NetSDK (Intelligent AI)

Programming Manual

V1.0.2

Foreword

Purpose

Welcome to use NetSDK (hereinafter referred to be "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 intelligent function modules for Intelligent Video Surveillance System (IVSS), Network Video Recorder (NVR), IP Camera (IPC), Intelligent Traffic Camera (ITC), people flow statistics devices and barrier. For more function modules and data structures, refer to NetSDK Development Manual.

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

Readers

- SDK software development engineers
- Project managers
- Product managers

Safety Instructions

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

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

Revision History

Version	Revision Content	Release Time
V1.0.2	Updated some descriptions.	February 2023
V1.0.1	Updated the dependent library information.	April 2021
V1.0.0	First release.	October 2018

Privacy Protection Notice

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

About the Manual

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

Glossary

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

Term	Definition
IVSS	Intelligent Video Surveillance System is different with the NVR devices which only support the storage function. The IVSS adds the intelligent analysis function to form an integrated management system.
Face Detection	Do the intelligent analysis to detect the people face, age, sex and expression in the video.
Target Recognition	It contains the face detection. You can detect whether the faces in the video are in the face library or not through the intelligent analysis.
History library	It can be used to storage the face pictures that captured by the device.
Face library	You can detect whether the faces are in the face library or not through importing some face images to the IVSS, NVR or the front-end devices in advance.
Arm by channel	Arm/Disarm one or multiple face library to one channel. The scene: The detected face in this channel contrast with the arm library and return the result. It belongs to one mode of the face library arming.
Arm by library	Arm the face library to one or multiple channel. The scene: The people face detected by the channel contrast with this face library and return the result. It belongs to one mode of the face library arming.
Search picture by picture	You can import a picture and a similarity value, and then the IVSS and NVR will search the history library and the face library by this picture to make sure whether there have two same faces between the two libraries. And then it will return the right picture.
ITC	Intelligent Traffic Camera. It can capture the vehicle pictures and automatically analyze the traffic events.
Tripwire detect	Automatically detect the cross trip wire.
Intrusion detect	Automatically detect whether the object enter the alert area or not.
People flow	People information in the camera marking area.
People counting	Real-time count the number of people entering or leaving the camera marking area.
People counting in area	Real-time count the number of people in the camera marking area.

Table of Contents

1 Overview

1.1 General

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

The main function contains the general functions, face detection and target recognition, body detection, people flow statistics, general behavior event, intelligent traffic and barrier.

The development kit might be different dependent on the different environments.

Table 1-1 The files included in development kit

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

Table 1-2 The files included in 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	Header file
	libdhconfigsdk.so	Library file
Auxiliary library of "libdhnetsdk.so"	libStreamConvertor.so	Transcoding library

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

1.2 Applicability

- Recommended memory: No less than 512 M.
- System supported by SDK:
  - ◇ Windows
    Windows 10/ Windows 8.1/ Windows 7/ vista/ 2000 and Windows Server 2008/ 2003.
  - ◇ Linux
    The common Linux systems such as Red Hat/SUSE

1.3 Application Scenario

1.3.1 Face Detection/Target Recognition/Body Detection

Figure 1-1 Target recognition

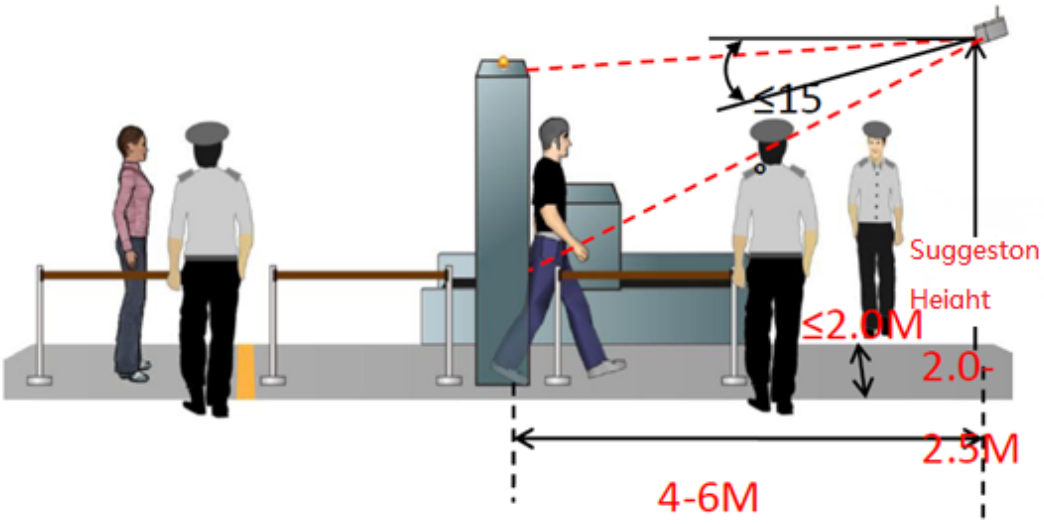
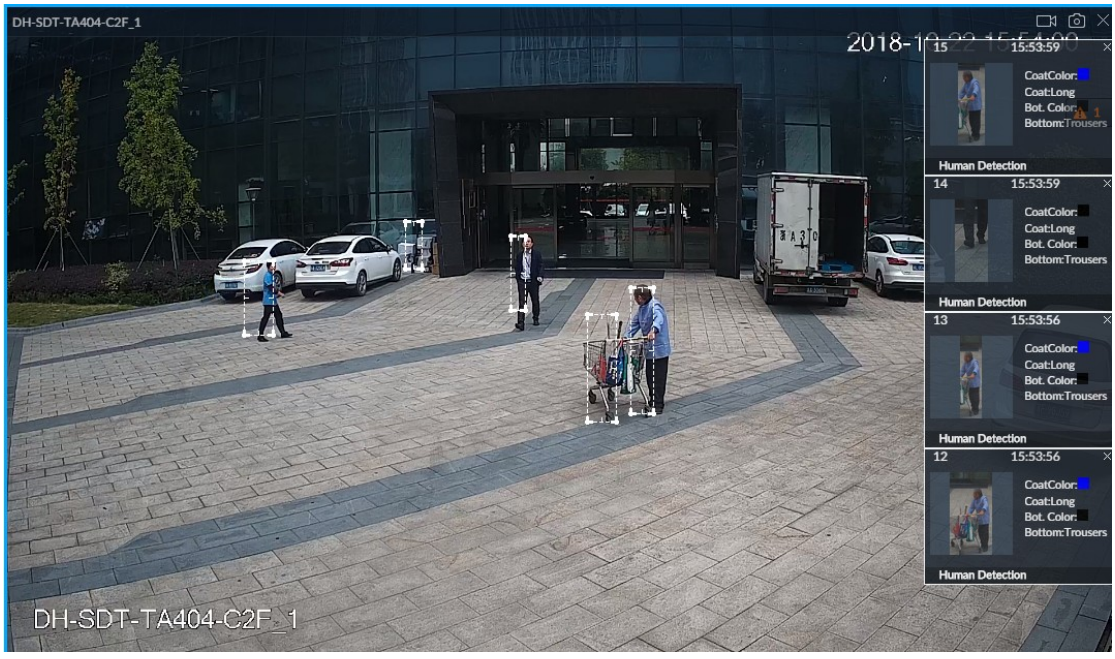


Figure 1-2 Body detection scenario



1.3.2 People Flow Statistics

For the application of people flow devices, see Figure 1-3.

Figure 1-3 People flow scene

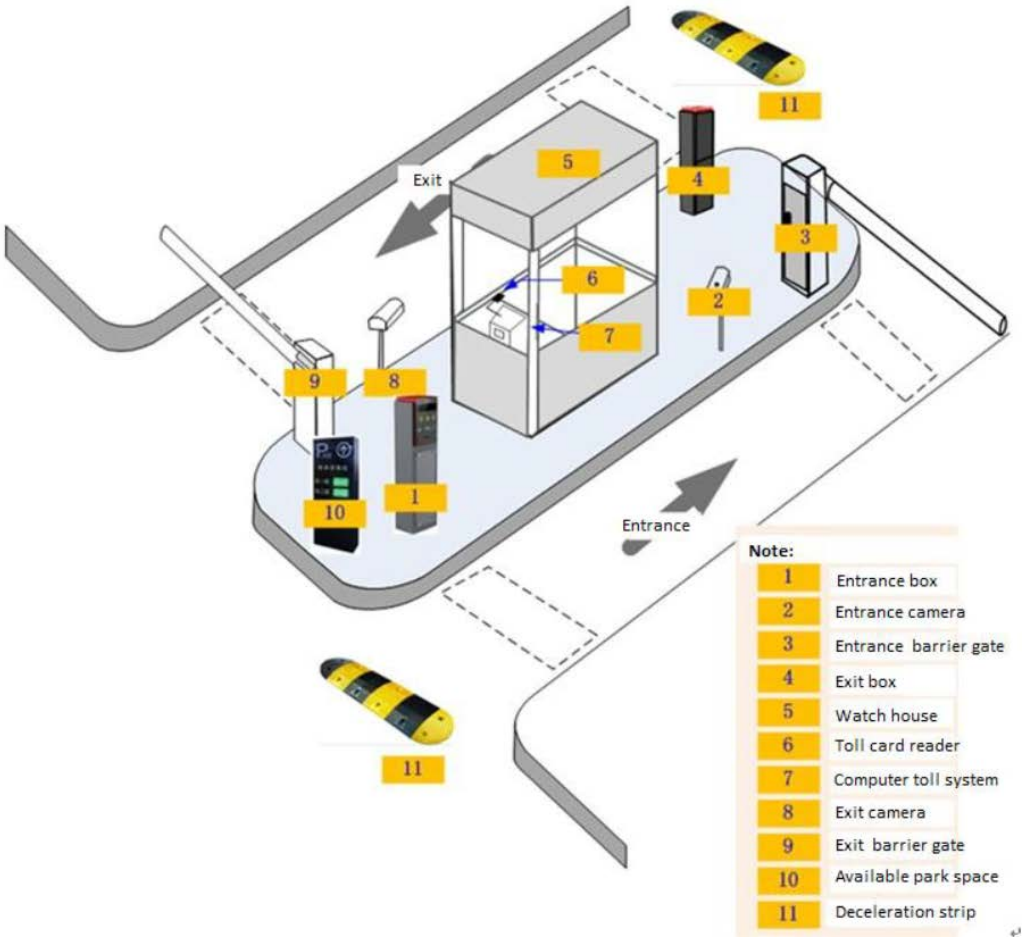
1.3.3 Intelligent Traffic

ITC and ITSE used at the traffic junction capture the traffic violations and count the vehicle flow. See Figure 1-4.

