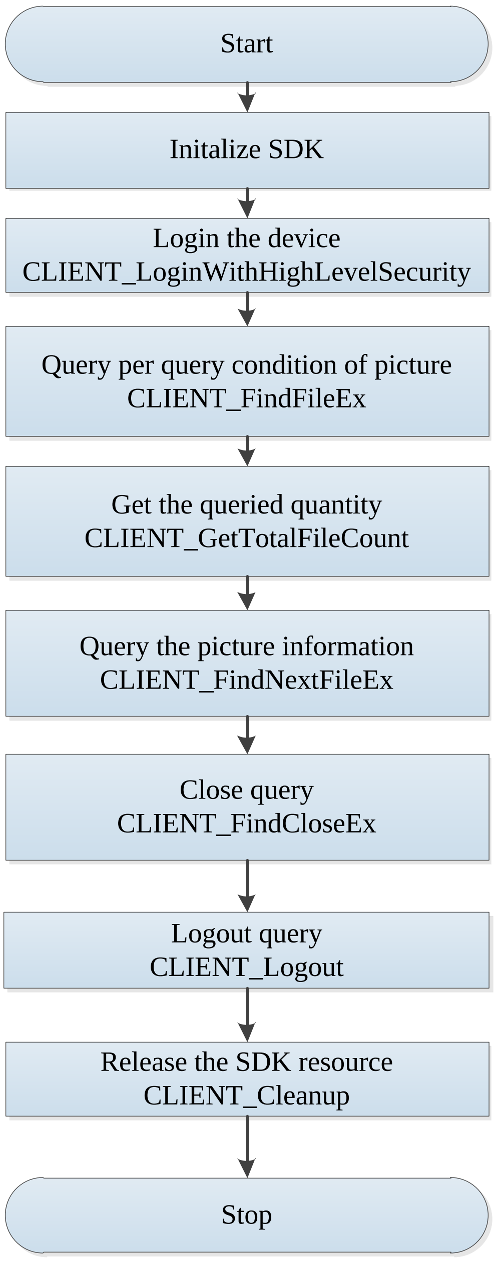
Interface	**Implication**
CLIENT_FindFileEx	Query per the query condition of file.
CLIENT_GetTotalFileCount	Get the queried quantity.
CLIENT_FindNextFileEx	Query the information of media file.
CLIENT_FindCloseEx	Close the query.
CLIENT_DownloadMediaFile	Download the media file.
CLIENT_StopDownloadMediaFile	Stop the download.

2.2.1.3 Process

The process of this function module is consisted of querying and downloading the media file.

2.2.1.3.1 Query of Media File

Figure 2-7 Querying the media file



Process Description

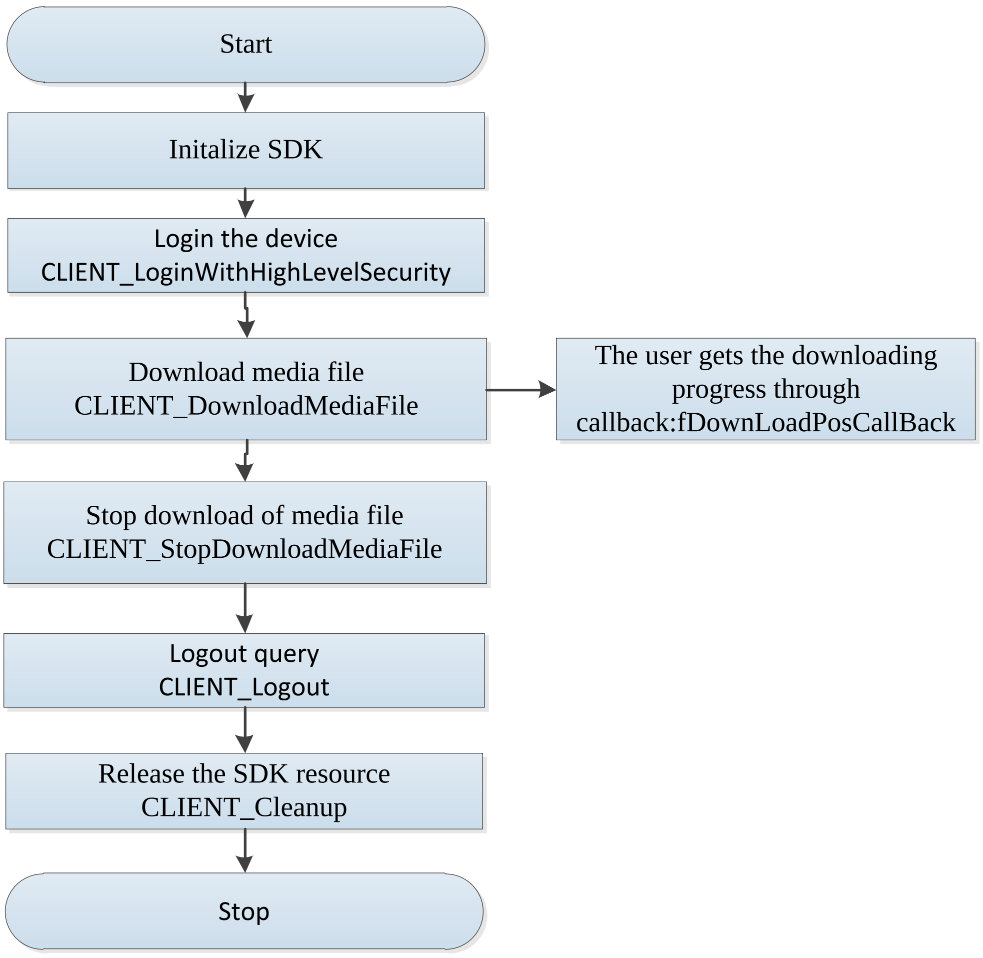
Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_FindFileEx** to query per the query condition of media file.
Step 4 Call **CLIENT_GetTotalFileCount** to get the queried total number.
Step 5 Call **CLIENT_FindNextFileExCall** to review information of all the files.
Step 6 Call **CLIENT_FindCloseEx** to close query.
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

- Applicable device
  This process applies to ITSE devices. Please be noted that ITC only captures and identifies pictures and it is not capable of storing the data
- Parameters
  Use DH_FILE_QUERY_TRAFFICCAR_EX for parameter emType in CLIENT_FindFileEx, and the corresponding structure is MEDIA_QUERY_TRAFFICCAR_PARAM_EX. Use the corresponding structure MEDIAFILE_TRAFFICCAR_INFO_EX for interface CLIENT_FindNextFileEx.

2.2.1.3.2 Download of Media File

Figure 2-8 Downloading the media file


Start
Initalize SDK
Login the device
CLIENT_LoginWithHighLevelSecurity
Download media file
CLIENT_DownloadMediaFile
The user gets the downloading progress through callback:fDownLoadPosCallBack
Stop download of media file
CLIENT_StopDownloadMediaFile
Logout query
CLIENT_Logout
Release the SDK resource
CLIENT_Cleanup
Stop

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_DownloadMediaFile** to download the media file.
Step 4 Call **CLIENT_StopDownloadMediaFile** to close the download.
Step 5 After using the function module, call **CLIENT_Logout** to log out of the device.
Step 6 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Applicable device
  ITSE device. Please be noted that ITC only captures and identifies pictures and it is not capable of storing the data.
- Parameters
  Use only DH_FILE_QUERY_TRAFFICCAR for parameter emType in CLIENT_DownloadMediaFile, and the DH_FILE_QUERY_TRAFFICCAR_EX is not supported. The parameter lpMediaFileInfo is obtained through querying the media file.

2.2.1.4 Example Code

2.2.1.4.1 Query of Media File

```
int main()
{
        …………
        //Query condition of media file
MEDIA_QUERY_TRAFFICCAR_PARAM_EX stuCondition = {0};
stuCondition.dwSize = sizeof(MEDIA_QUERY_TRAFFICCAR_PARAM_EX);
stuCondition.stuParam.nMediaType = 1;
        …………
        //Query the media file
LLONG lFindHandle = CLIENT_FindFileEx(lLoginHandle, DH_FILE_QUERY_TRAFFICCAR_EX,
(void*)&stuCondition, NULL);
if(NULL == lFindHandle)
{
                printf("CLIENT_FindFileEx: failed! Error code: %x.\n", CLIENT_GetLastError());
                return -1;
}
int nCount = 0;
//Gets the quantity of queried media files
BOOL bRet = CLIENT_GetTotalFileCount(lFindHandle,&nCount,NULL);
if(FLASE == bRet)
{
                printf("CLIENT_GetTotalFileCount: failed! Error code: %x.\n", CLIENT_GetLastError());
                return -2;
}
//Review one queried media file per one time
int nMaxConut = 1;
do
{
MEDIAFILE_TRAFFICCAR_INFO_EX mediaFileInfo = {0};
mediaFileInfo.dwSize = sizeof(MEDIAFILE_TRAFFICCAR_INFO_EX);
//Query a single media file
bRet = CLIENT_FindNextFileEx(lFindHandle, nMaxConut, (void*)&mediaFileInfo,
sizeof(MEDIAFILE_TRAFFICCAR_INFO_EX), NULL);
if(FALSE == bRet)
{
                printf("CLIENT_FindNextFileEx: failed! Error code: %x.\n", CLIENT_GetLastError());
}
}
While ((nCount -= nMaxConut) > 0);
//Close query
bRet = CLIENT_FindCloseEx(lFindHandle);
if(FALSE == bRet)
{
        printf("CLIENT_FindCloseEx: failed! Error code: %x.\n", CLIENT_GetLastError());
        return -3;
}
}
```

2.2.1.4.2 Download of Media File

```
int main()
{
         …………
//Query the obtained media file
         MEDIAFILE_TRAFFICCAR_INFO info = mediaFileInfo.stuInfo;
         //Download the media file
LLONG lDownloadHandle = CLIENT_DownloadMediaFile(lLoginHandle,
DH_FILE_QUERY_TRAFFICCAR, (void*)&info, szFileName, DownLoadPosCallBack, NULL, NULL);
if(NULL == lDownloadHandle)
{
printf("CLIENT_DownloadMediaFile: failed! Error code: %x.\n", CLIENT_GetLastError());
}
Sleep(5000);
//Close download
BOOL bRet = CLIENT_StopDownloadMediaFile(lDownloadHandle);
if(FALSE == bRet)
{
         printf("CLIENT_StopDownloadMediaFile: failed! Error code: %x.\n", CLIENT_GetLastError());
}
}
//Download progress callback
void CALLBACK DownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD
dwDownLoadSize, LDWORD dwUser)
{
        if (dwDownLoadSize == -1) //Download finished
        {
            printf("lPlayHandle: %p Download end!\n", lPlayHandle);
}
}
```

2.2.2 Manual Capture

2.2.2.1 Introduction

You can send the command through SDK to ITC or ITSE to capture pictures. The device will automatically analyze the pictures and report to you.

This function mainly applies to analyze the vehicles, detect if the vehicles have broken any regulations, and save the vehicles information.

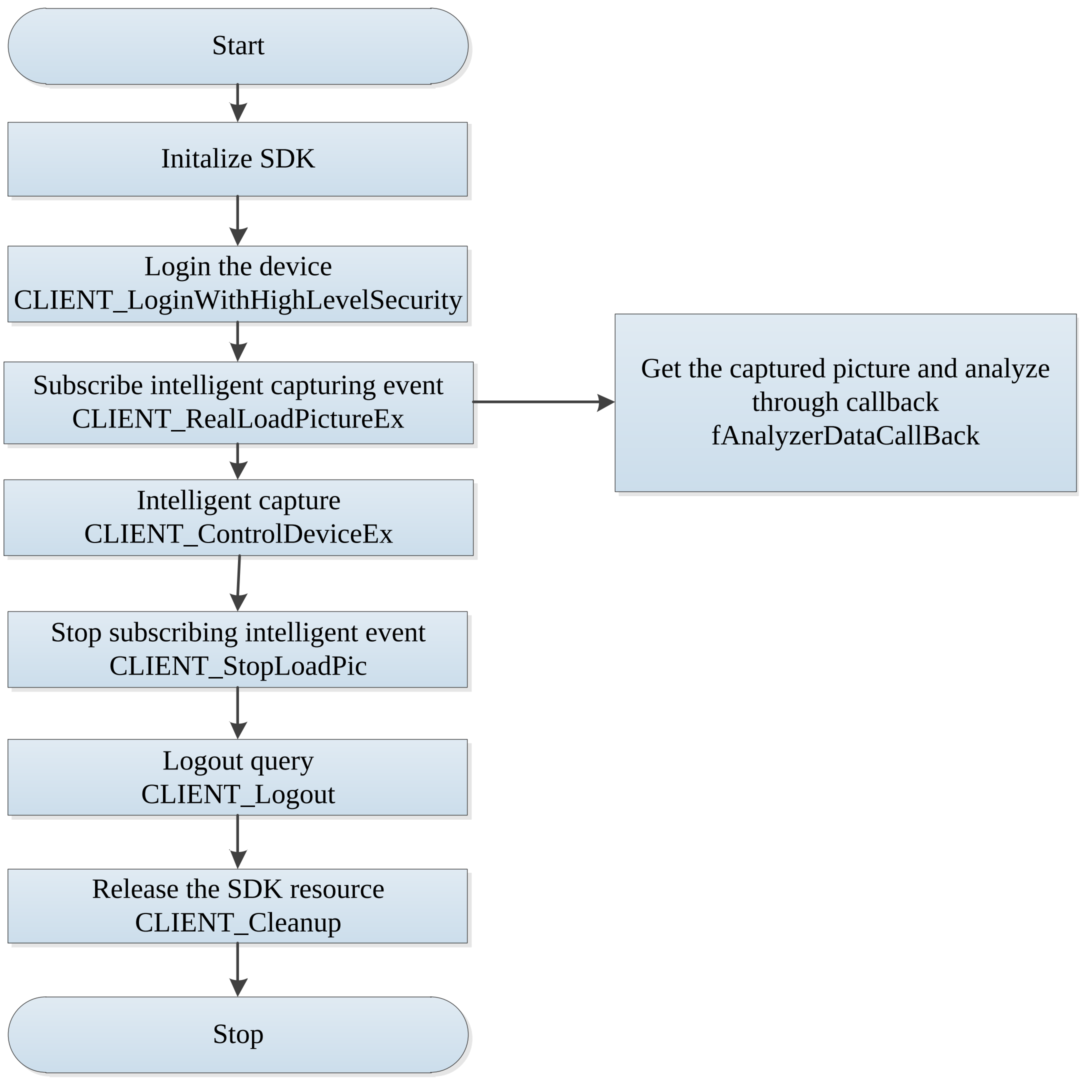
2.2.2.2 Interface Overview

Table 2-7 Manual capture interfaces

Interface	**Implication**
CLIENT_RealLoadPictureEx	Subscribe intelligent traffic event.
CLIENT_ControlDeviceEx	Manual capture.
CLIENT_StopLoadPic	Stop subscribing intelligent traffic event.

2.2.2.3 Process

Figure 2-9 Manual capture



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 start subscribing intelligent traffic event.
Step 4 Call **CLIENT_ControlDeviceEx** to trigger intelligent capturing. Set parameter emType as DH_MANUAL_SNAP.
Step 5 Inform you of manual capturing event through the callback fAnalyzerDataCallBack.
Step 6 Call **CLIENT_StopLoadPic** to stop subscribing intelligent event.
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

Setting of cache for receiving pictures:

Because SDK default cache is 2M, when the data is over 2M, call **CLIENT_SetNetworkParam** to set the receiving cache; otherwise the data pack will be lost.