Figure 1-4 ITC and ITSET used at the traffic junction

ITC, ITSE and IPMECK used at the parking access control the vehicle enter and exit the parking and monitor whether there is any parking space.See Figure 1-5.

Figure 1-5 ITC, ITSE and IPMECK used at the parking access



1.3.4 General Behavior

People or vehicle across the rule line (tripwire) or intruding the alert zone (intrusion) can generate alarm events. Meanwhile this function can distinguish the target objects (People or vehicle).

Figure 1-6 General behavior scenario (tripwire)

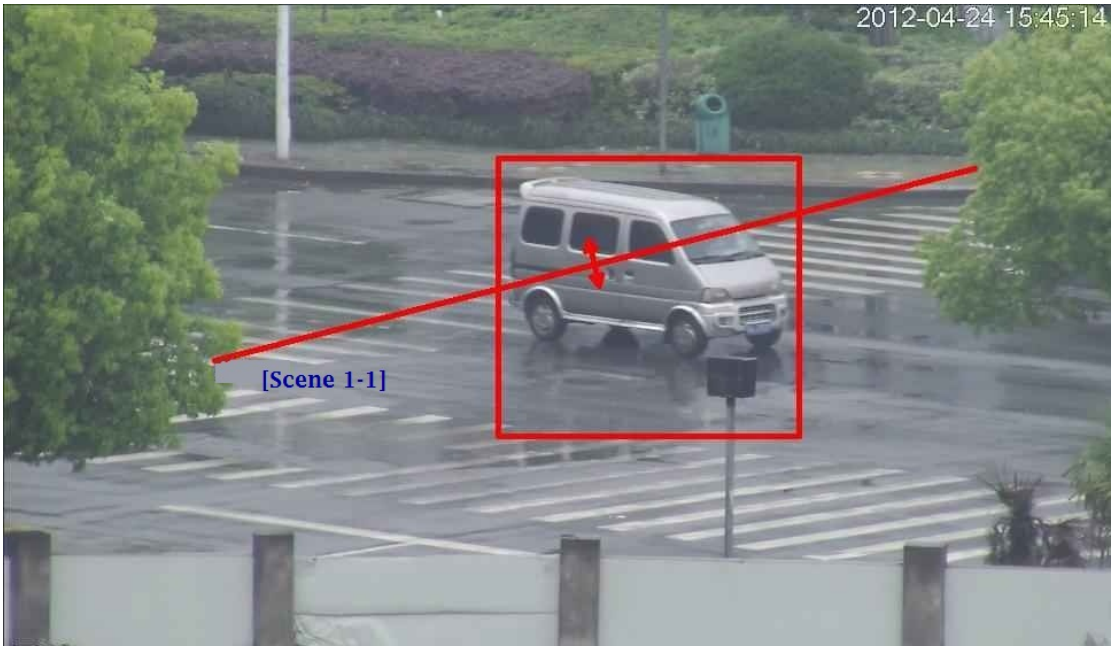


Figure 1-7 General behavior scenario (intrusion)

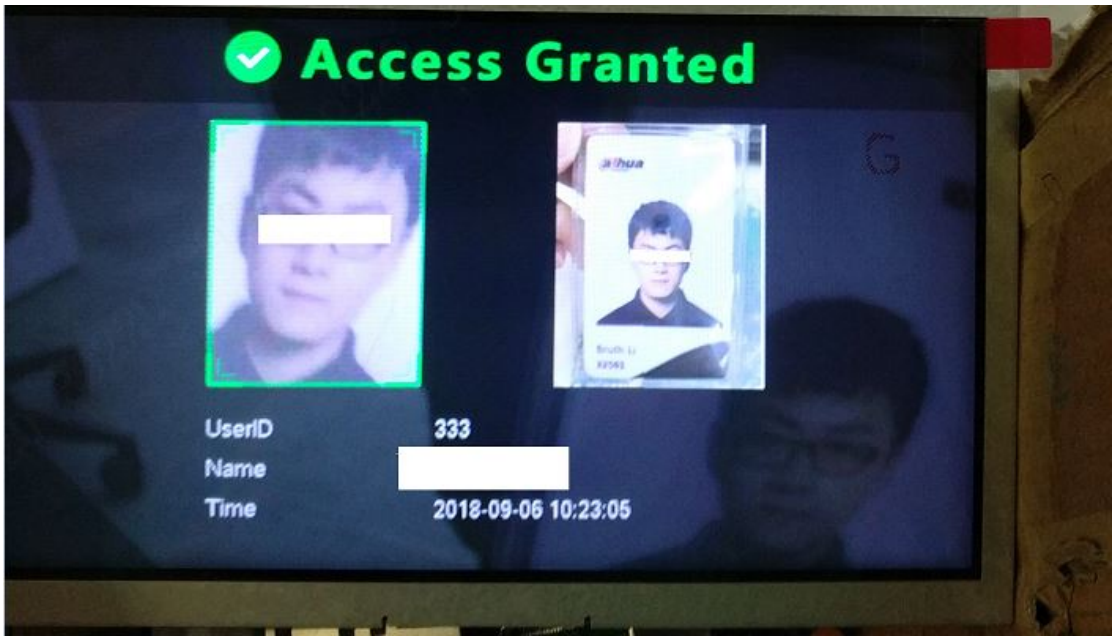
1.3.5 Access Control System

Access control barrier mainly used at the campus, school, business building. You can send the people face pictures and people information to the platform, and then the platform send the data to the barrier system.

Figure 1-8 The appearance of swing barrier

You can open the barrier system by people face or card.

Figure 1-9 Open barrier by people face



2 Function Modules

2.1 SDK Initialization

2.1.1 Introduction

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

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

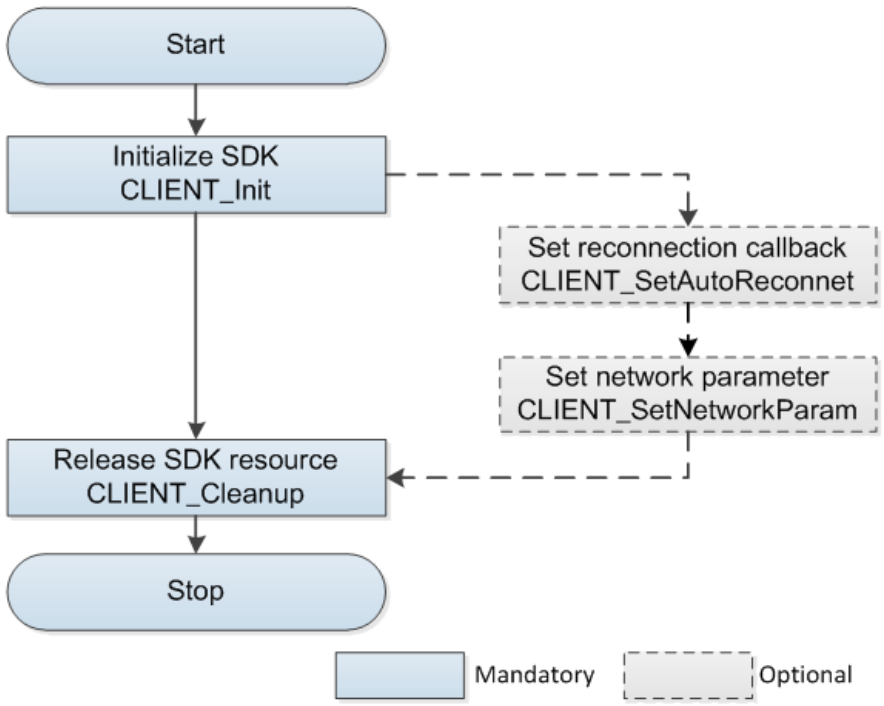
2.1.2 Interface Overview

Table 2-1 Interfaces of SDK initialization

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

2.1.3 Process

Figure 2-1 Process of SDK initialization



Process Description

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

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

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

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

Notes for Process

- Call **CLIENT_Init** and **CLIENT_Cleanup** in pairs. It supports 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.

2.1.4 Example Code

```
// Set this callback through CLIENT_Init. When the device is disconnected, SDK informs the user through this callback.
void CALLBACK DisConnectFunc(LONG lLoginID, char *pchDVRIP, LONG nDVRPort, DWORD dwUser)
{
	printf("Call DisConnectFunc: lLoginID[0x%x]\n", lLoginID);
}
// Initialize SDK
BOOL bNetSDKInitFlag = CLIENT_Init(DisConnectFunc, 0);
if (FALSE == bNetSDKInitFlag)
{
	printf("Initialize client SDK fail; \n");
	return -1;
}
// Clean up the SDK resource
if (TRUE == bNetSDKInitFlag)
{
	CLIENT_Cleanup();
}
```

2.2 Device Login

2.2.1 Introduction

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

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

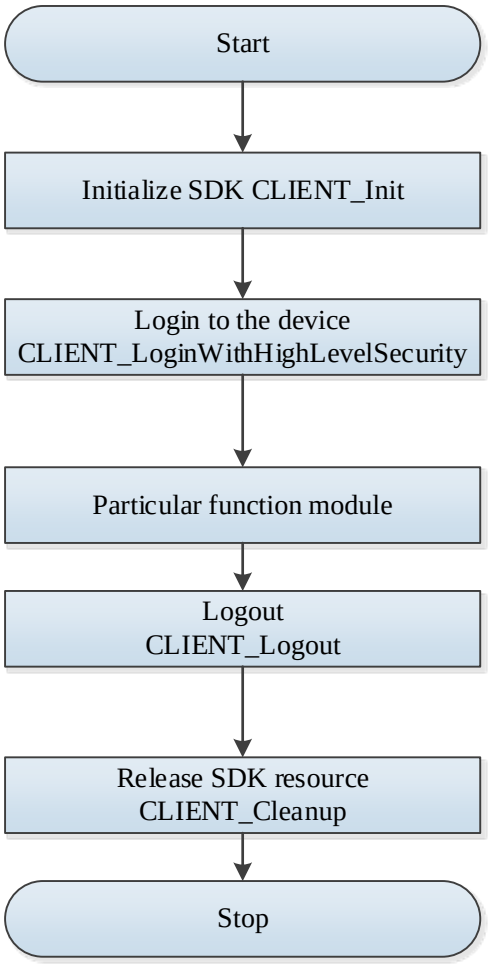
2.2.2 Interface Overview

Table 2-2 Interfaces of device login

Interface	Implication
CLIENT_LoginWithHighLevelSecurity	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.

2.2.3 Process

Figure 2-2 Proces of login



Process Description

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

Notes for Process

- Login handle: When the login is successful, the returned value is not 0 (even the handle is smaller than 0, the login is also successful). One device can login multiple times with different handle at each login. If there is not special function module, it is suggested to login only one time. The login handle can be repeatedly used on other function modules.
- Logout: The interface will release the opened functions internally, but it is not suggested to rely on the cleaning up function. For example, if you opened the monitoring function, you should call the interface that stops the monitoring function when it is no longer required.
- Use login and logout in pairs: The login consumes some memory and socket information and release sources once logout.
- Login failure: It is suggested to check the failure through the error parameter of the login interface.

Table 2-3 Common error code

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 blocklisted.
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	Lack avnetsdk or the dependent libraries of avnetsdk.
12	USB flash disk is not inserted into device, or the USB flash disk information error.
13	The IP at client is not authorized for login.

For more information about error codes, see " CLIENT_LoginWithHighLevelSecurity interface" in Network SDK Development Manual.chm. When the network is poor and this make the error code 3 occur easily, then you can use the following codes to increase timeout time:

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

2.2.4 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);
```

2.3 Real-time Monitoring

2.3.1 Introduction

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

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

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

2.3.2 Interface Overview

Table 2-4 Interfaces of real-time monitoring

Interface	Implication
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_SetRealDataCallBackEx	Set real-time monitoring data callback.

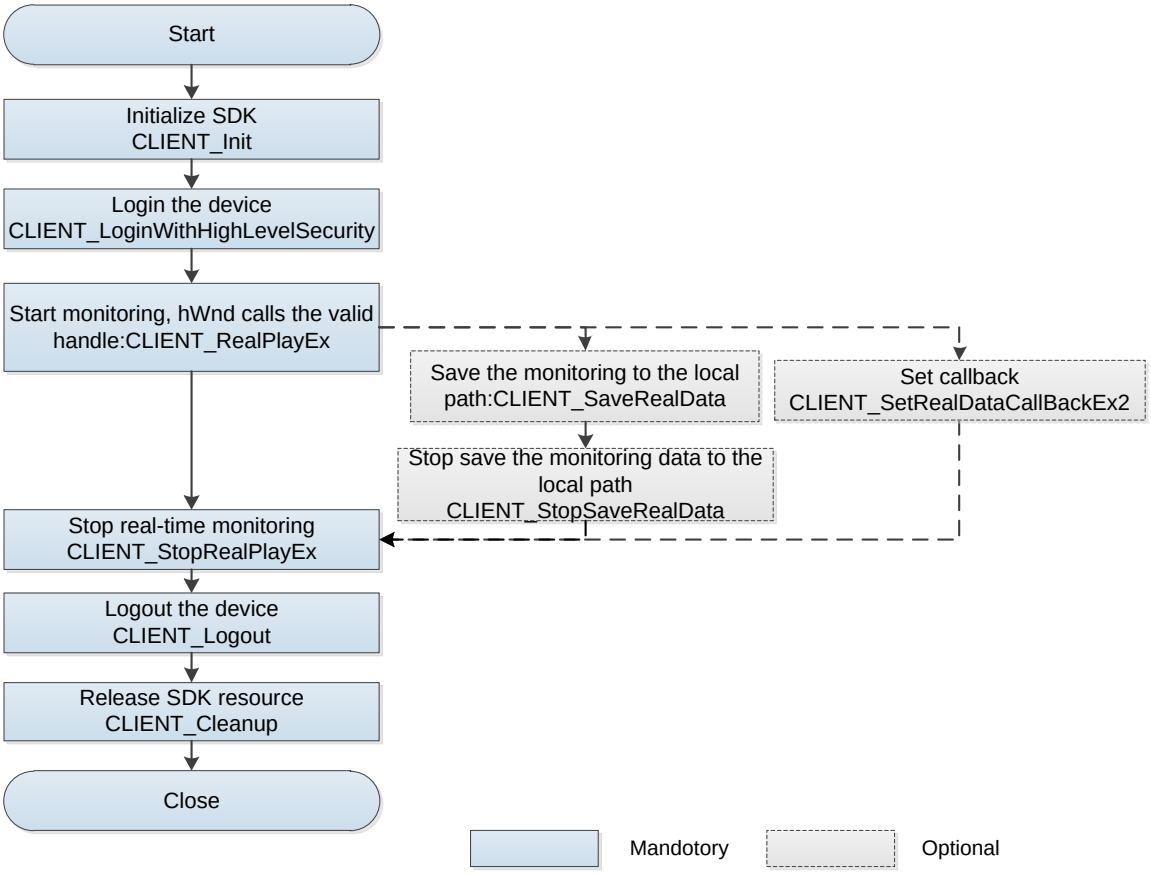
2.3.3 Process

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

2.3.3.1 SDK Decoding Play

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

Figure 2-3 Process of playing by SDK decoding library



Process Description

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

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

Step 3 Call **CLIENT_RealPlayEx** to enable the real-time monitoring. The parameter hWnd is a valid window handle.

Step 4 (Optional) Call **CLIENT_SaveRealData** to start saving the monitoring data.

Step 5 (Optional) Call **CLIENT_StopSaveRealData** to end the saving process and generate the local video file.

Step 6 (Optional) If you call **CLIENT_SetRealDataCallBackEx2**, you can choose to save or forward the video file. If save the video file, see the step 4 and step 5.

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

Step 8 After using the function module, call **CLIENT_Logout** to logout the device.

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

Notes for Process

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

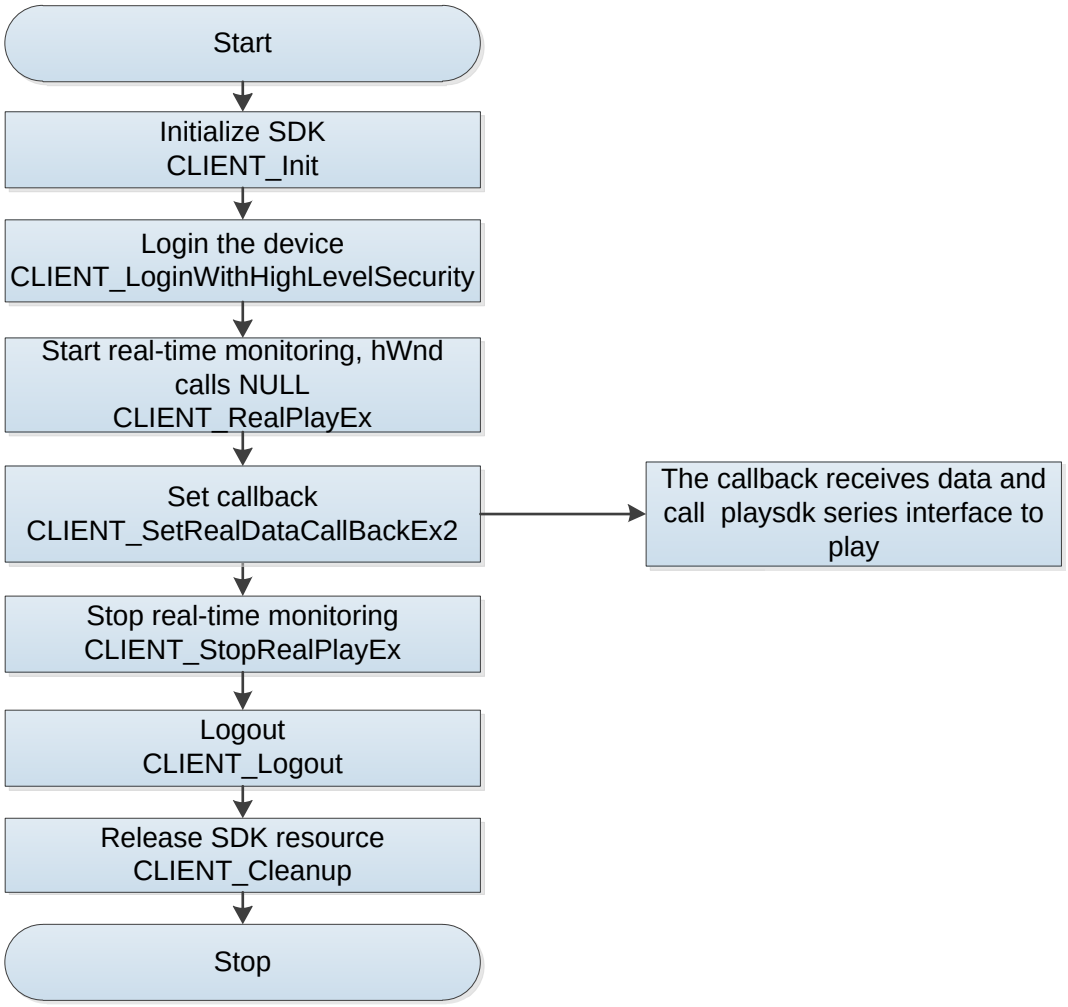
```
NET_PARAM stuNetParam = {0};
stuNetParam. nGetConnInfoTime = 5000; // unit ms
CLIENT_SetNetworkParam (&stuNetParam);
```

- Failed to repeat opening: For some models, the same channel cannot be opened for multiple times during 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 "2.3.3.2 Call Third Party Library".

2.3.3.2 Call Third Party Library

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

Figure 2-4 Process of calling the third party library



Process Description

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

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

Step 3 After successful login, call **CLIENT_RealPlayEx** to enable real-time monitoring. The parameter hWnd is NULL.

Step 4 Call **CLIENT_SetRealDataCallBackEx** to set the real-time data callback.

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

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

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

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

Notes for Process

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

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

2.3.4 Example Code

2.3.4.1 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);
}
```

2.3.4.2 Call Third Party Library

```
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 = 0x00000001;
    CLIENT_SetRealDataCallBackEx(lRealHandle, &RealDataCallBackEx, NULL, dwFlag);
}
// Stop preview
if (0 != lRealHandle)
{
    CLIENT_StopRealPlayEx(lRealHandle);
```

```
}


void CALLBACK RealDataCallBackEx(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LONG 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);
}
```

2.4 Subscribing Intelligent Event

2.4.1 Introduction

Intelligent event subscribe, is that the front-end devices or the back-end devices do the real-time stream analyzing. When detect the preset intelligent event, it uploads the event to the user. The intelligent events in this manual contain general action analysis (such as tripwire, Intrusion), face detection, target recognition, body detection, the intelligent events of intelligent traffic (such as traffic junction, over speed, low speed and traffic jam).