2.2.2.4 Example Code

```
int main()
{
        …………
        //Subscribe intelligent capturing event
        LLONG lAnalyerHandle = CLIENT_RealLoadPictureEx(lLoginHandle, 0,
(DWORD)EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, NULL, NULL);
        if(NULL == lAnalyerHandle)
        {
                printf("CLIENT_RealLoadPictureEx: failed! Error code %x.\n", CLIENT_GetLastError());
                return -1;
        }
        MANUAL_SNAP_PARAMETER stuManualSnap = {0};
        stuManualSnap.nChannel = 0;
        sprintf((char*)stuManualSnap.bySequence,"abc");
        //Intelligent capturing
        BOOL bRet = CLIENT_ControlDeviceEx(lLoginHandle,DH_MANUAL_SNAP,&stuManualSnap);
        if(FALSE == bRet)
        {
                printf("CLIENT_ControlDeviceEx: failed! Error code %x.\n", CLIENT_GetLastError());
                return -2;
        }
Sleep(5000);
        //Stop subscribing intelligent capturing event
        BOOL bRet = CLIENT_StopLoadPic(lAnalyerHandle);
        if(FALSE == bRet)
        {
                printf("CLIENT_StopLoadPic: failed! Error code %x.\n", CLIENT_GetLastError());
                return -3;
        }
        return 0;
}
//Callback of intelligent capturing
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void*
pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
        switch(dwAlarmType)
        {
                case EVENT_IVS_TRAFFIC_MANUALSNAP:
```

```
{
                    DEV_EVENT_TRAFFIC_MANUALSNAP_INFO*              pInfo              =
                    (DEV_EVENT_TRAFFIC_MANUALSNAP_INFO*)pAlarmInfo;
                    printf("ManualSnapNo: %s", (char*)pInfo-> szManualSnapNo);
                        …………
                   break;
              }
         default:
              break;
    }
    return 0;
}
```

2.2.3 Upload of Intelligent Traffic Event

2.2.3.1 Introduction

The device decodes the real-time stream and sends the detected intelligent traffic event to you. The event includes the situations such as traffic violation, availability of parking space.

To upload the event, SDK connects to the device and subscribe the intelligent event. The device will send the event to SDK once such event has been detected.

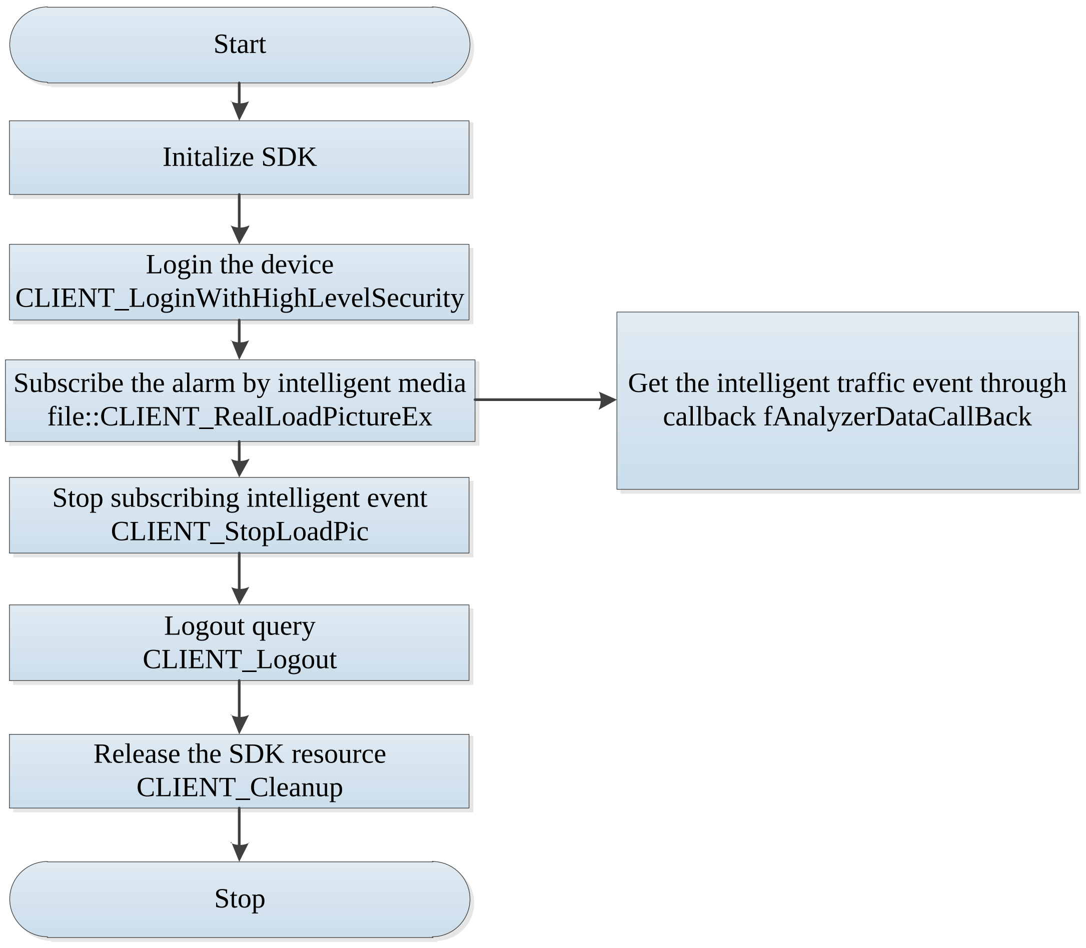
2.2.3.2 Interface Overview

Table 2-8 Intelligent traffic event uploading interfaces

Interface	Description
CLIENT_RealLoadPictureEx	Subscribe intelligent traffic event.
CLIENT_StopLoadPic	Stop subscribing intelligent traffic event.

2.2.3.3 Process

Figure 2-10 Uploading intelligent traffic event


Start
Initalize SDK
Login the device
CLIENT_LoginWithHighLevelSecurity
Subscribe the alarm by intelligent media file::CLIENT_RealLoadPictureEx
Get the intelligent traffic event through callback fAnalyzerDataCallBack
Stop subscribing intelligent event
CLIENT_StopLoadPic
Logout query
CLIENT_Logout
Release the SDK resource
CLIENT_Cleanup
Stop

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 the intelligent traffic event.
Step 4 Get the uploaded event through callback fAnalyzerDataCallBack and send to you.
Step 5 Call **CLIENT_StopLoadPic** to stop subscribing event.
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

- Subscribed event type: Support subscribing all event type (EVENT_IVS_ALL) at the same time or subscribing a single event type.
- Setting of cache for receiving pictures: Because SDK default cache is 2M, when the data is over 2M, call **CLIENT_SetNetworkParam** to set the receiving cache, otherwise the data pack will be lost.
- Setting of whether to receive pictures: Because some devices have 3G or 4G network, when SDK is connecting to the device, if it does not need to receive picture, set the

parameter bNeedPicFile as False in interface CLIENT_ RealLoadPictureEx to only receive the intelligent event without picture.

2.2.3.4 Example Code

```
int main()
{
    …………
    //Subscribe the upload of intelligent traffic event
    LLONG lAnalyerHandle = CLIENT_RealLoadPictureEx(lLoginHandle, 0,
(DWORD)EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, NULL, NULL);
    if(NULL == lAnalyerHandle)
    {
        printf("CLIENT_RealLoadPictureEx: failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    Sleep(5000);
    //Stop subscribing the upload of intelligent traffic event
    BOOL bRet = CLIENT_StopLoadPic(lAnalyerHandle);
    if(FALSE == bRet)
    {
        printf("CLIENT_StopLoadPic: failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    return 0;
}
//Callback for upload of intelligent traffic event
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void*
pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    switch(dwAlarmType)
    {
        …………
        case EVENT_IVS_TRAFFIC_RUNREDLIGHT: // Event of running the red light
            {
                DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO* pInfo =
                (DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO*)pAlarmInfo;
                …………
              break;
            }
        …………
```

```
            default:
                  break;
      }
      return 0;
}
```

2.2.4 Vehicle Flow Statistics

2.2.4.1 Introduction

ITC device counts on all the passing vehicles to analyze the traffic status and directly send the result to you or to ITSE that sends to you.

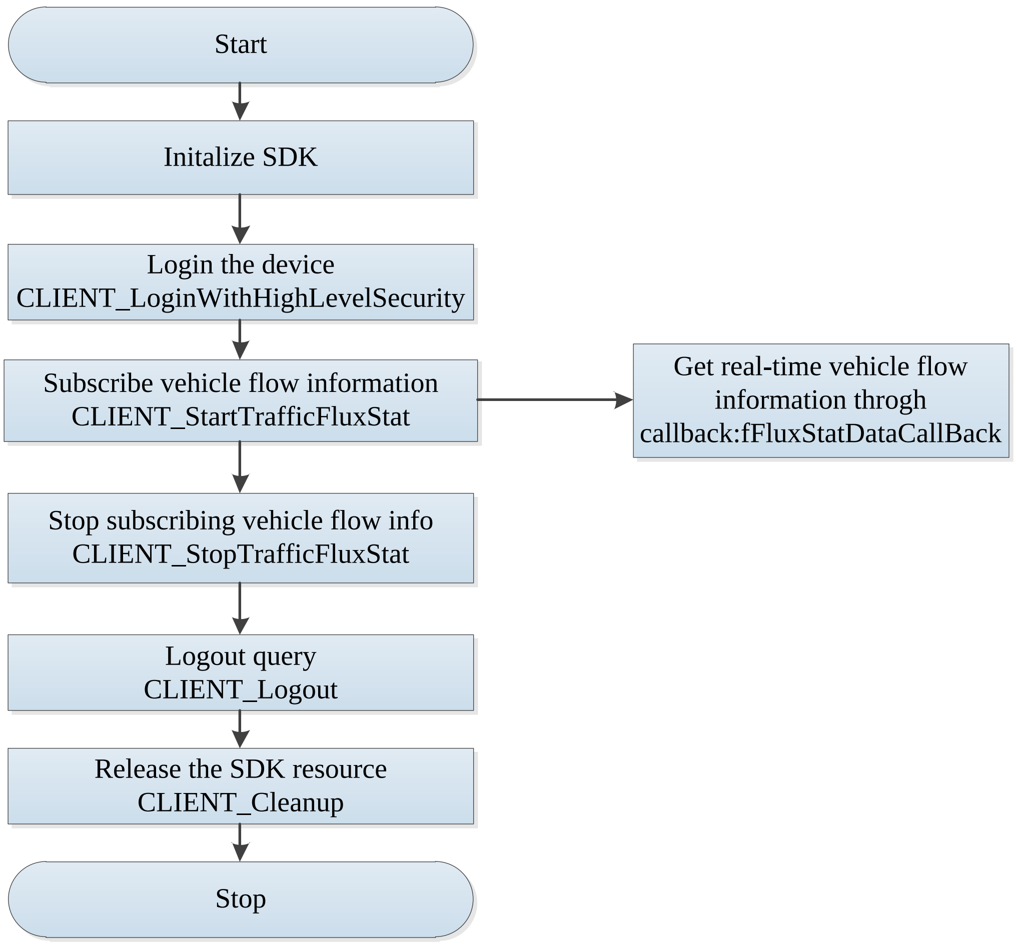
2.2.4.2 Interface Overview

Table 2-9 Vehicle flow statistics interfaces

Interface	Description
CLIENT_StartTrafficFluxStat	Subscribe intelligent traffic event
CLIENT_StopTrafficFluxStat	Stop subscribing intelligent traffic event

2.2.4.3 Process

Figure 2-11 Vehicle flow statistics


Start
Initalize SDK
Login the device
CLIENT_LoginWithHighLevelSecurity
Subscribe vehicle flow information
CLIENT_StartTrafficFluxStat
Get real-time vehicle flow information throgh callback:fFluxStatDataCallBack
Stop subscribing vehicle flow info
CLIENT_StopTrafficFluxStat
Logout query
CLIENT_Logout
Release the SDK resource
CLIENT_Cleanup
Stop

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_StartTrafficFluxStat** to subscribe the vehicle flow information.
Step 4 Get the vehicles information uploaded by ITC or ITSE through callback fFluxStatDataCallBack and inform you.
Step 5 Call **CLIENT_StopTrafficFluxStat** to stop subscribing the vehicle flow information.
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

Callback data type: The parameter pEventInfo corresponds to structure of DEV_EVENT_TRAFFIC_FLOWSTAT_INFO.

2.2.4.4 Example Code

```
int main()
```

```
{
    …………
    NET_IN_TRAFFICFLUXSTAT stuIn = {0};
    stuIn.dwSize = sizeof(NET_IN_TRAFFICFLUXSTAT);
    stuIn.cbData = FluxStatDataCallBack;
    NET_OUT_TRAFFICFLUXSTAT stuOut = {0};
    stuOut.dwSize = sizeof(NET_OUT_TRAFFICFLUXSTAT);
    //Subscribe the vehicle flow statistics
    LLONG lFluxStatHandle = CLIENT_StartTrafficFluxStat(lLoginHandle, &stuIn, &stuOut);
    if(NULL == lFluxStatHandle)
    {
            printf("CLIENT_StartTrafficFluxStat: failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    Sleep(5000);
    //Stop subscribing the vehicle flow statistics
    BOOL bRet = CLIENT_StopTrafficFluxStat(lFluxStatHandle);
    if(FALSE == bRet)
    {
          printf("CLIENT_StopTrafficFluxStat: failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    return 0;
}
//Callback of vehicle flow statistics
int CALLBACK FluxStatDataCallBack (LLONG lFluxStatHandle, DWORD dwEventType, void* pEventInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
        DEV_EVENT_TRAFFIC_FLOWSTAT_INFO* pInfo = (DEV_EVENT_TRAFFIC_FLOWSTAT_INFO*)pEventInfo;
    …………
    return 0;
}
```

2.3 Parking Lot

2.3.1 Barrier Control

2.3.1.1 Introduction

IPMECK device can control the opening and closing operations of road barrier. You can send the command through SDK to IPMECK for the manual barrier control. For example:

- Issue the configuration to IPMECK through SDK, to set the barrier normal open or normal close, and set the period.
- Barrier will opened in case of vehicle location event (dominant) or traffic junction event to link opening barrier gate.

The barrier control mainly applies to the places such as parking lot, toll gate, and gate of district.

Applicable device: IPMECK device.

2.3.1.2 Interface Overview

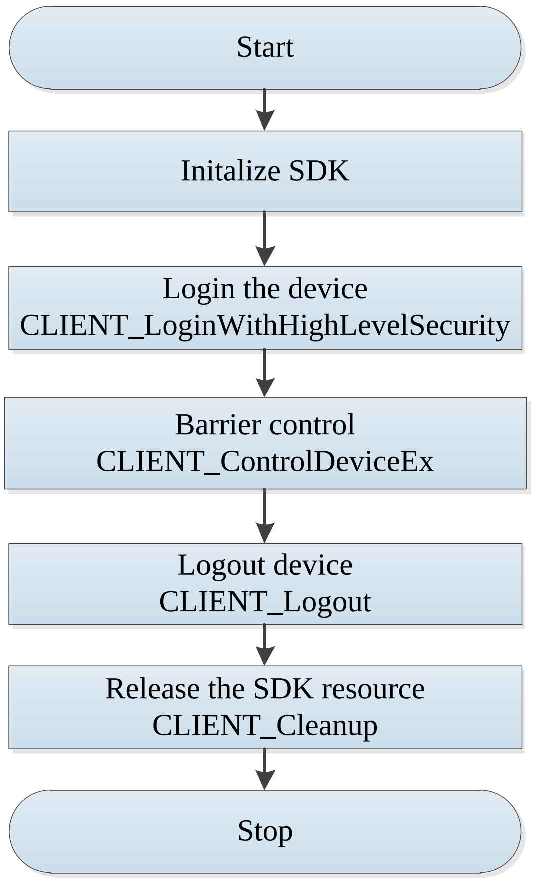
Table 2-10 Barrier control interfaces

Interface	Description
CLIENT_ControlDeviceEx	Barrier control.
CLIENT_GetConfig	Get the configuration of barrier gate.
CLIENT_SetConfig	Isuee the configuration of barrier gate.
CLIENT_SetDVRMessCallBack	Set alarm callback function.
CLIENT_StartListenEx	Subscribe vehicle location event.
CLIENT_StopListen	Stop subscribing vehicle location event.
CLIENT_RealLoadPictureEx	Subscribe traffic junction event.
CLIENT_StopLoadPic	Stop subscribing traffic event.

2.3.1.3 Process

2.3.1.3.1 Manual Barrier Control

Figure 2-12 Manual barrier control

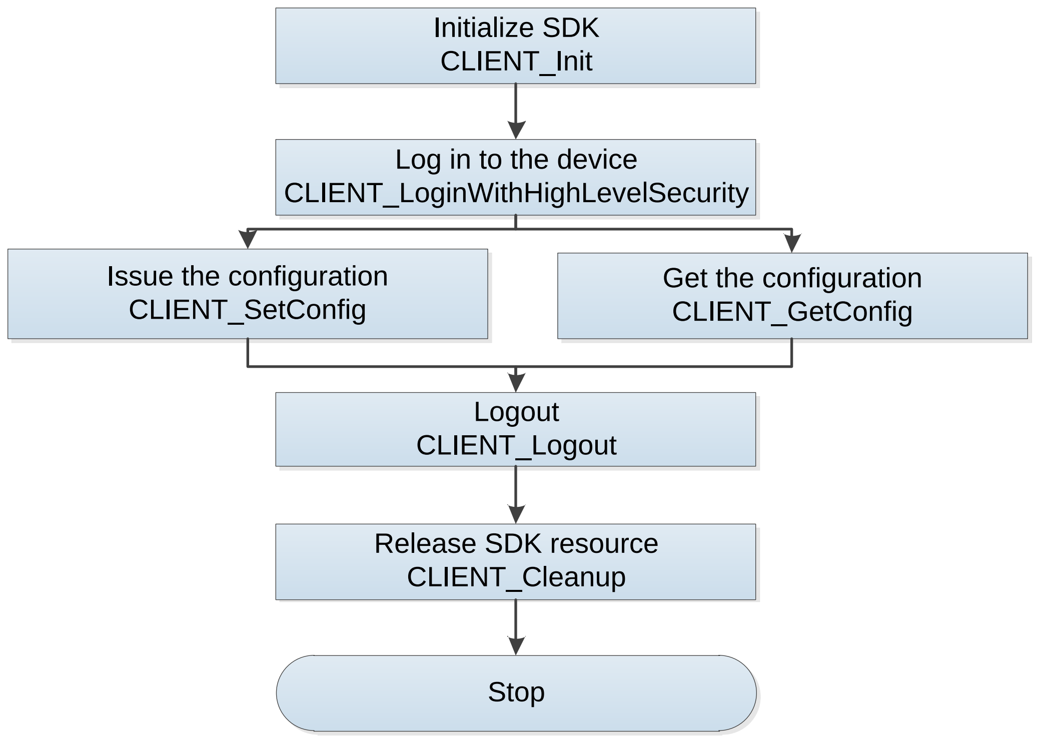


Process Description

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_ControlDeviceEx** to open or close the barrier.
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.

2.3.1.3.2 Barrier Control Configuration

Figure 2-13 Barrier control configuration



Process Description

Setting

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

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

Step 3 Call **CLIENT_SetConfig** to set the period of barrier normally open enable or barrier normally open mode.

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.

Getting

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

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

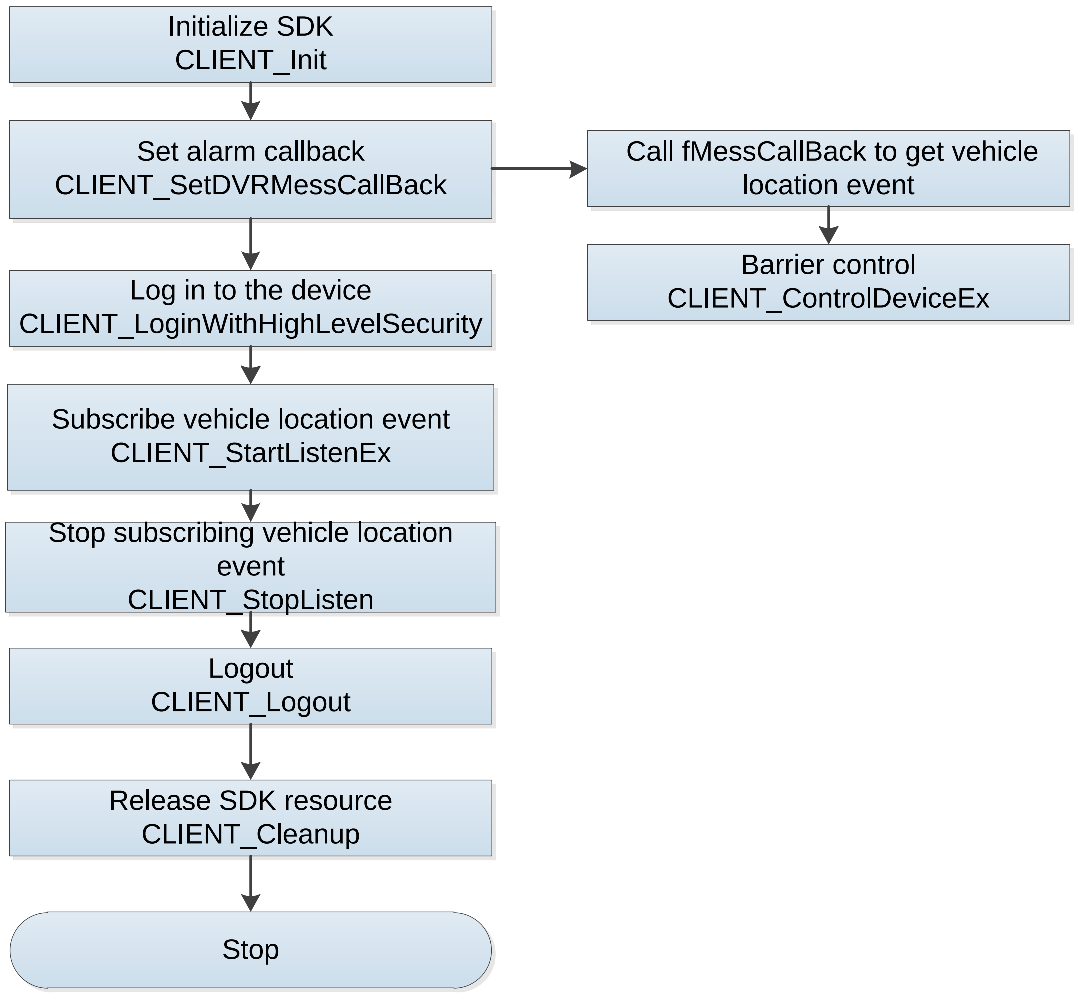
Step 3 Call **CLIENT_GetConfig** to get the configuration of the period of barrier normally open enable or barrier normally open mode.

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.

2.3.1.3.3 Vehicle Location Event links Barrier Control

Figure 2-14 Vehicle location event linking barrier control



Process Description

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

Step 2 Call **CLIENT_SetDVRMessCallBack** to set alarm callback function. When vehicle location comes, call **CLIENT_ControlDeviceEx** to open barrier gate.

Step 3 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.

Step 4 Call **CLIENT_StartListenEx** to subscribe vehicle location event.

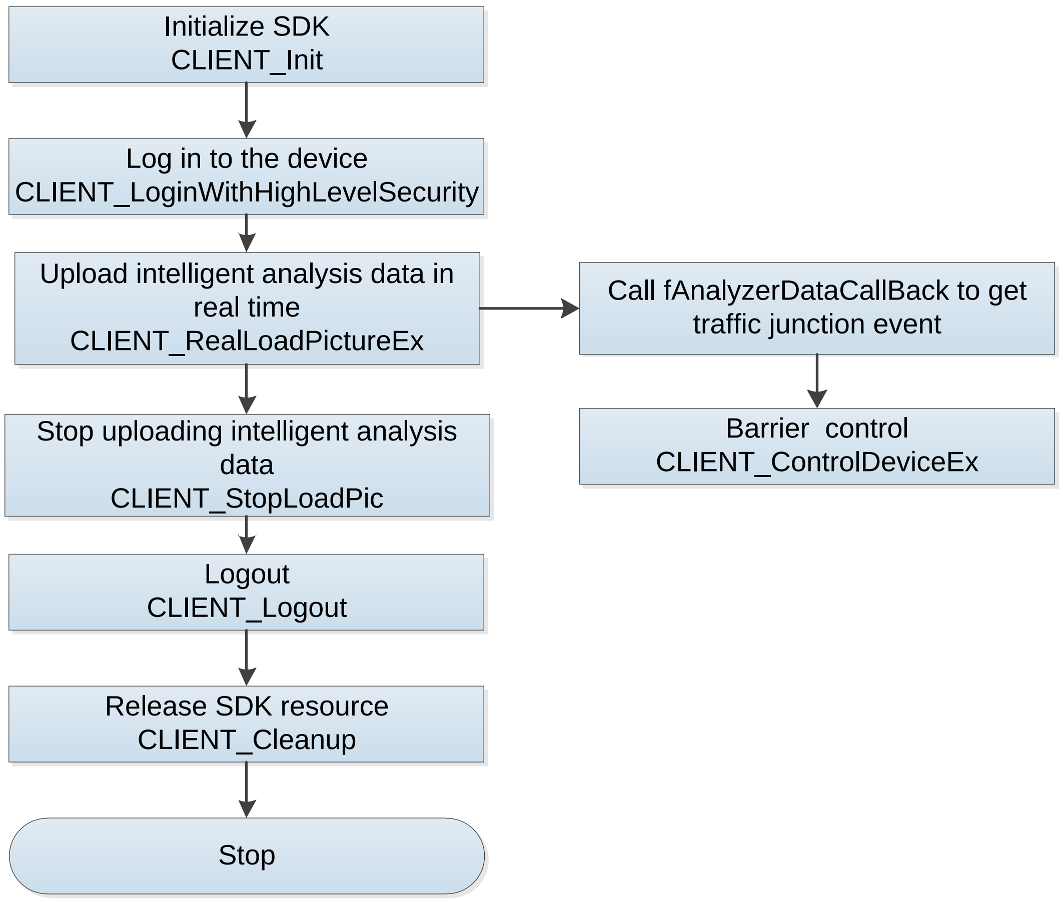
Step 5 Call **CLIENT_StopListen** to stop subscribing vehicle location event.

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.

2.3.1.3.4 Traffic Junction Event links Barrier Control

Figure 2-15 Traffic junction event linking barrier 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_RealLoadPictureEx** to subscribe traffic junction event. When an event is triggeried **fAnalyzerDataCallBack** calls **CLIENT_ControlDeviceEx** to open the barrier gate.
Step 4 Call **CLIENT_StopLoadPic** to stop subscribing traffic junction event.
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.

2.3.1.4 Example Code

2.3.1.4.1 Manual Barrier Control

```
int main()
{
    …………
    NET_CTRL_OPEN_STROBE stuOpenStrobe = {0};
    stuOpenStrobe.dwSize = sizeof(NET_CTRL_OPEN_STROBE);
    stuOpenStrobe.nChannelId = 0;
    sprintf(stuOpenStrobe.szPlateNumber,"123456");
    //Open the barrier gate
    BOOL bRet = CLIENT_ControlDeviceEx(lLoginHandle,DH_CTRL_OPEN_STROBE,&stuOpenStrobe);
    if(FALSE == bRet)
    {
        printf("CLIENT_ControlDeviceEx: Open strobe failed! Error code %x.\n", CLIENT_GetLastError());
        return -1;
    }
    NET_CTRL_CLOSE_STROBE stuCloseStrobe = {0};
    stuCloseStrobe.dwSize = sizeof(NET_CTRL_CLOSE_STROBE);
    stuCloseStrobe.nChannelId = 0;
    //Close the barrier gate
    bRet = CLIENT_ControlDeviceEx(lLoginHandle,DH_CTRL_CLOSE_STROBE,&stuCloseStrobe);
    if(FALSE == bRet)
    {
        printf("CLIENT_ControlDeviceEx: Close strobe failed! Error code %x.\n", CLIENT_GetLastError());
        return -2;
    }
    return 0;
}
```

2.3.1.4.2 Barrier Control Configuration

```
//Get barrier gate configuration
NET_CFG_TRAFFICSTROBE_INFO m_stuTrafficstrobeInfo = {sizeof(m_ stuTrafficstrobeInfo)};
BOOL bRet = CLIENT_GetConfig(m_LoginID, NET_EM_CFG_TRAFFICSTROBE,m_nChannel,&m_
stuTrafficstrobeInfo,sizeof(m_ stuTrafficstrobeInfo), 5000);
if (! bRet)
{
    //Failed
}
//Set barrier gate configuration
NET_CFG_TRAFFICSTROBE_INFO m_ stuTrafficstrobeInfo = {sizeof(m_ stuTrafficstrobeInfo)};
…….
BOOL bRet = CLIENT_SetConfig(m_LoginID, NET_EM_CFG_TRAFFICSTROBE,m_nChannel,&m_
stuTrafficstrobeInfo,sizeof(m_ stuTrafficstrobeInfo), 5000);
if (! bRet)
{
        //Failed
}
```

2.3.1.4.3 Vehicle location Event links Opening Barrier

```
//Event callback
int CALLBACK    afMessCallBack(LONG lCommand, LLONG lLinID, char *pBuf, DWORD dwBufLen,
char *pchDVRIP, LONG nDVRPort, LDWORD dwUser)
{
        if(lCommand == DH_ALARM_TRAFFIC_VEHICLE_POSITION) //Vehicle location event
        {
            ALARM_TRAFFIC_VEHICLE_POSITION* pstAccessInfo =
                                        (ALARM_TRAFFIC_VEHICLE_POSITION*)pBuf;
            //Then control the barrier gate through the pstAccessInfo information.
        }
}
//Set event callback
CLIENT_SetDVRMessCallBack(afMessCallBack,0);
//Subscribe vehicle location event
CLIENT_StartListenEx(lLoginHandle);
//Stop subscribing vehicle location event
CLIENT_StopListen(lLoginHandle);
```

2.3.1.4.4 Traffic Junction Event links Opening Barrier

```
int main()
{
        …………
        //Subscribe traffic junction event
        LLONG    lAnalyerHandle    =    CLIENT_RealLoadPictureEx(lLoginHandle,    0,
(DWORD)EVENT_IVS_ALL, TRUE, AnalyzerDataCallBack, NULL, NULL);
        if(NULL == lAnalyerHandle)
        {
                printf("CLIENT_RealLoadPictureEx: failed! Error code %x.\n", CLIENT_GetLastError());
                return -1;
        }
        // Stop subscribing traffic junction event
        BOOL bRet = CLIENT_StopLoadPic(lAnalyerHandle);
        if(FALSE == bRet)
        {
                printf("CLIENT_StopLoadPic: failed! Error code %x.\n", CLIENT_GetLastError());
                return -3;
```

```
    }
    return 0;
}
// traffic junction callback
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    switch(dwAlarmType)
    {
        case EVENT_IVS_TRAFFICJUNCTION:
            {
                DEV_EVENT_TRAFFICJUNCTION_INFO* pInfo = (DEV_EVENT_TRAFFICJUNCTION_INFO*)pAlarmInfo;
                //Control the barrier gate according to the pInfo information.
               break;
            }
        default:
            break;
    }
    return 0;
}
```

2.3.2 Importing/Exporting Banned/Trusted List

2.3.2.1 Introduction

Importing or exporting the banned list or trusted list is applicable to quick configuration of the camera. You can use the imported list only when you have configured the camera.

Applicable device: IPMECK device

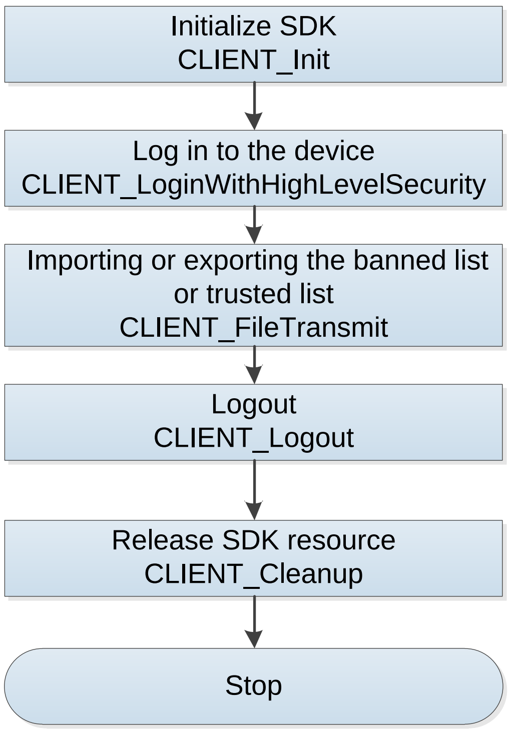
2.3.2.2 Interface Overview

Table 2-11 Importing/exporting the banned/trusted list interfaces

Interface	Description
CLIENT_FileTransmit	Import or export the banned list or trusted list.