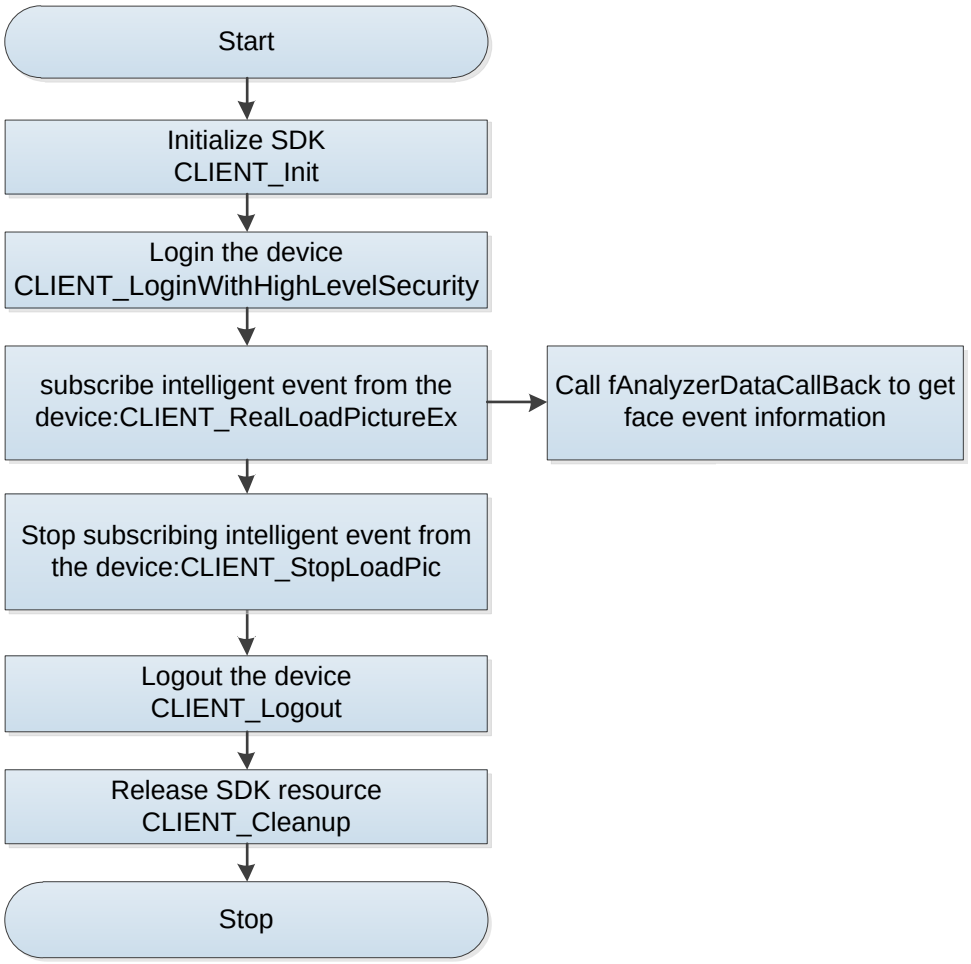
2.4.2 Interface Overview

Table 2-5 Interfaces of subscribing intelligent event

Interface	Implication
CLIENT_RealLoadPictureEx	Subscribe intelligent event.
CLIENT_StopLoadPic	Cancel subscribing the intelligent event.

2.4.3 Process

Figure 2-5 Process of uploading face event



Process Description

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

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

Step 3 Call **CLIENT_RealLoadPictureEx** to subscribe intelligent event from the device.

Step 4 After successful subscribe, call fAnalyzerDataCallBack to upload the intelligent events. Through this function, you can filter out the intelligent events you need.

Step 5 After using the intelligent event function, call **CLIENT_StopLoadPic** to stop subscribing intelligent events.

Step 6 After using the function module, call **CLIENT_Logout** to logout the device.

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

Notes for Process

- Support to subscribe single intelligent event and all the intelligent events (EVENT_IVS_ALL).
- 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.

- Set whether to receive picture or not: You can call CLIENT_RealLoadPictureEx to set bNeedPicFile as False, and then SDK will only receive the face event without picture.

2.4.4 Example Code

```
// Intelligent event uploading callback function
int CALLBACK AnalyzerDataCallBack(LLONG lAnalyzerHandle, DWORD dwAlarmType, void* pAlarmInfo, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser, int nSequence, void *reserved)
{
    switch(dwAlarmType)
    {
    // Filter out the right intelligent events
    ……
    case EVENT_IVS_FACERECOGNITION: // events
    ……
    default:
    break;
    }
}

// Subscribe the uploading of the intelligent 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;
}

// Cancel Subscribing the uploading of the intelligent event
CLIENT_StopLoadPic(lAnalyerHandle);
```

2.5 Searching/Playbacking/Downloading Video and Picture

2.5.1 Introduction

When the device intelligent calculation analysis the real-time stream, once one intelligent event is detected, and then the video and picture of this intelligent event will be saved. You can search the video and picture of the intelligent events which are saved in the device, and also you can do the downloading and playing back operation to the searching result.

2.5.2 Interface Overview

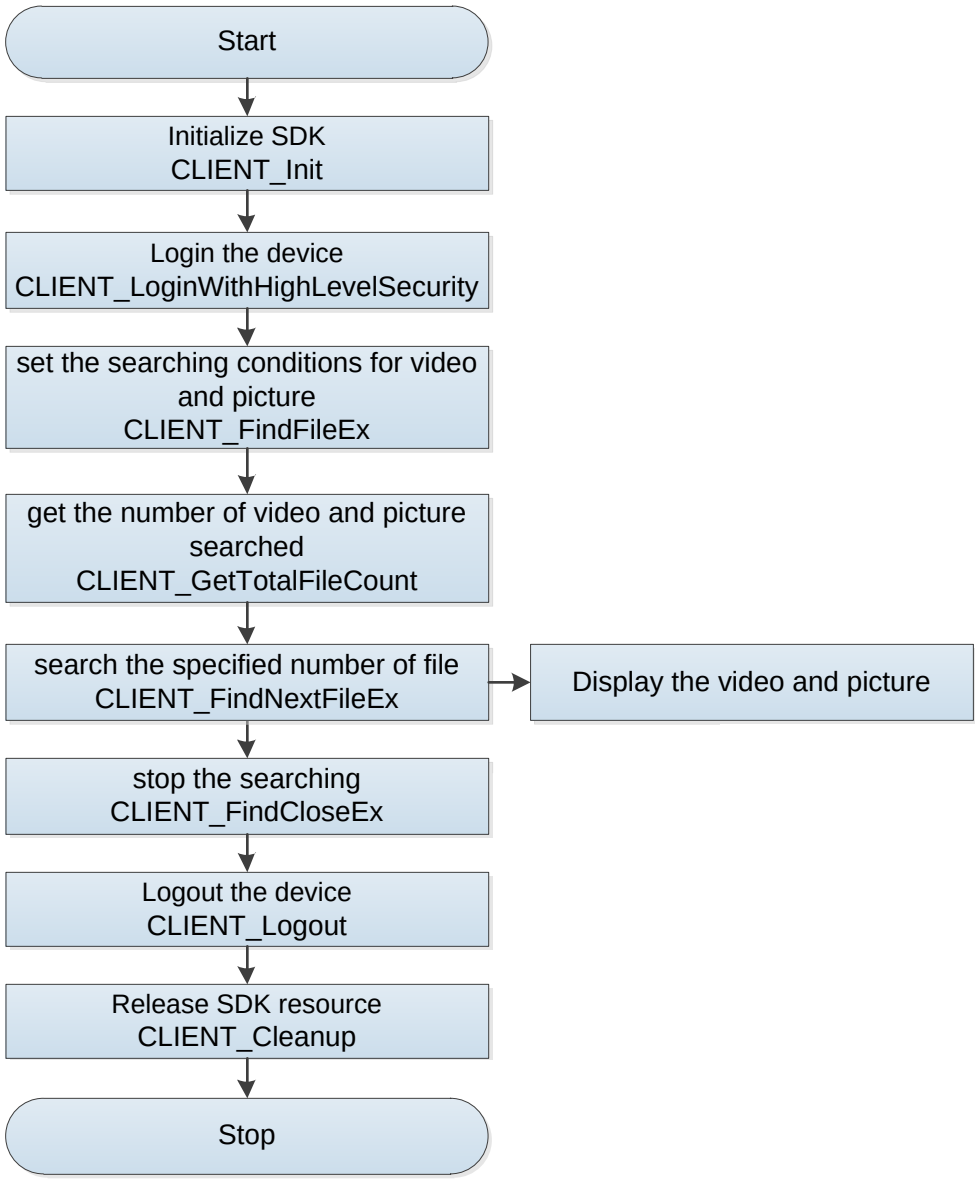
Table 2-6 Interfaces of searching/playvacking/downloading video and picture

Interface	Implication
CLIENT_FindFileEx	Search the video and picture by conditions, and set the searching conditions.
CLIENT_GetTotalFileCount	Obtain the number of video and picture searched now.
CLIENT_FindNextFileEx	Search the specified number of video and picture.
CLIENT_FindCloseEx	Stop searching.
CLIENT_PlayBackByTimeEx2	Start playing back the video by time.
CLIENT_StopPlayBack	Stop playing back the video.
CLIENT_DownloadByTimeEx	Download video.
CLIENT_StopDownload	Stop downloading the video.
CLIENT_DownloadRemoteFile	Download pictures.