2.3.2.3 Process

Figure 2-16 Importing/exporting the banned/trusted list



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_FileTransmit** to control importing or exporting the banned list or trusted list.
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

Keep the table head of the imported table consistent with the camera template; otherwise, the query will fail.

2.3.2.4 Example Code

```
//File transmission progress callback
void CALLBACK   bfTransFileCallBack(LLONG lHandle, int nTransType, int nState, int nSendSize, int
nTotalSize, LDWORD dwUser)
{
    if (nTransType == DH_DEV_BLACKWHITE_LOAD)
    {
        if (nState == 0)
```

```
            {
                //After calling stopLoadFileTransmit, export the banned list or the trusted list
            }
        }
        else if(nTransType == DH_DEV_BLACKWHITETRANS_SEND)
        {
            if (nState == 0)
            {
                //After calling stopSendFileTransmit, send the banned list or the trusted list
            }
        }
        //Display file transmission progress
}
//Stop exporting the banned list or the trusted list
Void stopLoadFileTransmit(LLONG lHandle )
{
LLONG nRet =
CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITE_LOAD_STOP,(char*)&lHandle,sizeof(
LLONG),NULL,NULL,5000);
}
// Stop sending the banned list or the trusted list
void CBWListDlg::stopSendFileTransmit(LLONG lHandle )
{
        LLONG                                   nRet                                    =
CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITETRANS_STOP,(char*)&lHandle,sizeof(
LLONG),NULL,NULL,5000);
}

int main()
{
        //Export the banned list or the trusted list
        DHDEV_LOAD_BLACKWHITE_LIST_INFO stulistinfo;
        CString strPath = "C:\\1\\3.CSV";
        strncpy(stulistinfo.szFile, strPath.GetBuffer(), sizeof(stulistinfo.szFile)-1);
        stulistinfo.byFileType = 1;
        LLONG nRet =
CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITE_LOAD,(char*)&stulistinfo,sizeof(DHD
EV_LOAD_BLACKWHITE_LIST_INFO),bfTransFileCallBack,(LDWORD)this,5000);
        if (nRet <= 0)
        {
```

```
        //Failed
    }

    //Send the banned list or the trusted list
    DHDEV_BLACKWHITE_LIST_INFO stulistinfo;
    CString strPath = "C:\\1\\3.CSV";
    strncpy(stulistinfo.szFile, strPath.GetBuffer(), sizeof(stulistinfo.szFile)-1);
    stulistinfo.byFileType = 1;
    stulistinfo.byAction = 0;
    LLONG nHandle = CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITETRANS_START,(char*)&stulistinfo,sizeof(DHDEV_BLACKWHITE_LIST_INFO),bfTransFileCallBack,(LDWORD)this,5000);
    if (nHandle > 0)
    {
        LLONG nRet = CLIENT_FileTransmit(m_lLoginHandle,DH_DEV_BLACKWHITETRANS_SEND,(char*)&nHandle,sizeof(LLONG),bfTransFileCallBack,(LDWORD)this,5000);
        if (nRet <= 0)
        {
            //Failed
        }
    }
    else
    {
        //Failed
    }
    return 0;
}
```

2.3.3 Voice talk

2.3.3.1 Introduction

Voice talk is used to realize the intercom between local platform and the scene where cameras installed. For example: In unattended solution, customers want to communicate the barrier abnormality with the center platform.

This section introduces how to realize voice talk between the platform and device through SDK.

2.3.3.2 Interface Overview

Table 2-12 Voice talk interfaces

Interface	**Description**
CLIENT_StartTalkEx	Extension interface of satrting vioce talk.
CLIENT_StopTalkEx	Extension interface of stopping vioce talk
CLIENT_RecordStartEx	Extension interface of satrting client sound reording (It is valid only when the device connects to Windows paltform).
CLIENT_RecordStopEx	Extension interface of stopping client sound reording. (It is valid only when the device connects to Windows paltform).
CLIENT_TalkSendData	Send sound reording data to devices.
CLIENT_AudioDecEx	Extension interface of decoding sound reording data (It is valid only when the device is working with Windows paltform).
CLIENT_SetDeviceMode	Set voice talk woring mode of the device.
CLIENT_SetDVRMessCallBack	Set the callback of ITC requesting the platform to start voice talk event.
CLIENT_StartListenEx	Subscribe ITC requesting the platform to start voice talk event.
CLIENT_StopListen	Stop subscribing ITC requesting the platform to start voice talk event.

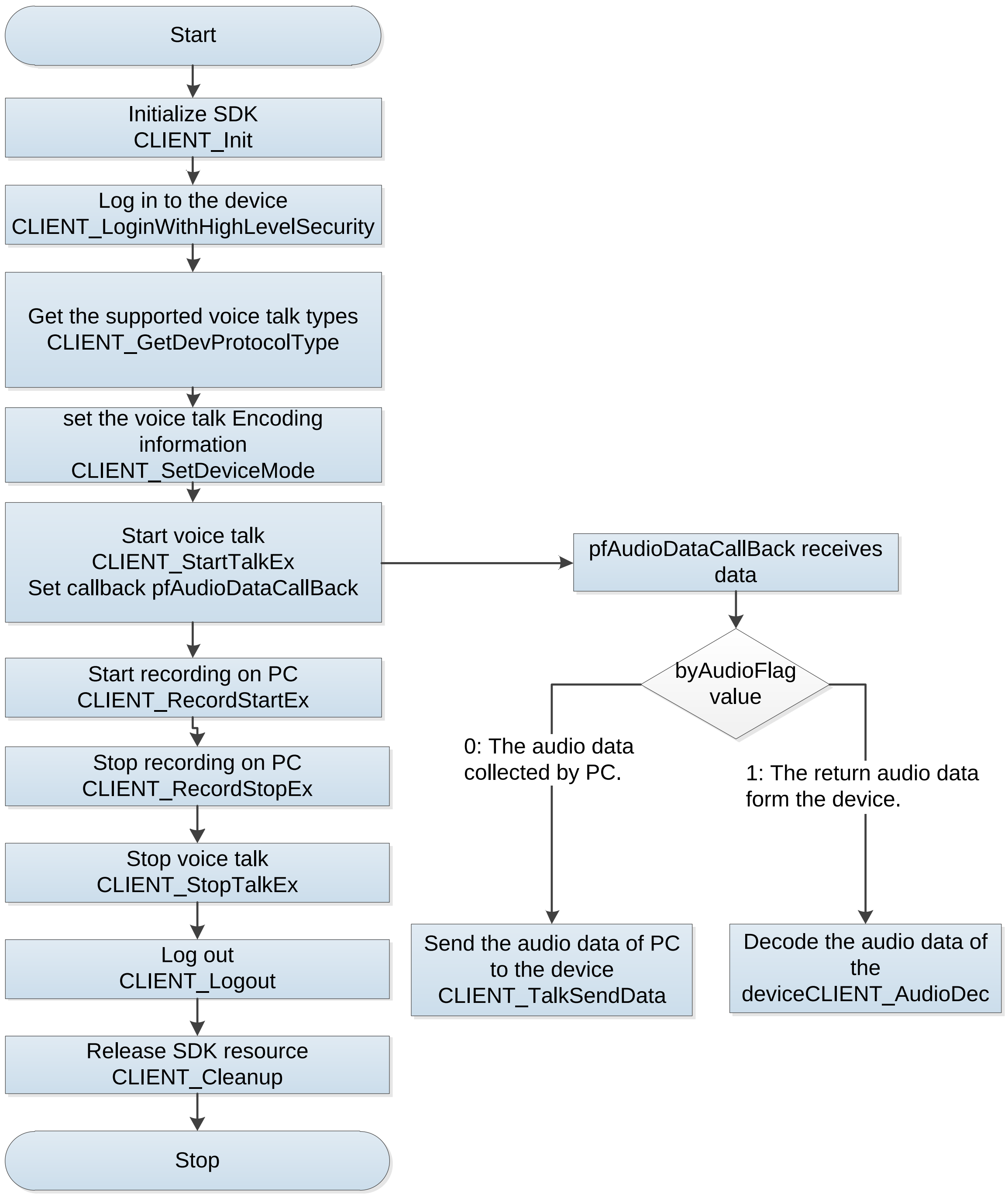
2.3.3.3 Process

2.3.3.3.1 Voice Talk Process

When SDK collects the audio data from local audio card, or SDK receives the audio data from the camera, it calls audio data callback. You can call SDK interface when calling the function to send the collected audio data to the camera, and also can call SDK interface to decode the received audio data from the camera.

- This model is valid only when working with Windows platform.
- There are voice talk (generation II) and voice talk (generation III) at present. You can call CLIENT_GetDevProtocolType to get the supported voice talk types of the device. Voice talk (generation II) and voice talk (generation III) have the same voice talk process, and different parameter configurations of CLIENT_SetDeviceMode.

Figure 2-17 voice talk (generation II)


Start
Initialize SDK
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Get the supported voice talk types
CLIENT_GetDevProtocolType
set the voice talk Encoding information
CLIENT_SetDeviceMode
Start voice talk
CLIENT_StartTalkEx
Set callback pfAudioDataCallBack
pfAudioDataCallBack receives data
byAudioFlag value
0: The audio data collected by PC.
1: The return audio data form the device.
Send the audio data of PC to the device
CLIENT_TalkSendData
Decode the audio data of the deviceCLIENT_AudioDec
Start recording on PC
CLIENT_RecordStartEx
Stop recording on PC
CLIENT_RecordStopEx
Stop voice talk
CLIENT_StopTalkEx
Log out
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
Stop

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_GetDevProtocolType** to get the supported voice talk types (generation II or generation III).

Step 4 Call **CLIENT_SetDeviceMode** to set the voice talk parameters.

If voice talk (generation II) is supported: Set coding mode, client mode, and speak mode. Set emType to be DH_TALK_ENCODE_TYPE, DH_TALK_CLIENT_MODE and DH_TALK_SPEAK_PARAM.

If voice talk (generation III) is supported: Set coding mode, client mode, and the parameters of voice talk (generation III). Set emType to be DH_TALK_ENCODE_TYPE, DH_TALK_CLIENT_MODE, and DH_TALK_MODE3.

Step 5 Call **CLIENT_StartTalkEx** to set callback and start voice talk. When call back function, call **CLIENT_AudioDec** to decode the audio data from the device; call **CLIENT_TalkSendData** to send audio data of PC to the device.

Step 6 Call **CLIENT_StartTalkEx** to start sound recording on PC. After calling the interface, voice talk callback of **CLIENT_StartTalkEx** will receive the local audio data.

Step 7 After using voice talk function, call **CLIENT_RecordStopEx** to stop PC sound recording.

Step 8 Call **CLIENT_StopTalkEx** to stop voice talk.

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

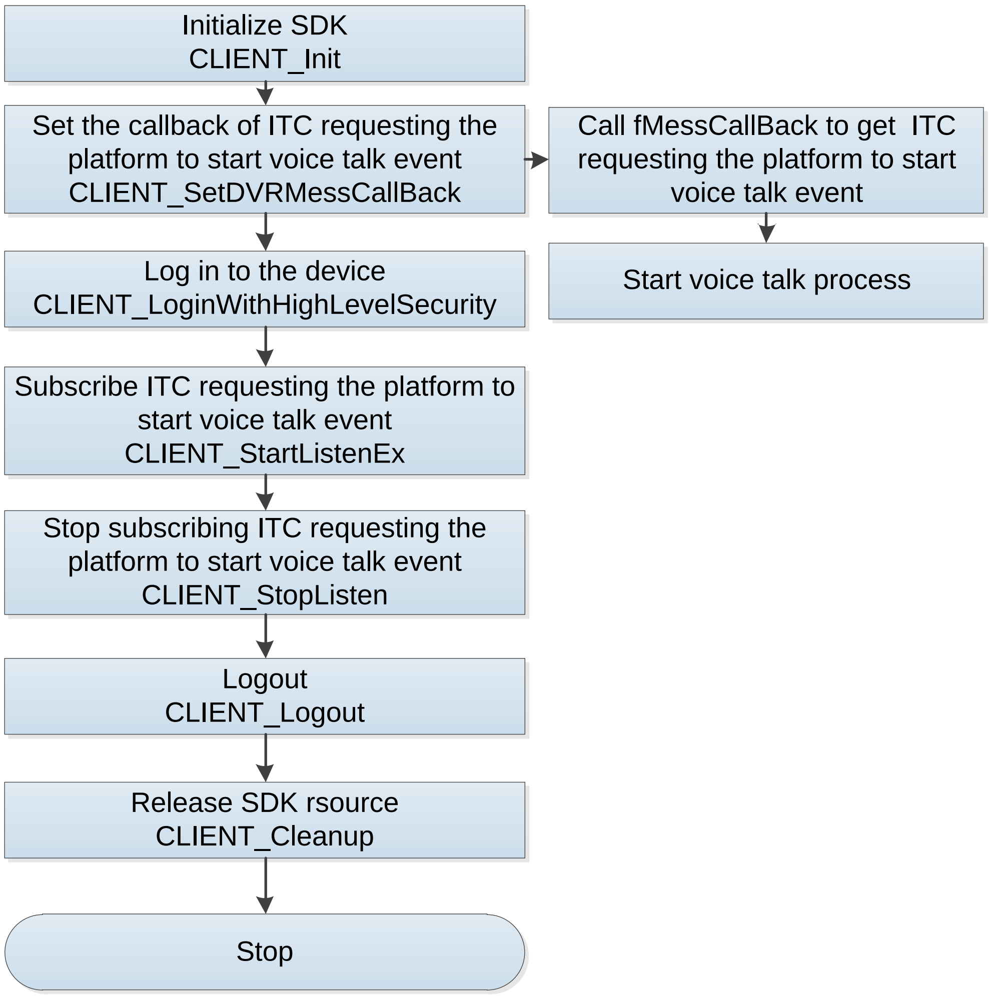
Step 10 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Audio encoding format: The example adopts the common format PCM, SDK supports getting the supported voice talk encoding format. For the source code, see the release package on the official website. If the default PCM can meet the user's demand, no need to get the supported voice talk encoding format.
- Device has no sound: Collect audio data from audio collection devices such as microphone. Check whether the device connects to an audio collection device, and whether CLIENT_RecordStartEx interface returns.

2.3.3.3.2 ITC Requesting the Platform to Start Voice Talk Event

Figure 2-18 ITC requesting the platform to start voice talk event


Initialize SDK
CLIENT_Init
Set the callback of ITC requesting the platform to start voice talk event
CLIENT_SetDVRMessCallBack
Call fMessCallBack to get ITC requesting the platform to start voice talk event
Start voice talk process
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Subscribe ITC requesting the platform to start voice talk event
CLIENT_StartListenEx
Stop subscribing ITC requesting the platform to start voice talk event
CLIENT_StopListen
Logout
CLIENT_Logout
Release SDK rsource
CLIENT_Cleanup
Stop

Process Description

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_SetDVRMessCallBack** to set alarm callback. When there is requesting voice talk event, call voice talk precess.
Step 3 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 4 Call **CLIENT_StartListenEx** to subscribe requesting voice talk event.
Step 5 Call **CLIENT_StopListen** to stop subscribing requesting voice talk event.
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.

2.3.3.4 Example Code

2.3.3.4.1 Voice Talk

```
//Get the supported voice talk type (generation II or generation III)
EM_DEV_PROTOCOL_TYPE emTpye = EM_DEV_PROTOCOL_UNKNOWN;
CLIENT_GetDevProtocolType(g_lLoginHandle, &emTpye);
```

```
DHDEV_TALKDECODE_INFO curTalkMode = {0};
curTalkMode.encodeType = DH_TALK_PCM;
curTalkMode.nAudioBit = 16;
curTalkMode.dwSampleRate = 8000;
curTalkMode.nPacketPeriod = 25;
CLIENT_SetDeviceMode(lLoginHandle, DH_TALK_ENCODE_TYPE, &curTalkMode); //Set voice talk
encoding format
CLIENT_SetDeviceMode(lLoginHandle, DH_TALK_CLIENT_MODE, NULL);//Set client voice talk

//Set parameters according to the supported voice talk type
if (emTpye == EM_DEV_PROTOCOL_V3) // Voice talk (generation III) requests this setting, and voice
talk (generation II) does not request this setting
{
    NET_TALK_EX stuTalk = {sizeof(stuTalk)};
        stuTalk.nAudioPort      = RECEIVER_AUDIO_PORT;   //Custom receiving port
    stuTalk.nChannel = 0;
        stuTalk.nWaitTime       = 5000;
        CLIENT_SetDeviceMode(m_lLoginHandle, DH_TALK_MODE3, &stuTalk)
}
//Start voice talk
lTalkHandle = CLIENT_StartTalkEx(lLoginHandle, AudioDataCallBack, (LDWORD)NULL);
//Start local sound recording
CLIENT_RecordStartEx(lLoginHandle);
//Stop local sound recording
CLIENT_RecordStopEx(lLoginHandle)
//Stop voice talk
CLIENT_StopTalkEx(lTalkHandle);
//Voice talk callback data processing
void CALLBACK AudioDataCallBack(LLONG lTalkHandle, char *pDataBuf, DWORD dwBufSize, BYTE
byAudioFlag, LDWORD dwUser)
{
if(0 == byAudioFlag)
    {
        //Send the audio card data detected by PC to the device
        CLIENT_TalkSendData(lTalkHandle, pDataBuf, dwBufSize);
    }
else if(1 == byAudioFlag)
    {
        //Send the audio data from the device to SDK for decoding play
        CLIENT_AudioDec(pDataBuf, dwBufSize);
    }
}
```

2.3.3.4.2 ITC Requesting the Platform to Start Voice Talk Event

```
// Call ITC requesting the platform to start voice talk event
int CALLBACK    afMessCallBack(LONG lCommand, LLONG lLinID, char *pBuf, DWORD dwBufLen,
char *pchDVRIP, LONG nDVRPort, LDWORD dwUser)
{
    if(lCommand == DH_ALARM_TALKING_INVITE) // ITC requesting the platform to start voice talk
event
    {
        //Callback voice talk process 2.3.3.4.1
    }
}
// Call ITC requesting the platform to start voice talk event
CLIENT_SetDVRMessCallBack(afMessCallBack,0);
// Subscribe ITC requesting the platform to start voice talk event
CLIENT_StartListenEx(lLoginHandle);
// Stop subscribing ITC requesting the platform to start voice talk event
CLIENT_StopListen(lLoginHandle);
```

2.3.4 Dot-matrix Display Character Control

2.3.4.1 Introduction

There are two statuses of dot-matrix display: Car pass status and normal status.

- Car pass status: When a car passes the access, the camera captures it, which triggers the event, and the car pass status will be activated, and it lasts a certain period (the period can be set). When the car is passing, the dot-matrix display displays plate number, parking card validity and custom data, and it broadcasts the displayed data automatically.
- Normal status: The status appears after car pass status, and dot-matrix display displays the available space information.

- When the mornal status display information is issues in car pass status, it cannot be displayed immediately. The information will be displayed after the display enters normal status.
- When the car pass status display information is issues in normal status, it cannot be displayed immediately. The information will be displayed after the display enters car pass status.

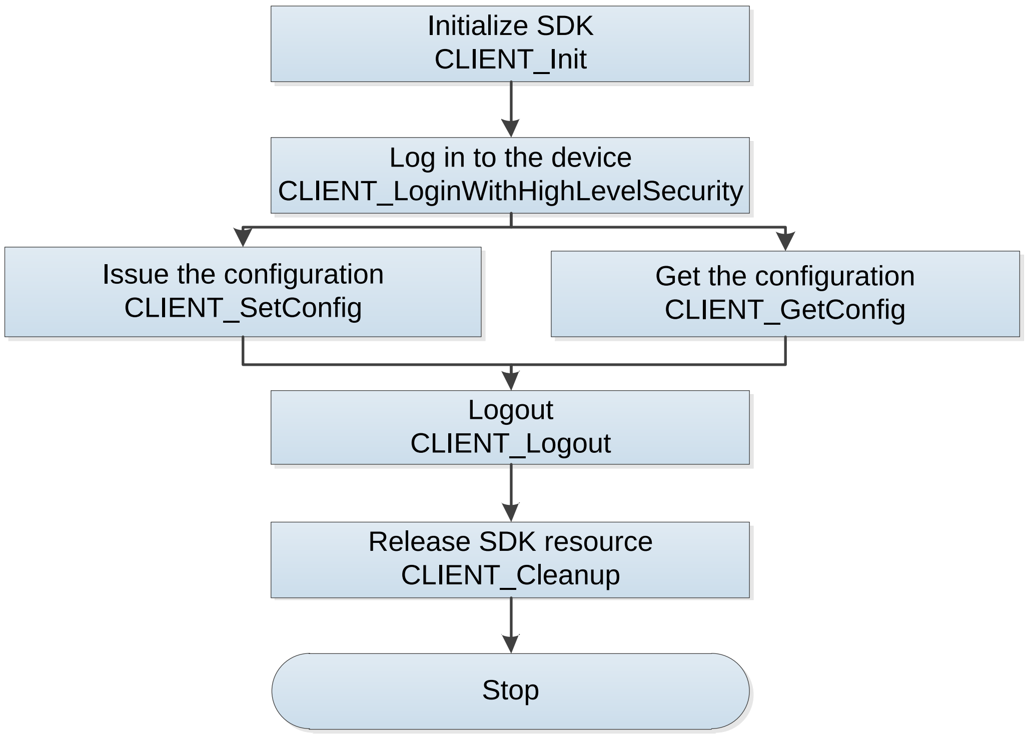
2.3.4.2 Interface Overview

Table 2-13 Dot-matrix display interfaces

Interface	Description
CLIENT_GetConfig	Get the LED Lattice screen display configuration
CLIENT_SetConfig	Set the LED Lattice screen display configuration

2.3.4.3 Process

Figure 2-19 Dot-matrix display character control



Process Description

Setting

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_SetConfig** to set the LED Lattice screen display configuration.
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.

Getting

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_GetConfig** to get the LED Lattice screen display configuration.
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.

2.3.4.4 Example Code

```
//Get the LED Lattice screen display configuration
NET_CFG_TRAFFIC_LATTICE_SCREEN_INFO m_stuTrafficscreenInfo= {sizeof(m_
stuTrafficscreenInfo)};
BOOL bRet = CLIENT_GetConfig(m_LoginID,
NET_EM_CFG_TRAFFIC_LATTICE_SCREEN,m_nChannel,&m_ stuTrafficscreenInfo,sizeof(m_
stuTrafficscreenInfo), 5000);
if (！ bRet)
{
    //Failed
}

//Set the LED Lattice screen display configuration
NET_CFG_TRAFFIC_LATTICE_SCREEN_INFO m_ stuTrafficscreenInfo= {sizeof(m_
stuTrafficscreenInfo)};
……
BOOL bRet = CLIENT_SetConfig(m_LoginID,
NET_EM_CFG_TRAFFIC_LATTICE_SCREEN,m_nChannel,&m_ stuTrafficscreenInfo,sizeof(m_
stuTrafficscreenInfo), 5000);
if (！ bRet)
{
    //Failed
}
```

2.3.5 Parking Space Indicator Configuration

2.3.5.1 Introduction

Get the supervision status of the indicator group.

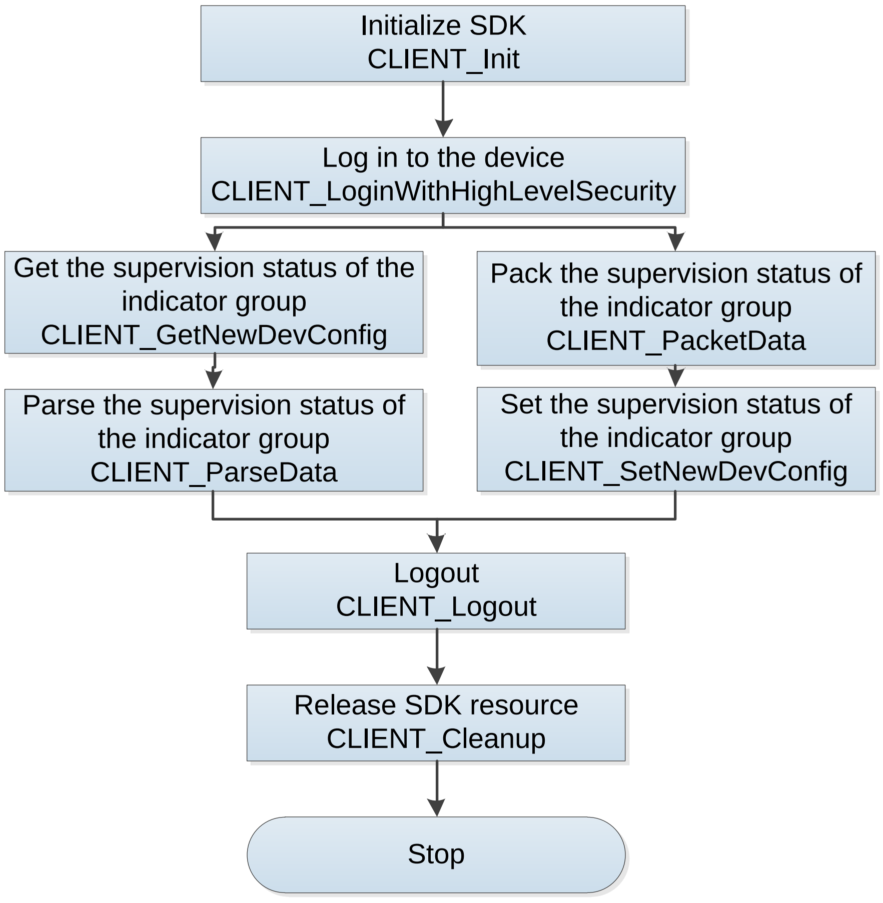
2.3.5.2 Interface Overview

Table 2-14 Parking space indicator configuration interfaces

Interface	Description
CLIENT_SetNewDevConfig	Set the supervision status of the indicator group
CLIENT_GetNewDevConfig	Get the supervision status of the indicator group
CLIENT_ParseData	Parse the supervision status of the indicator group
CLIENT_PacketData	Pack the supervision status of the indicator group

2.3.5.3 Process

Figure 2-20 Parking space indicator configuration


Initialize SDK
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Get the supervision status of the indicator group
CLIENT_GetNewDevConfig
Parse the supervision status of the indicator group
CLIENT_ParseData
Pack the supervision status of the indicator group
CLIENT_PacketData
Set the supervision status of the indicator group
CLIENT_SetNewDevConfig
Logout
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
Stop

Process Description

Getting

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_GetNewDevConfig** to get the parking space indicator configuration.
Step 4 Call **CLIENT_ParseData** to parse the parking space indicator light configuration.
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.

Setting

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call CLIENT_LoginWithHighLevelSecurity to log in to the device.
Step 3 Call **CLIENT_PackatData** to pack the parking space indicator configuration.
Step 4 Call **CLIENT_SetConfig** to set the parking space indicator light configuration.
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.

2.3.5.4 Example Code

```
//Set parking space indicator configuration
```

```
CFG_PARKING_SPACE_LIGHT_GROUP_INFO_ALL stuInfo = {0};
stuInfo. nCfgNum= m_ nCfgNum;
for (int i = 0;i<m_ nCfgNum;i++)
{
    stuInfo.stuLightGroupInfo.bEnable = TRUE;
    ......
}
BOOL bRet = CLIENT_PacketData(CFG_CMD_PARKING_SPACE_LIGHT_GROUP,(LPVOID)&stuInfo,
sizeof(stuInfo), szJsonBuf, sizeof(szJsonBuf));
if (bRet)
{
    int nerror = 0;
    int nrestart = 0;
    int nChannelID = -1;
    bRet  =  CLIENT_SetNewDevConfig(m_iLoginID,  CFG_CMD_PARKING_SPACE_LIGHT_GROUP,
nChannelID, szJsonBuf, 512*40, &nerror, &nrestart, 3000);
}
//Get parking space indicator configuration
char szJsonBuf[1024 * 40] = {0};
int nerror = 0;
int nChannel = -1;
BOOL ret = CLIENT_GetNewDevConfig(m_iLoginID,
CFG_CMD_PARKING_SPACE_LIGHT_GROUP,nChannel,szJsonBuf,1024*40,&nerror,3000);
if (0 != ret)
{
    CFG_PARKING_SPACE_LIGHT_GROUP_INFO_ALL stuInfo = {0};
    DWORD dwRetLen = 0;
    ret                                                                    =
CLIENT_ParseData(CFG_CMD_PARKING_SPACE_LIGHT_GROUP,szJsonBuf,(char*)&stuInfo,sizeof(
stuInfo),&dwRetLen);
    if (!ret)
    {
        //Failed
        return ;
    }
}
else
{
    //Failed
    return ;
}
```

2.3.6 Parking Space Status Indicator Configuration

2.3.6.1 Introduction

Configure parking space status indicator.

- Set getting the indicator color of parking space free status.
- Set getting the indicator color of parking space full status.
- Set getting the indicator color of single network port exception.
- Set getting the indicator color of dual network port exception.

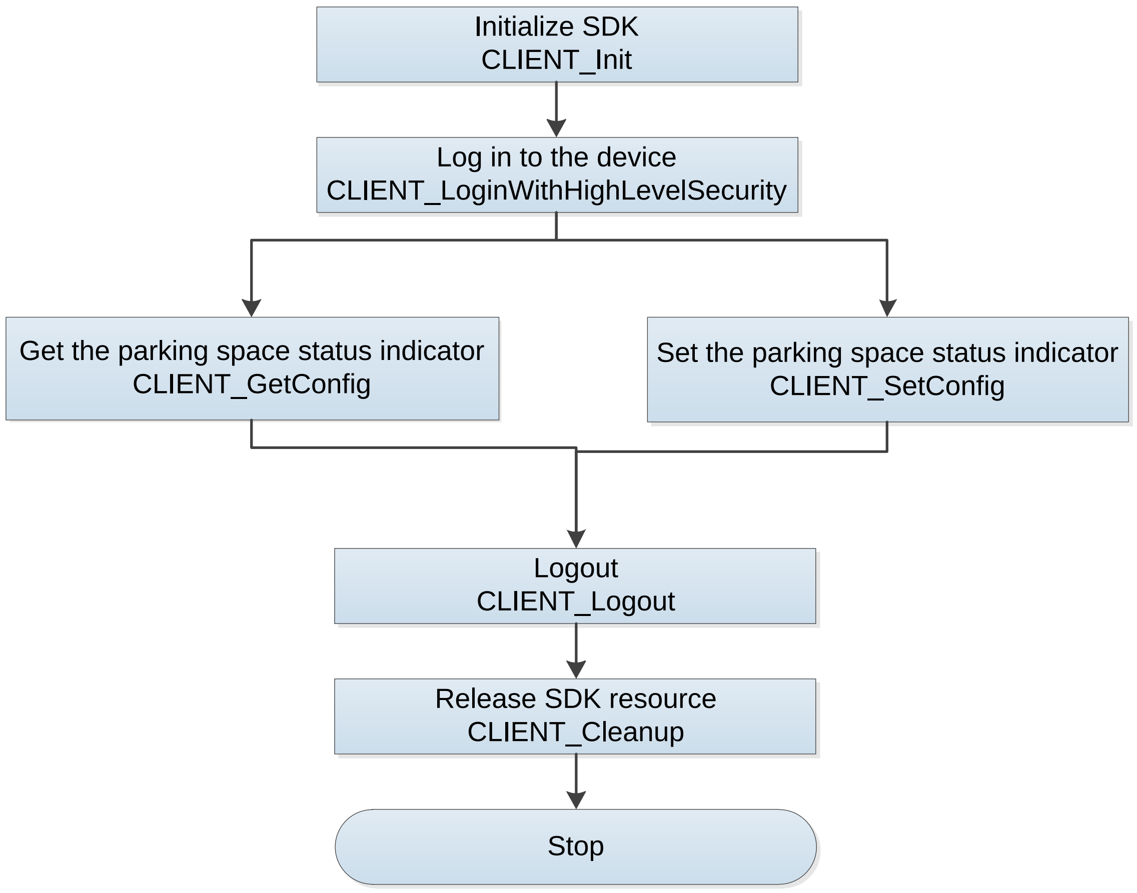
2.3.6.2 Interface Overview

Table 2-15 Parking space status indicator configuration interfaces

Interface	Description
CLIENT_SetConfig	Set the parking space status indicator
CLIENT_GetConfig	Get the parking space status indicator

2.3.6.3 Process

Figure 2-21 Parking space status indicator configuration



Process Description

Getting

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_GetConfig** to get the parking space status indicator configuration.
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.