2.5.3 Process

2.5.3.1 Searching Process of Video and Picture

Figure 2-6 Searching process of video and picture



Process Description

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

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

Step 3 Call **CLIENT_FindFileEx** to set the searching conditions. After successfully setting, return the searching handle. To judge the right searching type according to the different values of emType.

Step 4 Call **CLIENT_GetTotalFileCount** to get the total number of video and picture searched.

Step 5 Call **CLIENT_FindNextFileEx** to search the specified number of video and picture. Save the video and picture and do the playing back and downloading operation to the video and picture.

Step 6 Call **CLIENT_FindCloseEx** to stop the searching.

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

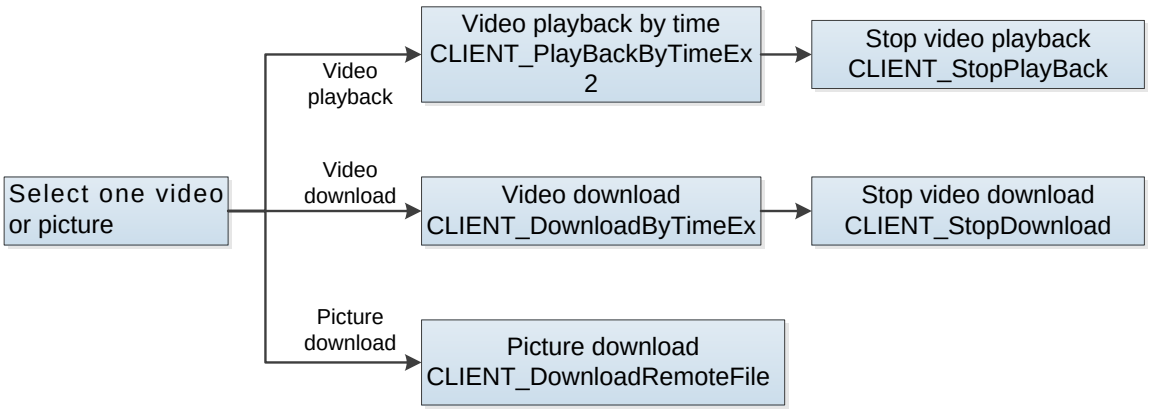
Step 8 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- The parameter pQueryCondition of **CLIENT_FindFileEx** is requested and released by the user. The specific type is defined by the enumeration type of emType.
- If **CLIENT_FindFileEx** sucessfully search, the searching handle will be returned. **CLIENT_FindNextFileEx** will take the searching handle as a parameter to search specific video and picture. You should call **CLIENT_FindCloseEx** to close the searching handle.
- Call **CLIENT_FindNextFileEx** to set the searching number. If the number is more than 1, then the parameter pMediaFileInfo should be taken as a data pointer.

2.5.3.2 Process of Playing Back and Downloading Video and Downloading Picture

Figure 2-7 Process of playing back and downloading video and downloading picture



Process Description

Select one result searched by **CLIENT_FindNextFileEx**, and then download or playback the result.

- Playing back the video

Step 1 If is video file, use start time and end time in the video searching result, call **CLIENT_PlayBackByTimeEx2** to playback the video.

Step 2 During the playback process or after playing back, call **CLIENT_StopPlayBack** to stop playing back the video.

- Download video

Step 1 If is video file, use start time and end time in the video searching result, call **CLIENT_DownloadByTimeEx** to download the video.

Step 2 After downloading, call **CLIENT_StopDownload** to stop downloading the video.

- Download the picture

If is picture file, use file name and picture type in the picture searching result, call CLIENT_DownloadRemoteFile to download the picture.

Notes for Process

The video playing back and downloading and picture downloading are all relied on the searching result of video and picture, and then you can take the result as the condition of playing back and downloading.

2.5.4 Example Code

2.5.4.1 Searching for Video and Picture

```
// Searching conditions
MEDIAFILE_FACE_DETECTION_PARAM param;
memset(&param, 0, sizeof(param));
param.dwSize = sizeof(param);
param.stuDetail.dwSize = sizeof(MEDIAFILE_FACE_DETECTION_DETAIL_PARAM);
param.nChannelID = -1;
param.stuStartTime = startTime;
param.stuEndTime = endTime
param.emPicType = NET_FACEPIC_TYPE_SMALL; // The small picture of people face.
param.bDetailEnable = FALSE;
param.emSex = EM_DEV_EVENT_FACEDETECT_SEX_TYPE_MAN;
param.bAgeEnable = FALSE;
param.nEmotionValidNum = 0;
param.emGlasses = EM_FACEDETECT_WITH_GLASSES;

// Search the small picture of face detection
LLONG lFindFileHandle = CLIENT_FindFileEx(g_lLoginHandle, DH_FILE_QUERY_FACE_DETECTION, &param, NULL,5000);
if (lFindFileHandle == 0)
{
    printf("CLIENT_FindFileEx: failed! Error code: %x.\n", CLIENT_GetLastError());
    return;
}

// Get the number of people face that searched
BOOL nRet = CLIENT_GetTotalFileCount(lFindFileHandle,&nCount,NULL);
if (!nRet)
{
    printf("CLIENT_GetTotalFileCount: failed! Error code: %x.\n", CLIENT_GetLastError());
    return;
```

```
}

// Searching number
int nMaxConut = 10;
MEDIAFILE_FACE_DETECTION_INFO* pMediaFileInfo = NEW MEDIAFILE_FACE_DETECTION_INFO[nMaxConut];
memset (pMediaFileInfo, 0, sizeof (MEDIAFILE_FACE_DETECTION_INFO) * nMaxConut);
for (int i = 0; i < nMaxConut; i++)
{
pMediaFileInfo[i].dwSize = sizeof(MEDIAFILE_FACE_DETECTION_INFO);
}
// Start searching
int nRet = CLIENT_FindNextFileEx(lFindFileHandle, nMaxConut, (void*)pMediaFileInfo, nMaxConut * sizeof(MEDIAFILE_FACE_DETECTION_INFO), NULL,3000);
if (nRet < 0)
{
    printf("CLIENT_FindNextFileEx: failed! Error code: %x.\n", CLIENT_GetLastError());
    return;
}

Close searching
CLIENT_FindCloseEx(lFindFileHandle);
```

2.5.4.2 Playing Back the Video

```
// Set the stream type when the video is playing back, here set it as the main stream
int nStreamType = 0; // 0-main and sub stream, 1-main stream, 2-sub stream
CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_STREAM_TYPE, &nStreamType);
// Set the file type of the video when playing back, here set it as all video.
NET_RECORD_TYPE emFileType = NET_RECORD_TYPE_ALL; // All video
CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_TYPE, &emFileType);
/ / Start playing back the video
int nChannelID = 0; // Channel number.
NET_IN_PLAY_BACK_BY_TIME_INFO stIn = {0};
NET_OUT_PLAY_BACK_BY_TIME_INFO stOut = {0};
memcpy(&stIn.stStartTime, &stuStartTime, sizeof(stuStartTime));
memcpy(&stIn.stStopTime, &stuStopTime, sizeof(stuStopTime));
stIn.hWnd = hWnd;
stIn.fDownLoadDataCallBack = DataCallBack;
stIn.dwDataUser = NULL;
stIn.cbDownLoadPos = NULL;
stIn.dwPosUser = NULL;
```

```
stIn.nPlayDirection = emDirection;
stIn.nWaittime = 10000;
LLONG lPlayHandle = CLIENT_PlayBackByTimeEx2(lLoginHandle, nChannelID, &stIn, &stOut);
if (0 == lPlayHandle)
{
    printf("CLIENT_PlayBackByTimeEx2: failed! Error code: %x.\n", CLIENT_GetLastError());
}

if (FALSE == CLIENT_StopPlayBack(lPlayHandle))
{
    printf("CLIENT_StopPlayBack Failed, lRealHandle[%x]!Last Error[%x]\n" , lPlayHandle,
    CLIENT_GetLastError());
}
```

2.5.4.3 Downloading Video

```
// Playback progress function
void CALLBACK TimeDownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD
dwDownLoadSize, int index, NET_RECORDFILE_INFO recordfileinfo, LDWORD dwUser);

// Playback or download data callback function
int CALLBACK DataCallBack(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD
dwBufSize, LDWORD dwUser);

int main()
{
    // Set the video stream type when searching, here set it as the main and sub stream.
    int nStreamType = 0; // 0-main and sub stream, 1-main stream, 2-sub stream
    CLIENT_SetDeviceMode(lLoginHandle, DH_RECORD_STREAM_TYPE, &nStreamType);

    // Set the downloading start and end time
    int nChannelID = 0; // Channel number.

    NET_TIME stuStartTime = {0};
    stuStartTime.dwYear = 2018;
    stuStartTime.dwMonth = 9;
    stuStartTime.dwDay = 17;

    NET_TIME stuStopTime = {0};
    stuStopTime.dwYear = 2018;
```

```
    stuStopTime.dwMonth = 9;
    stuStopTime.dwDay = 18;

    Start downloading the video.
    // One of the formal parameters sSavedFileName and fDownLoadDataCallBack should be valid, otherwise the input parameter is wrong.
    lDownloadHandle = CLIENT_DownloadByTimeEx(lLoginHandle, nChannelID, EM_RECORD_TYPE_ALL, &stuStartTime, &stuStopTime, "test.dav", TimeDownLoadPosCallBack, NULL, DataCallBack, NULL);
    if (lDownloadHandle == 0)
    {
        printf("CLIENT_DownloadByTimeEx: failed! Error code: %x.\n", CLIENT_GetLastError());
    }

    // Close downloading. Call the function after or during the downloading.
    if (0 != lDownloadHandle)
    {
        if (!CLIENT_StopDownload(lDownloadHandle))
        {
            printf("CLIENT_StopDownload Failed, lDownloadHandle[%x]!Last Error[%x]\n" ,
            lDownloadHandle, CLIENT_GetLastError());
        }
    }
}

void CALLBACK TimeDownLoadPosCallBack(LLONG lPlayHandle, DWORD dwTotalSize, DWORD
dwDownLoadSize, int index, NET_RECORDFILE_INFO recordfileinfo, LDWORD dwUser)
{
    // You can deal with the progress callback function.
}

int CALLBACK DataCallBack(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD
dwBufSize, LDWORD dwUser)
{
    switch(dwDataType)
    {
    case 0:
        // Original data
        // You can save the stream data here. After leaving callback function, do the decoding and forwarding and so on.
        break;
```

```
        case 1://Standard video data
            break;
        case 2:    //yuv data
            break;
        case 3://pcm audio data
            break;
        default:
            break;
        }
        return 0;
}
```