Setting

Step 1 Call **CLIENT_Init** to initialize SDK.
Step 2 Call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.
Step 3 Call **CLIENT_SetConfig** to set the parking space status indicator configuration.
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.

2.3.6.4 Example Code

```
//Set the parking space status indicator configuration
NET_PARKINGSPACELIGHT_STATE_INFO stuInfo;
memset(&stuInfo, 0, sizeof(stuInfo));
stuInfo.dwSize = sizeof(stuInfo);
stuInfo.stuSpaceFreeInfo.nRed = 1;          //Set the status indicator to be red normally on for the free parking space
BOOL bRet = CLIENT_SetConfig(m_lLoginID, NET_EM_CFG_PARKINGSPACELIGHT_STATE, -1, &stuInfo, sizeof(stuInfo));
if (bRet == FALSE)
{
    //Failed
    return;
}
// Get the parking space status indicator configuration
NET_PARKINGSPACELIGHT_STATE_INFO stuInfo;
memset(&stuInfo, 0, sizeof(stuInfo));
stuInfo.dwSize = sizeof(stuInfo);
BOOL bRet = CLIENT_GetConfig(m_lLoginID, NET_EM_CFG_PARKINGSPACELIGHT_STATE, -1, &stuInfo, sizeof(stuInfo));
if (bRet == FALSE)
{
    //Failed
    return;
}
```

2.3.7 Parking Space Detector Light Plan

2.3.7.1 Introduction

Set parking space detector light plan.

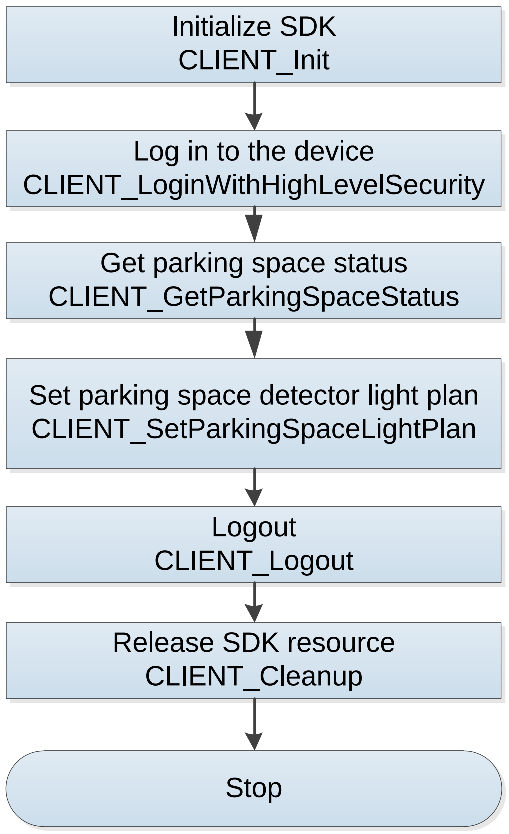
2.3.7.2 Interface Overview

Table 2-16 Parking space detector light schedule interfaces

Interface	Description
CLIENT_GetParkingSpaceStatus	Get parking space status
CLIENT_SetParkingSpaceLightPlan	Set parking space detector light plan

2.3.7.3 Process

Figure 2-22 Parking space detector plan configuration


Initialize SDK
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Get parking space status
CLIENT_GetParkingSpaceStatus
Set parking space detector light plan
CLIENT_SetParkingSpaceLightPlan
Logout
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
Stop

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_GetParkingSpaceStatus** to get the parking space status.
Step 4 Call **CLIENT_SetParkingSpaceLightPlan** to set parking space detector light plan.
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.

2.3.7.4 Example Code

```
//Get the parking space status
NET_IN_GET_PARKINGSPACE_STATUS stuInStatus = {sizeof(stuInStatus)};
stuInStatus.dwWaitTime = 5000;
stuInStatus.nChannelID = 0;
int nLandID[255];
nLandID [0] = 255;
stuInStatus.pLaneID = dd;
stuInStatus.nLaneCount =1;
NET_OUT_GET_PARKINGSPACE_STATUS stuOutStatus = {sizeof(stuOutStatus)};
stuOutStatus.nMaxStatus = 100;
char* ddg = new char[sizeof(NET_LANE_PARKINGSPACE_STATUS)*stuOutStatus.nMaxStatus];
stuOutStatus.pStatus = (NET_LANE_PARKINGSPACE_STATUS *)ddg;
BOOL bret = CLIENT_GetParkingSpaceStatus(m_lLoginID,   &stuInStatus,   &stuOutStatus);
// Set parking space detector light plan
NET_IN_SET_PARKING_SPACE_LIGHT_PLAN m_stuinInfo = sizeof(m_stuinInfo);

int nLightPlanNum = 2;
NET_PARKING_SPACE_LIGHT_PLAN_INFO *pInfo = new
NET_PARKING_SPACE_LIGHT_PLAN_INFO[nLightPlanNum];
m_stuinInfo.nPhysicalLane = stuOutStatus.pStatus[0].nLaneID;
m_stuinInfo.nLightPlanNum = nLightPlanNum;
m_stuinInfo.pstuLightPlan = pInfo;

NET_OUT_SET_PARKING_SPACE_LIGHT_PLAN m_stuoutInfo= sizeof(m_ stuoutInfo);
BOOL bRet = CLIENT_SetParkingSpaceLightPlan (m_LoginID, &m_stuinInfo, &m_stuoutInfo, 5000);
if (！ bRet)
{
    //Failed
    delete[] ddg;
    delete[] pInfo;
}
```

3 Interface Definition

3.1 General

3.1.1 SDK Initialization

3.1.1.1 SDK CLIENT_Init

Table 3-1 Initialize SDK

Item	Description	
Name	Initialize SDK.	

Function	BOOL CLIENT_Init(
	    fDisConnect cbDisConnect,
	    LDWORD dwUser
	);	

Parameter	[in]cbDisConnect	Disconnection callback.
	[in]dwUser	User parameter of disconnection callback.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	● The precondition for calling other function modules of SDK. ● The callback will not send to the user after the device is disconnected if the callback is set as NULL.	

3.1.1.2 CLIENT_Cleanup

Table 3-2 Clean up SDK

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

3.1.1.3 CLIENT_SetAutoReconnect

Table 3-3 Set reconnection callback

Item	Description
Name	Set auto reconnection callback.

Item	Description	

Function	void CLIENT_SetAutoReconnect(
	    fHaveReConnect cbAutoConnect,
	    LDWORD 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.	

3.1.1.4 CLIENT_SetNetworkParam

Table 3-4 Set network parameter

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

3.1.2 Device Initialization

3.1.2.1 CLIENT_StartSearchDevicesEx

Table 3-5 Search for device

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

3.1.2.2 CLIENT_InitDevAccount

Table 3-6 Initialize device

Item	Description	
Name	Initialize the device.	

Function	BOOL CLIENT_InitDevAccount(
	    const NET_IN_INIT_DEVICE_ACCOUNT *pInitAccountIn,
	    NET_OUT_INIT_DEVICE_ACCOUNT *pInitAccountOut,
	    DWORD dwWaitTime,
	    char *szLocalIp
	);	

Parameter	[in]pInitAccountIn	Corresponds to structure of NET_IN_INIT_DEVICE_ACCOUNT.
	[out]pInitAccountOut	Corresponds to structure of NET_OUT_INIT_DEVICE_ACCOUNT.
	[in]dwWaitTime	Timeout.
	[in]szLocalIp	● In case of single network card, the last parameter is not required to be filled. ● In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.2.3 CLIENT_GetDescriptionForResetPwd

Table 3-7 Get information for password reset

Name	Description	
Name	Get information for password reset.	

Function	BOOL CLIENT_GetDescriptionForResetPwd(
	    const NET_IN_DESCRIPTION_FOR_RESET_PWD *pDescriptionIn,
	    NET_OUT_DESCRIPTION_FOR_RESET_PWD *pDescriptionOut,
	    DWORD dwWaitTime,
	    char *szLocalIp
	);	

Parameter	[in]pDescriptionIn	Corresponds to structure of NET_IN_DESCRIPTION_FOR_RESET_PWD.
	[out]pDescriptionOut	Corresponds to structure of NET_OUT_DESCRIPTION_FOR_RESET_PWD.
	[in]dwWaitTime	Timeout.
	[in]szLocalIp	● In case of single network card, the last parameter is not required to be filled. ● In case of multiple network card, enter the IP of the host PC for the last parameter.

Name	Description
Return value	● Success: TRUE. ● Failure: FALSE.
Note	None.

3.1.2.4 CLIENT_CheckAuthCode

Table 3-8 Check the validity of security code

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

3.1.2.5 CLIENT_ResetPwd

Table 3-9 Reset the password

Item	Description	
Name	Reset the password.	
Function	BOOL CLIENT_ResetPwd(const NET_IN_RESET_PWD *pResetPwdIn, NET_OUT_RESET_PWD *pResetPwdOut, DWORD dwWaitTime, char *szLocalIp);	
Parameter	[in]pResetPwdIn	Corresponds to structure of NET_IN_RESET_PWD.
	[out]pResetPwdOut	Corresponds to structure of NET_OUT_RESET_PWD.
	[in]dwWaitTime	Timeout.

Item	Description	
	[in]szLocalIp	● In case of single network card, the last parameter is not required to be filled. ● In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.2.6 CLIENT_GetPwdSpecification

Table 3-10 Get password rules

Item	Description	
Name	Get password rules.	

Function	BOOL CLIENT_GetPwdSpecification(
	    const NET_IN_PWD_SPECI *pPwdSpeciIn,
	    NET_OUT_PWD_SPECI *pPwdSpeciOut,
	    DWORD dwWaitTime,
	    char *szLocalIp
	);	

Parameter	[in]pPwdSpeciIn	Corresponds to structure of NET_IN_PWD_SPECI.
	[out]pPwdSpeciOut	Corresponds to structure of NET_OUT_PWD_SPECI.
	[in]dwWaitTime	Timeout.
	[in]szLocalIp	● In case of single network card, the last parameter is not required to be filled. ● In case of multiple network card, enter the IP of the host PC for the last parameter.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.2.7 CLIENT_StopSearchDevices

Table 3-11 Stop searching device

Item	Description	
Name	Stop searching.	

Function	BOOL CLIENT_StopSearchDevices (
	    LLONG lSearchHandle
	);	

Parameter	[in] lSearchHandle	Searching handle.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	Multi-thread calling is not supported.	

3.1.3 Device Login

3.1.3.1 CLIENT_LoginWithHighLevelSecurity

Table 3-12 Log in with high level security

Item	Description	
Name	Login the device with high level security.	
Function	LLONG CLIENT_LoginWithHighLevelSecurity (NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY* pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY* pstOutParam);	
Parameter	[in] pstInParam	[in] dwSize
		[in] szIP
		[in] nPort
		[in] szUserName
		[in] szPassword
		[in] emSpecCap
		[in] pCapParam
	[out] pstOutParam	[in]dwSize
		[out] stuDeviceInfo
		[out] nError
Return value	● Success: Not 0. ● Failure: 0.	
Note	Login the device with high level security. CLIENT_LoginEx2 can still be used,but there are security risks,so it is highly recommended to use the latest interface CLIENT_LoginWithHighLevelSecurity to log in to the device.	

Table 3-13 Error code and meaning

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

3.1.3.2 CLIENT_Logout

Table 3-14 Log out

Item	Description	
Name	Logout the device.	

Function	BOOL CLIENT_Logout(
	    LLONG lLoginID
	);	

Parameter	[in]lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.4 Real-time Monitoring

3.1.4.1 CLIENT_RealPlayEx

Table 3-15 Start the real-time monitoring

Item	Description	
Name	Open the real-time monitoring.	

Function	LLONG CLIENT_RealPlayEx(
	    LLONG lLoginID,
	    int nChannelID,
	    HWND hWnd,
	    DH_RealPlayType rType
	);	

Parameter	[in]lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity
	[in]nChannelID	Video channel number is a round number starting from 0.
	[in]hWnd	Window handle valid only under Windows system.
	[in]rType	Preview type.
Return value	● Success: Not 0. ● Failure: 0.	
Note	Windows system: ● 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 treatment.	

Table 3-16 Live view type and meaning

Preview type	Meaning
DH_RType_Realplay	Real-time preview.
DH_RType_Multiplay	Multi-picture preview.

Preview type	Meaning
DH_RType_Realplay_0	Real-time monitoring—main stream, equivalent to DH_RType_Realplay.
DH_RType_Realplay_1	Real-time monitoring—sub stream 1.
DH_RType_Realplay_2	Real-time monitoring—sub stream 2.
DH_RType_Realplay_3	Real-time monitoring—sub stream 3.
DH_RType_Multiplay_1	Multi-picture preview—1 picture.
DH_RType_Multiplay_4	Multi-picture preview—4 pictures.
DH_RType_Multiplay_8	Multi-picture preview—8 pictures.
DH_RType_Multiplay_9	Multi-picture preview—9 pictures.
DH_RType_Multiplay_16	Multi-picture preview—16 pictures.
DH_RType_Multiplay_6	Multi-picture preview—6 pictures.
DH_RType_Multiplay_12	Multi-picture preview—12 pictures.
DH_RType_Multiplay_25	Multi-picture preview—25 pictures.
DH_RType_Multiplay_36	Multi-picture preview—36 pictures.

3.1.4.2 CLIENT_StopRealPlayEx

Table 3-17 Stop the real-time monitoring

Item	Description	
Name	Stop the real-time monitoring.	

Function	BOOL CLIENT_StopRealPlayEx(
	    LLONG lRealHandle
	);	

Parameter	[in]lRealHandle	Return value of CLIENT_RealPlayEx.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.4.3 CLIENT_SaveRealData

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

Item	Description	
Name	Save the real-time monitoring data as file.	

Function	BOOL CLIENT_SaveRealData(
	    LLONG lRealHandle,
	    const char *pchFileName
	);	

Parameter	[in] lRealHandle	Return value of CLIENT_RealPlayEx.
	[in] pchFileName	Save path.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.4.4 CLIENT_StopSaveRealData

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

Item	Description	
Name	Stop saving the real-time monitoring data as file.	

Function	BOOL CLIENT_StopSaveRealData(
	    LLONG lRealHandle
	);	

Parameter	[in] lRealHandle	Return value of CLIENT_RealPlayEx.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.1.4.5 CLIENT_SetRealDataCallBackEx2

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

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

Function	BOOL CLIENT_SetRealDataCallBackEx2(
	    LLONG lRealHandle,
	    fRealDataCallBackEx2 cbRealData,
	    LDWORD dwUser,
	    DWORD dwFlag
	);	

Parameter	[in] lRealHandle	Return value of CLIENT_RealPlayEx.
	[in] cbRealData	Callback of monitoring data flow.
	[in] dwUser	Parameter of callback for monitoring data flow.
	[in] dwFlag	Type of monitoring data in callback. The type is EM_REALDATA_FLAG and supports OR operation.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

Table 3-21 dwFlag type and parameter

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

3.2 Traffic Junction

3.2.1 Download of Medial File

3.2.1.1 CLIENT_FindFileEx

Table 3-22 Query the media file per query condition

Item	Description	
Name	Query the media file per query condition.	
Function	LLONG CLIENT_FindFileEx(LLONG lLoginID, EM_FILE_QUERY_TYPE emType, void* pQueryCondition, void* reserved, int waittime);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emType	Query information type of media file, see **错误!未找到引用源。**.
	[in] pQueryCondition	Query condition.
	[in]reserved	Reserved parameter, not valid.
	[in] waittime	Timeout.
Return value	● Success: Not 0. ● Failure: 0.	
Note	When querying the media file, use DH_FILE_QUERY_TRAFFICCAR_EX for parameter emType. The parameter pQueryCondition corresponds to structure MEDIA_QUERY_TRAFFICCAR_PARAM_EX.	