2.5.4.4 Downloading the Picture

```
DH_IN_DOWNLOAD_REMOTE_FILE stuRemoteFileParm;
memset(&stuRemoteFileParm, 0, sizeof(DH_IN_DOWNLOAD_REMOTE_FILE));
stuRemoteFileParm.dwSize = sizeof(DH_IN_DOWNLOAD_REMOTE_FILE);
stuRemoteFileParm.pszFileName = pInfo->stObjectPic.szFilePath ;
stuRemoteFileParm.pszFileDst = szFileName;

DH_OUT_DOWNLOAD_REMOTE_FILE *fileinfo = NEW DH_OUT_DOWNLOAD_REMOTE_FILE;
fileinfo->dwSize = sizeof(DH_OUT_DOWNLOAD_REMOTE_FILE);

if (!CLIENT_DownloadRemoteFile(g_lLoginHandle, &stuRemoteFileParm, fileinfo))
{
        printf("CLIENT_DownloadRemoteFile Failed,Last Error[%x]\n" , CLIENT_GetLastError());
}
```

3 Face Detection and Recognition

3.1 Subscribing Face Event

About more details, see "2.4 Subscribing Intelligent Event". Call fAnalyzerDataCallBack to filter out face detection and recognition events, which are EVENT_IVS_FACEDETECT for people face detection events and EVENT_IVS_FACERECOGNITION for people target recognition events.

3.2 Adding/Deleting/Modifying/Searching the Face Library

3.2.1 Introduction

A face library includes face picture and face information, supports the adding, deleting, modifying and searching the face library function.

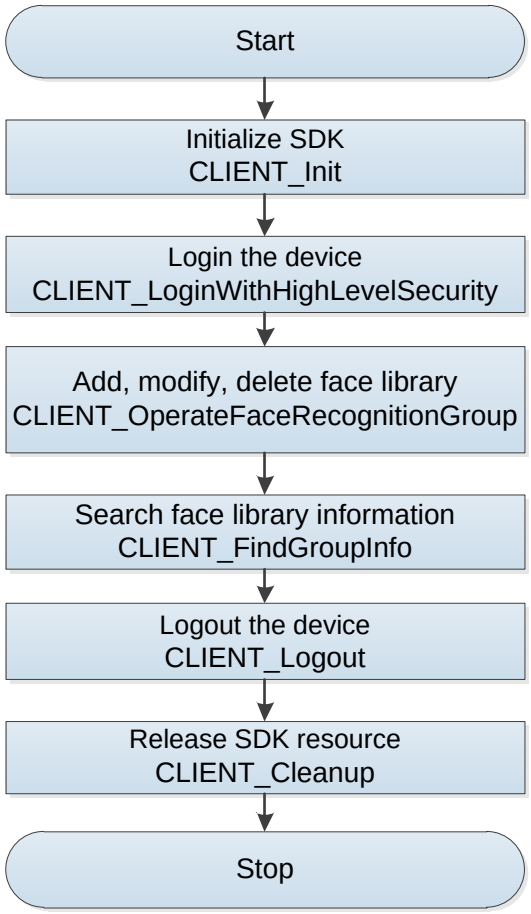
3.2.2 Interface Overview

Table 3-1 Interfaces of adding/deleting/modifying/searching the face library

Interface	Implication
CLIENT_OperateFaceRecognitionGroup	Add, delete and modify the face library.
CLIENT_FindGroupInfo	Search the information of face library.

3.2.3 Process

Figure 3-1 Process of face library operation



Process Description

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

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

Step 3 Call **CLIENT_OperateFaceRecognitionGroup** to add, modify and delete the face library according to enumeration type.

Step 4 Call **CLIENT_FindGroupInfo** to get the information of face library.

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

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

Notes for Process

- Adding face library: The corresponding value of operation type emOperateType is NET_FACERECONGNITION_GROUP_ADD, the corresponding structure is NET_ADD_FACERECONGNITION_GROUP_INFO.
- Modify face library: The corresponding value of operation type emOperateType is NET_FACERECONGNITION_GROUP_MODIFY, the corresponding structure is NET_MODIFY_FACERECONGNITION_GROUP_INFO. You need to specify GroupID and the face library type emFaceDBType when modifying face library. The specified GroupID should be exists in the device.

- Deleting face library: The corresponding value of operation type emOperateType is NET_FACERECONGNITION_GROUP_DELETE, the corresponding structure is NET_DELETE_FACERECONGNITION_GROUP_INFO. If you specify GroupID when deleting face library, the corresponding GroupID of face library is deleted; If you do not specify GroupID, all of the face libraries are deleted.

3.2.4 Example Code

3.2.4.1 Searching for Face Library Information

```
// Set the searching conditions of face library
NET_IN_FIND_GROUP_INFO stuInParam = {sizeof(stuInParam)};
NET_OUT_FIND_GROUP_INFO stuOutParam = {sizeof(stuOutParam)};
stuOutParam.nMaxGroupNum = 100;
NET_FACERECONGNITION_GROUP_INFO *pGroupInfo = NULL;
stuOutParam.pGroupInfos = new NET_FACERECONGNITION_GROUP_INFO[100];
memset(stuOutParam.pGroupInfos, 0,sizeof(NET_FACERECONGNITION_GROUP_INFO)*100);
for (int i = 0; i < 100; i++)
{
stuOutParam.pGroupInfos[i].dwSize = sizeof (FACERECONGNITION_GROUP_INFO);
}
// Search the face library
BOOL bRet = CLIENT_FindGroupInfo(lLoginHandle, &stuInParam, &stuOutParam, 5000);
if(FALSE == bRet)
{
    printf("CLIENT_FindGroupInfo: failed! Error code %x.\n", CLIENT_GetLastError());
    return -1;
}
delete[] pGroupInfo;
```

3.2.4.2 Adding/Deleting/Modifying the Face Library

```
enum EM_OPERATION_TYPE
{
    FACEDB_DELETE,    // Delete
    FACEDB_ADD,       // Add
    FACEDB_MODIFY     // Modify
};
// Set the face library ID for deleting.
NET_FACERECONGNITION_GROUP_INFO *pstGroupInfo = m_pstSelectGroup;
NET_IN_OPERATE_FACERECONGNITION_GROUP stuInParam = {sizeof(stuInParam)};
```

```
NET_OUT_OPERATE_FACERECONGNITION_GROUP stuOutParam = {sizeof(stuOutParam)};
NET_ADD_FACERECONGNITION_GROUP_INFO stuAddGroupInfo = {sizeof(stuAddGroupInfo)};
NET_MODIFY_FACERECONGNITION_GROUP_INFO stuEditGroupInfo = {sizeof(stuEditGroupInfo)};

EM_OPERATION_TYPE emType = mType;
switch(emType)
{
    // Delete the face library
    case FACEDB_DELETE:
    {
        stuInParam.emOperateType = NET_FACERECONGNITION_GROUP_DELETE;
        NET_DELETE_FACERECONGNITION_GROUP_INFO stuDeleteInfo;
        memset(&stuDeleteInfo, 0, sizeof(stuDeleteInfo));
        stuDeleteInfo.dwSize = {sizeof(stuDeleteInfo)};
        strncpy(stuDeleteInfo.szGroupId, pstGroupInfo->szGroupId, sizeof(stuDeleteInfo.szGroupId)-1);
        stuInParam.pOPerateInfo = &stuDeleteInfo;
        break;
        }
        // Add the face library
        case FACEDB_ADD:
        {
        stuInParam.emOperateType = NET_FACERECONGNITION_GROUP_ADD;
        stuAddGroupInfo.stuGroupInfo.dwSize = sizeof(stuAddGroupInfo.stuGroupInfo);
        stuAddGroupInfo.stuGroupInfo.emFaceDBType = NET_FACE_DB_TYPE_BLACKLIST;

        strncpy(stuAddGroupInfo.stuGroupInfo.szGroupName,pcGroupName,sizeof(stuAddGroupInfo.stuGr
oupInfo.szGroupName)-1);
        stuInParam.pOPerateInfo = &stuAddGroupInfo;
        break;
        }
        Modifying the face library
        case FACEDB_MODIFY:
        {
        stuInParam.emOperateType = NET_FACERECONGNITION_GROUP_MODIFY;
        stuEditGroupInfo.stuGroupInfo.dwSize = sizeof(stuEditGroupInfo.stuGroupInfo);
        stuEditGroupInfo.stuGroupInfo.emFaceDBType = NET_FACE_DB_TYPE_BLACKLIST;
        strncpy(stuEditGroupInfo.stuGroupInfo.szGroupName,                    pcGroupName,
sizeof(stuEditGroupInfo.stuGroupInfo.szGroupName)-1);
        strncpy(stuEditGroupInfo.stuGroupInfo.szGroupId,                m_stuGroupInfo.szGroupId,
sizeof(stuEditGroupInfo.stuGroupInfo.szGroupId)-1);
```

```
            stuInParam.pOPerateInfo = &stuEditGroupInfo;
            break;
            }
            default:
            break;
}
BOOL bRet = CLIENT_OperateFaceRecognitionGroup(m_lLoginID, &stuInParam, &stuOutParam, 5000);
if(FALSE == bRet)
{
    printf("CLIENT_OperateFaceRecognition: failed! Error code %x.\n", CLIENT_GetLastError());
    return -1;
}
```

3.3 Adding/Deleting/Modifying/Searching People Face

3.3.1 Introduction

The face library includes face information. This function supports adding, deleting, modifying and searching people face information.

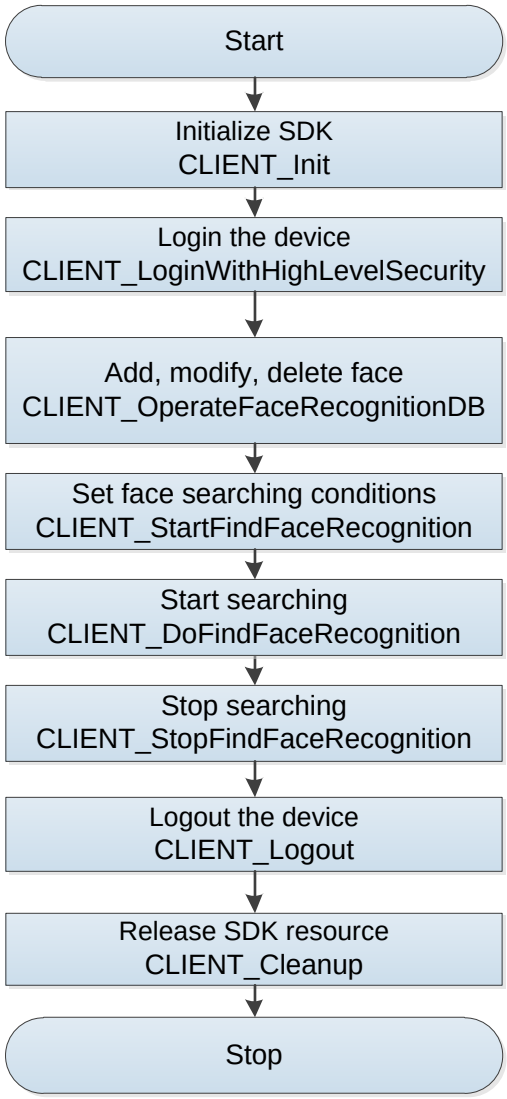
3.3.2 Interface Overview

Table 3-2 Interfaces of adding/deleting/modifying/searching people face

Interface	Implication
CLIENT_OperateFaceRecognitionDB	Add, delete and modify the people face.
CLIENT_StartFindFaceRecognition	Set the searching conditions of people face.
CLIENT_DoFindFaceRecognition	Search the face data of specified number.
CLIENT_StopFindFaceRecognition	Stop searching.

3.3.3 Process

Figure 3-2 Process of adding/deleting/modifying/searching people face



Process Description

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

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

Step 3 Call **CLIENT_OperateFaceRecognitionDB** to add, modify and delete the face library according to enumeration type.

Step 4 Call **CLIENT_StartFindFaceRecognition** to set the searching conditions of people face.

Step 5 Call **CLIENT_DoFindFaceRecognition** to get the searching result.

Step 6 Call **CLIENT_StopFindFaceRecognition** to stop searching.

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

- Adding people face to the library: The corresponding value of operation type emOperateType is NET_FACERECONGNITIONDB_ADD. The input parameter in the structure, pBuffer can save

people face you need. You can request and release the resource by yourself. GroupID must be filled-in. If people add successful, the device returns UID, is also called the only people ID, means the only people. See pstOutParam -> szUID.

- Modify the people face in the library: The corresponding value of operation type emOperateType is NET_FACERECONGNITIONDB_MODIFY, The input parameter in the structure pBuffer can save people face you need. You can request and release the resource by yourself. The GroupID and szUID must be filled-in in the stPersonInfo.
- Delete the people face in the library: The corresponding value of operation type emOperateType is NET_FACERECONGNITIONDB_DELETE. The GroupID and szUID must be filled-in in the stPersonInfo.

3.3.4 Example Code

3.3.4.1 Adding/Deleting/Modifying the People Face

```
// Add and modify the people
NET_IN_OPERATE_FACERECONGNITIONDB stuInParam = { sizeof(stuInParam) };
NET_OUT_OPERATE_FACERECONGNITIONDB stuOutParam = { sizeof(stuOutParam) };
// Add the information of people
{
    stuInParam.emOperateType = NET_FACERECONGNITIONDB_ADD;
}

// Modify the information of people
{
    //stuInParam.emOperateType = NET_FACERECONGNITIONDB_MODIFY;
    //strncpy(stuInParam.stPersonInfoEx.szUID, strUID, sizeof(stuInParam.stPersonInfoEx.szUID) - 1);
}

stuInParam.bUsePersonInfoEx = TRUE;
stuInParam.stPersonInfoEx.bySex = 1;    // Male
stuInParam.stPersonInfoEx.byIDType = 1;    // ID card
stuInParam.stPersonInfoEx.wYear = time.GetYear();
stuInParam.stPersonInfoEx.byMonth = time.GetMonth();
stuInParam.stPersonInfoEx.byDay = time.GetDay();
strncpy(stuInParam.stPersonInfoEx.szPersonName, pstrName,sizeof(stuInParam.stPersonInfoEx.szPersonName)
- 1);
strncpy(stuInParam.stPersonInfoEx.szID, pstrCardID, sizeof(stuInParam.stPersonInfoEx.szID) - 1);
strncpy(stuInParam.stPersonInfoEx.szGroupName,                                    m_szGroupName,
sizeof(stuInParam.stPersonInfoEx.szGroupName) - 1);
strncpy(stuInParam.stPersonInfoEx.szGroupID, m_szGroupId, sizeof(stuInParam.stPersonInfoEx.szGroupID) - 1);
stuInParam.nBufferLen = nPictureBufferLen;
stuInParam.pBuffer = pPictureBuffer;
stuInParam.stPersonInfoEx.wFacePicNum = 1;
stuInParam.stPersonInfoEx.szFacePicInfo[0].dwOffSet = 0;
stuInParam.stPersonInfoEx.szFacePicInfo[0].dwFileLenth = nLength;
```

```
bRet = CLIENT_OperateFaceRecognitionDB(m_lLoginID, &stuInParam, &stuOutParam, 5000);
if (FACE_PERSON_ADD == m_nOpreateType)
{
    printf("CLIENT_OperateFaceRecognitionDB failed! Error code %x.\n", CLIENT_GetLastError());
    return -1;
}


// Delete the information of people
NET_IN_OPERATE_FACERECONGNITIONDB stuInParam = { sizeof(stuInParam) };
NET_OUT_OPERATE_FACERECONGNITIONDB stuOutParam = { sizeof(stuOutParam) };
// Only need szGroupID and szUID.
stuInParam.emOperateType = NET_FACERECONGNITIONDB_DELETE;
stuInParam.bUsePersonInfoEx = TRUE;
strncpy(stuInParam.stPersonInfoEx.szUID,
m_pstPersonSelectInfo->stuCandidate.stPersonInfo.szUID,sizeof(stuInParam.stPersonInfoEx.szUID) - 1);
strncpy(stuInParam.stPersonInfoEx.szGroupID, m_szGroupId, sizeof(stuInParam.stPersonInfoEx.szGroupID) - 1);

BOOL bRet = CLIENT_OperateFaceRecognitionDB(lLoginHandle, &stuInParam, &stuOutParam, 5000);
if (!bRet)
{
    printf("CLIENT_OperateFaceRecognitionDB failed! Error code %x.\n",CLIENT_GetLastError());
    return -1;
}
```

3.3.4.2 Searching the People Face

```
// Set searching conditions of people face.
NET_IN_STARTFIND_FACERECONGNITION stuInParam = {sizeof(stuInParam)};
NET_OUT_STARTFIND_FACERECONGNITION stuOutParam = {sizeof(stuOutParam)};

stuInParam.stMatchOptions.dwSize = sizeof(stuInParam.stMatchOptions);
stuInParam.stFilterInfo.dwSize = sizeof(stuInParam.stFilterInfo);
stuInParam.bPersonExEnable = TRUE;
stuInParam.stFilterInfo.nRangeNum = 1;
stuInParam.stFilterInfo.szRange[0] = (BYTE)NET_FACE_DB_TYPE_BLACKLIST;
strncpy(stuInParam.stPersonInfoEx.szPersonName, m_PersonName,
sizeof(stuInParam.stPersonInfoEx.szPersonName)-1);
stuInParam.stPersonInfoEx.bySex = 0;
stuInParam.stFilterInfo.stBirthdayRangeStart = BirthdayRangeStart;
stuInParam.stFilterInfo.stBirthdayRangeEnd = BirthdayRangeEnd;
strncpy(stuInParam.stPersonInfoEx.szID, pcCard, sizeof(stuInParam.stPersonInfoEx.szID)-1);
strncpy(stuInParam.stFilterInfo.szGroupId[0], m_szGroupId, sizeof(m_szGroupId)-1);
```

```
stuInParam.stFilterInfo.nGroupIdNum = 1;
strncpy(stuInParam.stPersonInfoEx.szGroupID, m_szGroupId, sizeof(stuInParam.stPersonInfoEx.szGroupID)-1);

BOOL bRet = CLIENT_StartFindFaceRecognition(m_lLoginID, &stuInParam, &stuOutParam, DEFAULT_WAIT_TIME);
if (!bRet)
{
    printf("CLIENT_StartFindFaceRecognition failed! Error code %x.\n", CLIENT_GetLastError());
    return;
}

// Start searching
NET_IN_DOFIND_FACERECONGNITION stuInDoFind = {sizeof(stuInDoFind)};
NET_OUT_DOFIND_FACERECONGNITION stuOutDoFind = {sizeof(stuOutDoFind)};
stuOutDoFind.bUseCandidatesEx = TRUE;

stuInDoFind.lFindHandle = m_lFindPersonHandle;
stuInDoFind.emDataType = EM_NEEDED_PIC_TYPE_HTTP_URL;
stuInDoFind.nCount = 10;
stuInDoFind.nBeginNum = m_nCurPos;
bRet = CLIENT_DoFindFaceRecognition(&stuInDoFind, &stuOutDoFind, WAIT_TIMEOUT);
if (!bRet)
{
    printf("CLIENT_DoFindFaceRecognition failed! Error code %x.\n", CLIENT_GetLastError());
    return;
}
```

3.4 Arming by Channel or Library

3.4.1 Introduction

Arm by channel, means one channel arm one or multiple face libraries.