Table 3-23 emType and meaning

emType enumeration definition	Meaning	Corresponding structure of pQueryCondition
DH_FILE_QUERY_TRAFFICCAR	Traffic vehicles information	MEDIA_QUERY_TRAFFICCAR_PARAM
DH_FILE_QUERY_FACE	Face information	MEDIAFILE_FACERECOGNITION_PARAM
DH_FILE_QUERY_FILE	File information	NET_IN_MEDIA_QUERY_FILE
DH_FILE_QUERY_TRAFFICCAR_EX	Traffic vehicles information (extension)	MEDIA_QUERY_TRAFFICCAR_PARAM_EX
DH_FILE_QUERY_FACE_DETECTION	Face detection information	MEDIAFILE_FACE_DETECTION_PARAM

3.2.1.2 CLIENT_GetTotalFileCount

Table 3-24 Get the total number of queried files

Item	Description	
Name	Get the total number of queried files.	
Function	BOOL CLIENT_GetTotalFileCount(LLONG lFindHandle, int* pTotalCount, void* reserved, int waittime);	
Parameter	[in] lFindHandle	Return value of CLIENT_FindFileEx.
	[out] pTotalCount	The total number of queried information.
	[in]reserved	Reserved parameter, not valid.
	[in] waittime	Timeout.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.2.1.3 CLIENT_FindNextFileEx

Table 3-25 Query the media file

Item	Description	
Name	Query the media file.	
Function	int CLIENT_FindNextFileEx(LLONG lFindHandle, int nFilecount, void* pMediaFileInfo, int maxlen, void* reserved, int waittime);	
Parameter	[in] lFindHandle	Return value of CLIENT_FindFileEx.
	[in] nFilecount	Query number.
	[out] pMediaFileInfo	Output cache of media file information.
	[in] maxlen	Value of maximum cache area.
	[in]reserved	Reserved parameter, not valid.
	[in] waittime	Timeout.
Return value	Returns the total number of queried media files. The query is called finished if the return value is smaller than the query number.	
Note	None.	

3.2.1.4 CLIENT_FindCloseEx

Table 3-26 Stop querying the media file

Item	**Description**	
Name	Stop querying the media file.	

Function	BOOL CLIENT_FindCloseEx(
	　　LLONG　lFindHandle
	);	

Parameter	[in] lFindHandle	Return value of CLIENT_FindFileEx.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.2.1.5 CLIENT_DownloadMediaFile

Table 3-27 Download the media file

Item	**Description**	
Name	Download the media file.	

Function	LLONG　CLIENT_DownloadMediaFile(
	　　LLONG　　lLoginID,
	　　EM_FILE_QUERY_TYPE　　emType,
	　　void*　　lpMediaFileInfo,
	　　char*　　sSavedFileName,
	　　fDownLoadPosCallBack　　cbDownLoadPos,
	　　LDWORD　　dwUserData,
	　　void*　　reserved
	);	

Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emType	Media file type, see 错误!未找到引用源。.
	[in] lpMediaFileInfo	Media file information.
	[in] sSavedFileName	Save path.
	[in] cbDownLoadPos	Callback of download progress: fDownLoadPosCallBack.
	[in] dwUserData	Corresponding user number of callback.
	[in]reserved	Reserved parameter, not valid.
Return value	● Success: Not 0. ● Failure: 0.	
Note	When downloading vehicles pictures, the parameter emType only supports DH_FILE_QUERY_TRAFFICCAR.	

3.2.1.6 CLIENT_StopDownloadMediaFile

Table 3-28 Stop downloading the media file

Item	**Description**
Name	Stop downloading the media file.

Item	Description	
Function	BOOL CLIENT_StopDownloadMediaFile(LLONG lFileHandle);	
Parameter	[in] lFindHandle	Return value of CLIENT_DownloadMediaFile.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.2.2 Manual Capture

3.2.2.1 CLIENT_RealLoadPictureEx

Table 3-29 Subscribe intelligent event

Item	Description	
Name	Subscribe intelligent event.	
Function	LLONG CLIENT_RealLoadPictureEx(LLONG lLoginID, int nChannelID, DWORD dwAlarmType, BOOL bNeedPicFile, fAnalyzerDataCallBack cbAnalyzerData, LDWORD dwUser, void* Reserved);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nChannelID	Device channel number.
	[in] dwAlarmType	Type of intelligent traffic event, see 错误!未找到引用源。and 错误!未找到引用源。.
	[in] bNeedPicFile	Whether picture is needed.
	[in] cbAnalyzerData	Callback of intelligent event: fAnalyzerDataCallBack.
	[in] dwUser	Corresponding user data of callback.
	[in]Reserved	Reserved parameter, not valid.
Return value	● Success: Not 0. ● Failure: 0.	
Note	● Call this interface in advance for manual capturing to receive the captured pictures. ● Call this interface in advance for event upload to receive the event information and pictures.	

Table 3-30 dwAlarmType and meaning

dwAlarmType macro definition	Value of macro definition	Meaning	Call the corresponding structure of pAlarmInfo
EVENT_IVS_TRAFFIC_MA	0x00000118	Intelligent	DEV_EVENT_TRAFFIC_MANU

dwAlarmType macro definition	Value of macro definition	Meaning	Call the corresponding structure of pAlarmInfo
NUALSNAP		capturing event	ALSNAP_INFO

3.2.2.2 CLIENT_ControlDeviceEx

Table 3-31 Control device.

Item	Description	
Name	Control device.	
Function	BOOL CLIENT_ControlDeviceEx(LLONG lLoginID, CtrlType emType, void* pInBuf, void* pOutBuf, int nWaitTime);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emType	Control type, see 错误!未找到引用源。and Table 3-32.
	[in] pInBuf	Control input cache, see 错误!未找到引用源。 and Table 3-32.
	[in] pOutBuf	Controls output cache.
	[in] nWaitTime	Timeout.
Return value	● Success: TRUE ● Failure: FALSE	
Note	Manually trigger the capturing and receive pictures through subscribing the callback of interface.	

The following table shows information about parameter emType:

Table 3-32 emType and meaning (2)

emType enumeration definition	Meaning	The corresponding structure of pInBuf
DH_MANUAL_SNAP	Manual capture	MANUAL_SNAP_PARAMETER

3.2.2.3 CLIENT_StopLoadPic

Table 3-33 Cancel subscription of intelligent event

Item	Description	
Name	Cancel subscription of intelligent event.	
Function	BOOL CLIENT_StopLoadPic(LLONG lAnalyzerHandle);	
Parameter	[in] lAnalyzerHandle	Return value of CLIENT_RealLoadPictureEx.
Return value	● Success: TRUE. ● Failure: FALSE.	

Item	Description
Note	After calling this interface, you will not receive the pictures even if continue to trigger manual capturing.

3.2.3 Upload of Intelligent Traffic Event

3.2.3.1 CLIENT_RealLoadPictureEx

For the interface function, see "3.2.2.1 CLIENT_RealLoadPictureEx."

Table 3-34 Type of intelligent traffic event

dwAlarmType macro definition	Value of macro definition	Meaning	Corresponding structure of pAlarmInfo
EVENT_IVS_ALL	0x00000001	All events	No
EVENT_IVS_TRAFFICCONTROL	0x00000015	Event of traffic control	DEV_EVENT_TRAFFICCONTROL_INFO
EVENT_IVS_TRAFFICACCIDENT	0x00000016	Event of traffic accident	DEV_EVENT_TRAFFICACCIDENT_INFO
EVENT_IVS_TRAFFICJUNCTION	0x00000017	Event of traffic conjunction	DEV_EVENT_TRAFFICJUNCTION_INFO
EVENT_IVS_TRAFFICGATE	0x00000018	Event of traffic gate	DEV_EVENT_TRAFFICGATE_INFO
EVENT_IVS_TRAFFIC_RUNREDLIGHT	0x00000100	Event of running the red light	DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO
EVENT_IVS_TRAFFIC_OVERLINE	0x00000101	Event of running over line	DEV_EVENT_TRAFFIC_OVERLINE_INFO
EVENT_IVS_TRAFFIC_RETROGRADE	0x00000102	Event of retrograde	DEV_EVENT_TRAFFIC_RETROGRADE_INFO
EVENT_IVS_TRAFFIC_TURNLEFT	0x00000103	Event of violating regulations by left turn	DEV_EVENT_TRAFFIC_TURNLEFT_INFO
EVENT_IVS_TRAFFIC_TURNRIGHT	0x00000104	Event of violating regulations by right turn	DEV_EVENT_TRAFFIC_TURNRIGHT_INFO
EVENT_IVS_TRAFFIC_UTURN	0x00000105	Event of violating regulations by turning around	DEV_EVENT_TRAFFIC_UTURN_INFO
EVENT_IVS_TRAFFIC_OVERSPEED	0x00000106	Event of running over speed	DEV_EVENT_TRAFFIC_OVERSPEED_INFO
EVENT_IVS_TRAFFIC_UNDERSPEED	0x00000107	Event of running under speed	DEV_EVENT_TRAFFIC_UNDERSPEED_INFO

dwAlarmType macro definition	**Value of macro definition**	**Meaning**	**Corresponding structure of pAlarmInfo**
EVENT_IVS_TRAFFIC_PARKING	0x00000108	Event of illegal parking	DEV_EVENT_TRAFFIC_PARKING_INFO
EVENT_IVS_TRAFFIC_WRONGROUTE	0x00000109	Event of running along the wrong route	DEV_EVENT_TRAFFIC_WRONGROUTE_INFO
EVENT_IVS_TRAFFIC_CROSSLANE	0x0000010A	Event of violating regulations by crossing lanes	DEV_EVENT_TRAFFIC_CROSSLANE_INFO
EVENT_IVS_TRAFFIC_OVERYELLOWLINE	0x0000010B	Event of running on the yellow line	DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO
EVENT_IVS_TRAFFIC_DRIVINGONSHOULDER	0x0000010C	Event of running on the road shoulder	DEV_EVENT_TRAFFIC_DRIVINGONSHOULDER_INFO
EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE	0x0000010E	Event of yellow plate occupying the lanes	DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO
EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY	0x0000010F	Event of pedestrian priority at zebra crossing	DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO
EVENT_IVS_TRAFFIC_PARKINGONYELLOWBOX	0x0000012A	Event of capturing the cars parking at the yellow box	DEV_EVENT_TRAFFIC_PARKINGONYELLOWBOX_INFO
EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING	0x0000012B	Event of parking space taken by cars	DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO
EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING	0x0000012C	Event of parking space taken by no cars	DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO
EVENT_IVS_TRAFFIC_PEDESTRAIN	0x0000012D	Event about pedestrian	DEV_EVENT_TRAFFIC_PEDESTRAIN_INFO
EVENT_IVS_TRAFFIC_THROW	0x0000012E	Event of throwing objects	DEV_EVENT_TRAFFIC_THROW_INFO
EVENT_IVS_TRAFFIC_IDLE	0x0000012F	Idle event	DEV_EVENT_TRAFFIC_IDLE_INFO
EVENT_IVS_TRAFFIC_RESTRICTED_PLATE	0X00000136	Event of restricted plate	DEV_EVENT_TRAFFIC_RESTRICTED_PLATE
EVENT_IVS_TRAFFIC_OVERSTOPLINE	0X00000137	Event of pressing on the stop line	DEV_EVENT_TRAFFIC_OVERSTOPLINE

dwAlarmType macro definition	Value of macro definition	Meaning	Corresponding structure of pAlarmInfo
EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT	0x00000138	Event of safety belt unfastened	DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT
EVENT_IVS_TRAFFIC_DRIVER_SMOKING	0x00000139	Event of driver smoking	DEV_EVENT_TRAFFIC_DRIVER_SMOKING
EVENT_IVS_TRAFFIC_DRIVER_CALLING	0x0000013A	Event of driver calling	DEV_EVENT_TRAFFIC_DRIVER_CALLING
EVENT_IVS_TRAFFIC_PEDESTRAINRUNREDLIGHT	0x0000013B	Event of pedestrian running the red light	DEV_EVENT_TRAFFIC_PEDESTRAINRUNREDLIGHT_INFO
EVENT_IVS_TRAFFIC_PASSNOTINORDER	0x0000013C	Event of passing without order	DEV_EVENT_TRAFFIC_PASSNOTINORDER_INFO

3.2.3.2 CLIENT_StopLoadPic

For the interface function, see "3.2.2.3 CLIENT_StopLoadPic."

3.2.4 Vehicle Flow Statistics

3.2.4.1 CLIENT_StartTrafficFluxStat

Table 3-35 Subscribe the statistics of vehicle flow

Item	Description	
Name	Subscribe the statistics of vehicle flow.	
Function	LLONG CLIENT_StartTrafficFluxStat(LLONG lLoginID, NET_IN_TRAFFICFLUXSTAT* pstInParam, NET_OUT_TRAFFICFLUXSTAT* pstOutParam);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity
	[in] pstInParam	Input parameter. Vehicle flow statistics callback: fFluxStatDataCallBack.
	[out] pstOutParam	Output parameter.
Return value	● Success: Not 0. ● Failure: 0.	
Note	None.	

3.2.4.2 CLIENT_StopTrafficFluxStat

Table 3-36 Stop subscribing the statistics of vehicle flow

Item	Description	
Name	Stop subscribing the statistics of vehicle flow	

Function	BOOL CLIENT_StopTrafficFluxStat(
	    LLONG lFluxStatHandle
	);	

Parameter	[in] lFluxStatHandle	Return value of CLIENT_StartTrafficFluxStat
Return value	● Success: TRUE ● Failure: FALSE	
Note	None	

3.3 Parking Lot

3.3.1 Barrier Control

3.3.1.1 CLIENT_ControlDeviceEx

For the interface function, see "3.2.2.2 CLIENT_ControlDeviceEx."

Table 3-37 Control type

emType enumeration definition	Meaning	Corresponding structure of pInBuf
DH_CTRL_OPEN_STROBE	Open barrier	NET_CTRL_OPEN_STROBE
DH_CTRL_CLOSE_STROBE	Close barrier	NET_CTRL_CLOSE_STROBE

3.3.1.2 CLIENT_SetConfig

Table 3-38 Set barrier configuration

Item	Description	
Name	Set barrier configuration.	

Function	BOOL CLIENT_SetConfig (
	    LLONG lLoginID
	    NET_EM_CFG_OPERATE_TYPE emCfgOpType
	    int nChannelID
	    void* szInBuffer
	    DWORD dwInBufferSize
	    int waittime=3000
	    int * restart=NULL
	    void * reserve=NULL
	);	

Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.

Item	Description	
	[in] emCfgOpType	Set cofniguration type Barrier configuration: NET_EM_CFG_TRAFFICSTROBE
	[out] nChannelID	Channel number.
	[in] szInBuffer	The buffer address of the confuguration.
	[in] dwInBufferSize	The size of the buffer address.
	[in] waittime	Timeout.
	[in] restart	Whether to restart.
	[in] reserve	Reserved parameters
Return value	● Success: TRUE ● Failure: FALSE.	
Note	None.	

3.3.1.3 CLIENT_GetConfig

Table 3-39 Get barrier configuration

Item	Description	
Name	Get barrier configuration	
Function	BOOL CLIENT_GetConfig (LLONG lLoginID NET_EM_CFG_OPERATE_TYPE emCfgOpType int nChannelID void* szOutBuffer DWORD dwOutBufferSize int waittime=3000 void * reserve=NULL);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] emCfgOpType	Set cofniguration type Barrier configuration: NET_EM_CFG_TRAFFICSTROBE
	[out] nChannelID	Channel number.
	[in] szInBuffer	Get he buffer address of the confuguration.
	[in] dwInBufferSize	The size of the buffer address.
	[in] waittime	Timeout.
	[in] reserve	The size of gotten configuration.
Return value	● Success: TRUE ● Failure: FALSE.	
Note	None.	

3.3.1.4 CLIENT_SetDVRMessCallBack

Table 3-40 Set vehicle location information callback

Item	Description	
Name	Set vehicle location information callback	
Function	void CLIENT_SetDVRMessCallBack(fMessCallBack cbMessage, LDWORD dwUser);	
Parameters	[in] cbMessage	Alarm callback
	[in] dwUser	User data.
Return value	None.	
Note	Call CLIENT_SetDVRMessCallBack interface before alarm subscribe; the set callback cannot include the event with pictures.	

3.3.1.5 CLIENT_StartListenEx

Table 3-41 Subscribe vehicle location information

Item	Description	
Name	Subscribe vehicle location information	
Function	BOOL CLIENT_StartListenEx(LLONG lLoginID);	
	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	● Success: TRUE ● Failure: FALSE.	
Note	The all alarm events are reported to the users through the calback set by CLIENT_SetDVRMessCallBack interface.	

3.3.1.6 CLIENT_StopListen

Table 3-42 Stop subscribing vehicle location information

Item	Description	
Name	Stop subscribing vehicle location information	
Function	BOOL CLIENT_StopListen(LLONG lLoginID);	
Parameters	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	● Success: TRUE ● Failure: FALSE.	
Note	None.	

3.3.1.7 CLIENT_RealLoadPictureEx

For details, see "3.2.3.1 CLIENT_RealLoadPictureEx."

3.3.1.8 CLIENT_StopLoadPic

For details, see "CLIENT_StopLoadPic"

3.3.2 Importing/Exporting Banned/Trusted List: CLIENT_FileTransmit

Table 3-43 Importing/Exporting Banned/Trusted List

Item	Description	
Name	Transmit files.	
Function	LLONG CLIENT_FileTransmit (LLONG lLoginID, Int nTransType char* szInBuf int nInBufLen fTransFileCallBack cbTransFile LDWORD dwUserData Int waittime);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] nTransType	File control type. See Table 3-44.
	[in] szInBuf	Input data, see Table 3-44.
	[in] nInBufLen	The size of nInBufLen is no smaller tham that of szInBufszInBuf structure.
	[in] cbTransFile	fTransFileCallBack.
	[in] dwUserData	Custom data.
	[in] waittime	Timeout.
Return value	● Start sending/downloading banned/trusted list, when the return file handle> 0, it is a valid handle; when the return file handle ≤0, it is an invalid handle. ● Success: TRUE; failure: FALSE.	
Note	None.	

Table 3-44 File control type

nTransType enumerate definition	Value	Description	szInBuf
DH_DEV_BLACKWHITETRANS_START	0x0003	Start sending banned/trusted list	DHDEV_BLACKWHITE_LIST_INFO

nTransType enumerate definition	Value	Description	szInBuf
DH_DEV_BLACKWHITETRANS_SEND	0x0004	Send banned/trusted list	LONG, the return enumerate of starting sending file
DH_DEV_BLACKWHITETRANS_STOP	0x0005	Stop sending banned/trusted list	LONG, the return enumerate of starting sending file
DH_DEV_BLACKWHITE_LOAD	0x0006	Download banned/trusted list	DHDEV_LOAD_BLACKWHITE_LIST_INFO
DH_DEV_BLACKWHITE_LOAD_STOP	0x0007	Stop downloading banned/trusted list	LONG, the return enumerate of starting sending file

3.3.3 Voice Talk

3.3.3.1 CLIENT_GetDevProtocolType

Table 3-45 Get the supported voice talk type

Item	Description	
Name	Get the supported voice talk type.	
Function	BOOL CLIENT_GetDevProtocolType(LLONG lLoginID, EM_DEV_PROTOCOL_TYPE *pemProtocolType);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] pemProtocolType	The supported protocol type, the corresponding structure is EM_DEV_PROTOCOL_TYPE.
Return value	● Success: TRUE ● Failure: FALSE.	
Note	None.	

3.3.3.2 CLIENT_SetDeviceMode

Table 3-46 Set the working mode of voice talk

Item	Description
Name	Set the working mode of voice talk.
Function	BOOL CLIENT_SetDeviceMode(LLONG lLoginID, EM_USEDEV_MODE emType, void *pValue

Item	Description	
	);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] emType	enumeration value.
	[in] pValue	The the corresponding structure data pointer of the enumeration value, see Table 3-47.
Return value	● Success: TRUE ● Failure: FALSE.	
Note	None.	

Table 3-47 Relationship of emType and pValue

emType	Description	pValue
DH_TALK_ENCODE_TYPE	Talk in the pointed node.	DHDEV_TALKDECODE_INFO
DH_TALK_CLIENT_MODE	Set voice talk client.	None.
DH_TALK_SPEAK_PARAM	Set speak parameters.	NET_SPEAK_PARAM
DH_TALK_MODE3	Set speak parameters of the the third generation deveice.	NET_TALK_EX

3.3.3.3 CLIENT_StartTalkEx

Table 3-48 Start voice talk

Item	Description	
Name	Start voice talk.	

Function	LLONG CLIENT_StartTalkEx(
	    LLONG lLoginID,
	    pfAudioDataCallBack pfcb,
	    LDWORD dwUser
	);	

Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] pfcb	Audio data callback.
	[in] dwUser	The parameters of audio data callback.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.3.4 CLIENT_StopTalkEx

Table 3-49 Stop voice talk

Item	Description
Name	Stop voice talk.

Function	BOOL CLIENT_StopTalkEx(
	    LLONG lTalkHandle
	);

Item	Description	
Parameter	[in] lTalkHandle	Return value of CLIENT_StartTalkEx.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.3.5 CLIENT_RecordStartEx

Table 3-50 Start local record

Item	Description	
Name	Start local record.	
Function	BOOL CLIENT_RecordStartEx(LLONG lLoginID);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	● Success: TURE. ● Failure: FALSE.	
Note	This interface is only valid in Windows.	

3.3.3.6 CLIENT_RecordStopEx

Table 3-51 Stop local record

Item	Description	
Name	Stop local record.	
Function	BOOL CLIENT_RecordStopEx(LLONG lLoginID);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	This interface is only valid in Windows.	

3.3.3.7 CLIENT_TalkSendData

Table 3-52 Set audio data to devices

Item	Description
Name	Set audio data to devices.
Function	LONG CLIENT_TalkSendData(LLONG lTalkHandle, char *pSendBuf, DWORD dwBufSize

Item	Description	
	);	
Parameter	[in] lTalkHandle	Return value of CLIENT_StartTalkEx.
	[in]pSendBuf	The pointer of the audio data module to be sent.
	[in]dwBufSize	The length of the audio data module to be sent, unit: byte.
Return value	● Success: The length of the audio data module. ● Failure: –1.	
Note	None.	

3.3.3.8 CLIENT_AudioDecEx

Table 3-53 Decode audio data

Item	Description	
Name	Decode audio data.	
Function	BOOL CLIENT_AudioDecEx(LLONG lTalkHandle, char *pAudioDataBuf, DWORD dwBufSize);	
Parameter	[in] lTalkHandle	Return value of CLIENT_StartTalkEx.
	[in] pAudioDataBuf	The pointer of the audio data module to be decoded.
	[in] dwBufSize	The length of the audio data module to be decoded, unit: byte.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.3.9 CLIENT_SetDVRMessCallBack

Set the device requesting the other device to start voice talk event. For details, see "CLIENT_SetDVRMessCallBack."

3.3.3.10 CLIENT_StartListenEx

Subscribe the device requesting the other device to start voice talk event. For details, see "3.3.1.5 CLIENT_StartListenEx."

3.3.3.11 CLIENT_StopListen

Stop subscribing the device requesting the other device to start voice talk event. For details, see "3.3.1.6 CLIENT_StopListen."

3.3.4 Dot-matrix Display Character Control

3.3.4.1 CLIENT_SetConfig

Set the dot-matrix display configuration. For details, see "CLIENT_SetConfig." emCfgOpType is NET_EM_CFG_TRAFFIC_LATTIC_SCREEN.

3.3.4.2 CLIENT_GetConfig

Get the dot-matrix display configuration. For details, see "3.3.1.3 CLIENT_GetConfig." emCfgOpType is NET_EM_CFG_TRAFFIC_LATTIC_SCREEN.

3.3.5 Parking Space Indicator Configuration

3.3.5.1 CLIENT_PacketData

Table 3-54 Pack the configuration

Item	Description	
Name	Pack the cofiguration.	
Function	BOOL CLIENT_PacketData(char* szCommand, LPVOID lpInBuffer, DWORD dwInBufferSize, char* szOutBuffer, DWORD dwOutBufferSize);	
Parameter	[in] szCommand	Command parameter. Parking space indicator configuration: CFG_CMD_PARKING_SPACE_LIGHT_GROUP.
	[in] lpInBuffer	Input buffer.
	[in] dwInBufferSize	The size of the input buffer.
	[out] szOutBuffer	Output buffer.
	[in] dwOutBufferSize	The size of the output buffer.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.5.2 CLIENT_SetNewDevConfig

Table 3-55 Set the configuration

Item	Description
Name	Set the cofiguration.

Item	Description	
Function	BOOL CLIENT_SetNewDevConfig(LLONG lLoginID, char* szCommand, int nChannelID, char* szInBuffer, DWORD dwInBufferSize, int *error, int *restart, int waittime=500);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] szCommand	Command parameter. Parking space indicator configuration: CFG_CMD_PARKING_SPACE_LIGHT_GROUP.
	[in] nChannelID	Channel number.
	[in] szInBuffer	Input buffer. It is used for the configured json series information.
	[in] dwInBufferSize	The size of the buffer address.
	[out] error	Error code address.
	[in] restart	Restart sign address.
	[in] waittime	Timeout.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.5.3 CLIENT_GetNewDevConfig

Table 3-56 Get the configuration

Item	Description	
Name	Get the cofiguration.	
Function	BOOL CLIENT_GetNewDevConfig(LLONG lLoginID, char* szCommand, int nChannelID, char* szOutBuffer, DWORD dwOutBufferSize, int *error, int waittime=500);	
Parameter	[in] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[in] szCommand	Command parameter. Parking space indicator configuration:

Item	Description	
		CFG_CMD_PARKING_SPACE_LIGHT_GROUP.
	[in] nChannelID	Channel number.
	[in] szOutBuffer	Output buffer. It is used for the configured json series information..
	[in] dwInBufferSize	The size of the buffer address.
	[out] error	Error code address.
	[in] waittime	Timeout.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.5.4 CLIENT_ParseData

Table 3-57 Parse the configuration

Item	Description	
Name	Paese the cofiguration.	

Function	BOOL CLIENT_ParseData(
	    char* szCommand,
	    char* szInBuffer,
	    LPVOID lpOutBuffer,
	    DWORD dwOutBufferSize,
	    void* pReserved
	);	

Parameter	[in] szCommand	Command parameter. Parking space indicator configuration: CFG_CMD_PARKING_SPACE_LIGHT_GROUP.
	[in] szInBuffer	Input buffer, character configuration buffer.
	[in] lpOutBuffer	Output buffer.
	[out]dwOutBufferSize	The size of output buffer.
	[in] pReserved	Reserved parameters.
Return value	● Success: TRUE. ● Failure: FALSE.	
Note	None.	

3.3.6 Parking Space Status Indicator Configuration

3.3.6.1 CLIENT_SetConfig

Set the parking space status indicator. For details, see "3.3.1.2 CLIENT_SetConfig." emCfgOpType is NET_EM_CFG_PARKINGSPACELIGHT_STATE。

3.3.6.2 CLIENT_GetConfig

Get the parking space status indicator. For details, see "3.3.1.3 CLIENT_GetConfig."

emCfgOpType is NET_EM_CFG_PARKINGSPACELIGHT_STATE。

4 Callback Definition

4.1 fSearchDevicesCB

Table 4-1 Callback of searching devices (1)

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

4.2 fSearchDevicesCBEx

Table 4-2 Callback of searching devices (2)

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

4.3 fDisConnect

Table 4-3 Disconnection callback

Item	**Description**
Name	Disconnection callback.
Function	typedef void (CALLBACK *fDisConnect)(LLONG lLoginID, char *pchDVRIP, LONG nDVRPort, LDWORD dwUser

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

4.4 fHaveReConnect

Table 4-4 Reconnection callback

Item	Description	
Name	Reconnection callback.	

Function	typedef void (CALLBACK *fHaveReConnect)(
	    LLONG lLoginID,
	    char *pchDVRIP,
	    LONG nDVRPort,
	    LDWORD dwUser
	);	

Parameter	[out] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] pchDVRIP	IP of the reconnected device.
	[out] nDVRPort	Port of the reconnected device.
	[out] dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.5 fRealDataCallBackEx2

Table 4-5 Callback of real-time monitoring data

Item	Description	
Name	Callback of real-time monitoring data.	

Function	typedef void (CALLBACK * fRealDataCallBackEx2)(
	    LLONG lRealHandle,
	    DWORD dwDataType,
	    BYTE *pBuffer,
	    DWORD dwBufSize,
	    LLONG param,
	    LDWORD dwUser
	);	

Parameter	[out] lRealHandle	Return value of CLIENT_RealPlayEx.
	[out] dwDataType	Data type:

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

4.6 fDownLoadPosCallBack

Table 4-6 Callback of media file download process

Item	Description	
Name	Callback of media file download process.	
Function	typedef void (CALLBACK *fDownLoadPosCallBack)(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, LDWORD dwUser);	
Parameter	[out]lPlayHandle	Return value of CLIENT_DownloadMediaFile.
	[out]dwTotalSize	Total size.
	[out]dwDownLoadSize	The downloaded data size. ● -1: Download finish. ● -2: Data write error during downloading.
	[out]dwUser	User parameter of the callback.
Return value	None.	
Note	None.	

4.7 fAnalyzerDataCallBack

Table 4-7 Callback of intelligent event information

Item	Description	
Name	Callback of intelligent event information.	
Function	typedef int (CALLBACK *fAnalyzerDataCallBack)(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE* pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void* reserved);	
Parameter	[out]lAnalyzerHandle	Return value of CLIENT_RealLoadPictureEx.
	[out]dwAlarmType	Type of intelligent event, see 错误!未找到引用源。.
	[out]pAlarmInfo	Cache of event information, see 错误!未找到引用源。.
	[out]pBuffer	Pictures cache.
	[out]dwBufSize	Cache size of pictures.
	[out]dwUser	User parameter of the callback.
	[out]reserved	Reserved.
Return value	None.	
Note	None.	

4.8 fFluxStatDataCallBack

Table 4-8 Callback of intelligent event information

Item	Description	
Name	Callback of intelligent event information.	
Function	typedef int (CALLBACK *fFluxStatDataCallBack)(LLONG lFluxStatHandle, DWORD dwEventType, void* pEventInfo, BYTE* pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void* reserved);	
Parameter	[out]lFluxStatHandle	Return value of CLIENT_StartTrafficFluxStat.
	[out]dwEventType	Type of intelligent event information.
	[out]pEventInfo	Vehicle flow event information.
	[out]pBuffer	Data cache.
	[out]dwBufSize	Data size.

Item	Description	
	[out]dwUser	User parameter of the callback.
	[out]nSequence	Sequence.
	[out]reserved	Reserved.
Return value	None.	
Note	The pEventInfo corresponds to DEV_EVENT_TRAFFIC_FLOWSTAT_INFO structure.	

4.9 fTransFileCallBack

Table 4-9 Callback of file transmission

Item	Description	
Name	Callback of file transmission.	
Function	typedef int (CALLBACK *fFluxStatDataCallBack)(LLONG lHandle, int nTransType, int nState, int nSendSize, int nTotalSize, LDWORD dwUser);	
Parameter	c	File transmission handle.
	[out] nTransType	The type of file transmission.
	[out] nState	The status of file transmission.
	[out] nSendSize	The length of the sent file.
	[out] nTotalSize	The total size of the file.
	[out] dwUser	Custom data.
Return value	None.	
Note	None.	

4.10 pfAudioDataCallBack

Table 4-10 Callback of audio data of voice talk

Item	Description
Name	Callback of audio data of voice talk.
Function	typedef void (CALLBACK *pfAudioDataCallBack)(LLONG lTalkHandle, char *pDataBuf, DWORD dwBufSize, BYTE byAudioFlag, LDWORD dwUser);

Item	**Description**	
Parameter	[out] lTalkHandle	Return value of CLIENT_StartTalkEx.
	[out] pDataBuf	The address of audio data module.
	[out] dwBufSize	The length of audio data module.
	[out] byAudioFlag	Data type signs: ● 0: Indicates it is from local collection. ● 1: Indicates it is from device sending.
	[out] dwUser	Callback of user parameters .
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.