Arm by library, means one face library arm one or multiple channels.

These two ways are all arms of face library.

3.4.2 Interface Overview

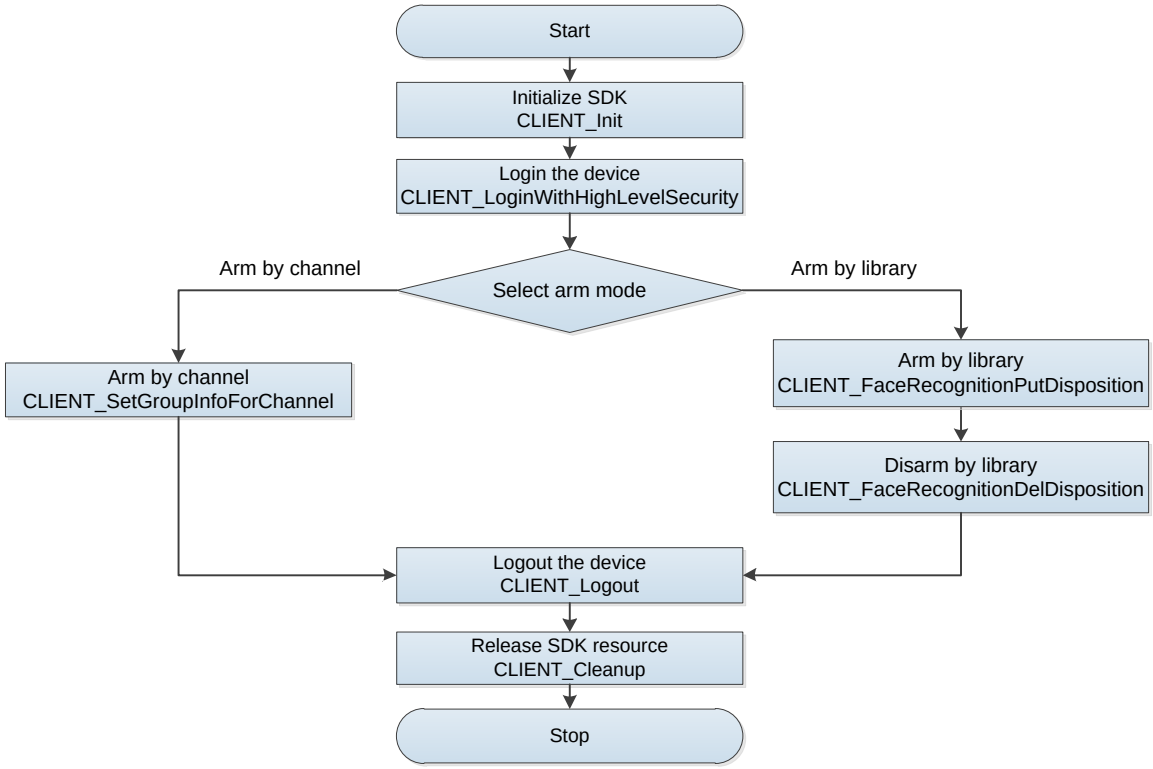
Table 3-3 Interfaces of arming by channel or library

Interface	Implication
CLIENT_FaceRecognitionPutDisposition	Arm by library.

Interface	Implication
CLIENT_FaceRecognitionDelDisposition	Disarm by library.
CLIENT_SetGroupInfoForChannel	Arm by channel.

3.4.3 Process

Figure 3-3 Process of arming by channel or library



Process Description

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

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

Step 3 Select arm way.

- Arm by library

1) Select arm by library, call **CLIENT_FaceRecognitionPutDisposition** to arm the library.
2) After using the function module, call **CLIENT_FaceRecognitionDelDisposition** to disarm the library.

- Arm by channel

  Select arm by channel, call **CLIENT_SetGroupInfoForChannel** to arm the channel.

Step 4 After using the function module, call **CLIENT_Logout** to logout the device.

Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Arm by channel and arm by library are two ways of arming the face library.
- Arm by channel, means one channel arm multiple face libraries. Arm by library, means multiple channel arm one face libraries.

- When arm by channel, call **CLIENT_SetGroupInfoForChannel** to cover the old configurations. Disarm by channel, sent the empty arm information.
- When arm by library, call **CLIENT_FaceRecognitionDelDisposition** to disarm some channels. For example, 3 channels are armed, you can disarm 2 channels, and the other one stay the same.

3.4.4 Example Code

3.4.4.1 Arm by channel

```
// Input parameter
NET_IN_SET_GROUPINFO_FOR_CHANNEL stInChannelDeploy = { sizeof(NET_IN_SET_GROUPINFO_FOR_CHANNEL)};
stInChannelDeploy.nChannelID = 0;
stInChannelDeploy.nGroupIdNum = 2; // Set the face library number at this channel.
strncpy(stInChannelDeploy.szGroupId[0], strGroupId1, DH_COMMON_STRING_64-1);  // Copy the face library ID.
strncpy(stInChannelDeploy.szGroupId[1], strGroupId2, DH_COMMON_STRING_64-1);
stInChannelDeploy.nSimilaryNum = 2；// The similarity value number, which is equal to people groups.
stInChannelDeploy.nSimilary[0] = 85; // The similarity value at the first face library.
stInChannelDeploy.nSimilary[1] = 90; // The similarity value at the second face library.

// Output parameter
NET_OUT_SET_GROUPINFO_FOR_CHANNEL stOutChannelDeploy = { sizeof(NET_OUT_SET_GROUPINFO_FOR_CHANNEL)};

/ / Arm by library
BOOL bRet = CLIENT_SetGroupInfoForChannel(lLoginHandle, &stInChannelDeploy, &stOutChannelDeploy);
if (flase == bRet)
{
    printf("CLIENT_SetGroupInfoForChannel: failed! Error code: %x.\n", CLIENT_GetLastError());
}

/ / Disarm by channel, sent the empty arm information.
if (NULL != lRealHandle)
{
    memset(stInChannelDeploy, 0, sizeof(NET_IN_SET_GROUPINFO_FOR_CHANNEL));
    memset(stOutChannelDeploy, 0, sizeof(NET_OUT_SET_GROUPINFO_FOR_CHANNEL));
    stInChannelDeploy.dwSize = sizeof(NET_IN_SET_GROUPINFO_FOR_CHANNEL);
    stOutChannelDeploy.dwSize = sizeof(NET_OUT_SET_GROUPINFO_FOR_CHANNEL);
    CLIENT_SetGroupInfoForChannel(lLoginHandle, &stInChannelDeploy, &stOutChannelDeploy);
}
```

3.4.4.2 Arm by library

```
// Input parameter
NET_IN_FACE_RECOGNITION_PUT_DISPOSITION_INFO stInFaceRecognitionDeploy = { sizeof(NET_IN_FACE_RECOGNITION_PUT_DISPOSITION_INFO};
strncpy(stInFaceRecognitionDeploy.szGroupId, strGroupId, DH_COMMON_STRING_64-1); // Face library that need to arm
stInFaceRecognitionDeploy.nDispositionChnNum = 2; // Number of video channel that armed
stInFaceRecognitionDeploy.stuDispositionChnInfo[0].nChannelID = 0;      // Face library deploy channel.
stInFaceRecognitionDeploy.stuDispositionChnInfo[0].nSimilary = 90;      // Similarity value.
stInFaceRecognitionDeploy.stuDispositionChnInfo[1].nChannelID = 2;
stInFaceRecognitionDeploy.stuDispositionChnInfo[1].nSimilary = 85;

// Output parameter
NET_OUT_FACE_RECOGNITION_PUT_DISPOSITION_INFO stOutFaceRecognitionDeploy = {sizeof(NET_OUT_FACE_RECOGNITION_PUT_DISPOSITION_INFO)};

/ / Arm by face library
bool nRet = CLIENT_FaceRecognitionPutDisposition(lLoginHandle, &stInFaceRecognitionDeploy, &stOutFaceRecognitionDeploy);
if(false = nRet)
{
        printf("CLIENT_FaceRecognitionPutDisposition: failed! Error code: %x.\n", CLIENT_GetLastError());
}

// Disarm input parameter, you can disarm some channels.
NET_IN_FACE_RECOGNITION_DEL_DISPOSITION_INFO stInFaceRecognitionDel = {sizeof(NET_IN_FACE_RECOGNITION_DEL_DISPOSITION_INFO)};
stInFaceRecognitionDel
strncpy(stInFaceRecognitionDel.szGroupId, strGroupId, DH_COMMON_STRING_64-1);
stInFaceRecognitionDel.nDispositionChnNum = 2; // Number of channel that armed
stInFaceRecognitionDel.nDispositionChn[0] = 0;    // Disarm channel 1
stInFaceRecognitionDel.nDispositionChn[1] = 1;

// Disarm output parameter
NET_OUT_FACE_RECOGNITION_DEL_DISPOSITION_INFO stOutFaceRecognitionDel = {sizeof(NET_OUT_FACE_RECOGNITION_DEL_DISPOSITION_INFO)};

bool nRet = CLIENT_FaceRecognitionDelDisposition(lLoginHandle, &stInFaceRecognitionDel, &stOutFaceRecognitionDel);
if(false = nRet)
```

```
{
        printf("CLIENT_FaceRecognitionDelDisposition: failed! Error code: %x.\n", CLIENT_GetLastError());
}
```

3.5 Searching Picture by Picture

3.5.1 Introduction

You can import a picture and a similarity value, and then the IVSS and NVR devices will search the history library and the face library by this picture to make sure whether there is the matched face in the two libraries. And then it will return the right picture.

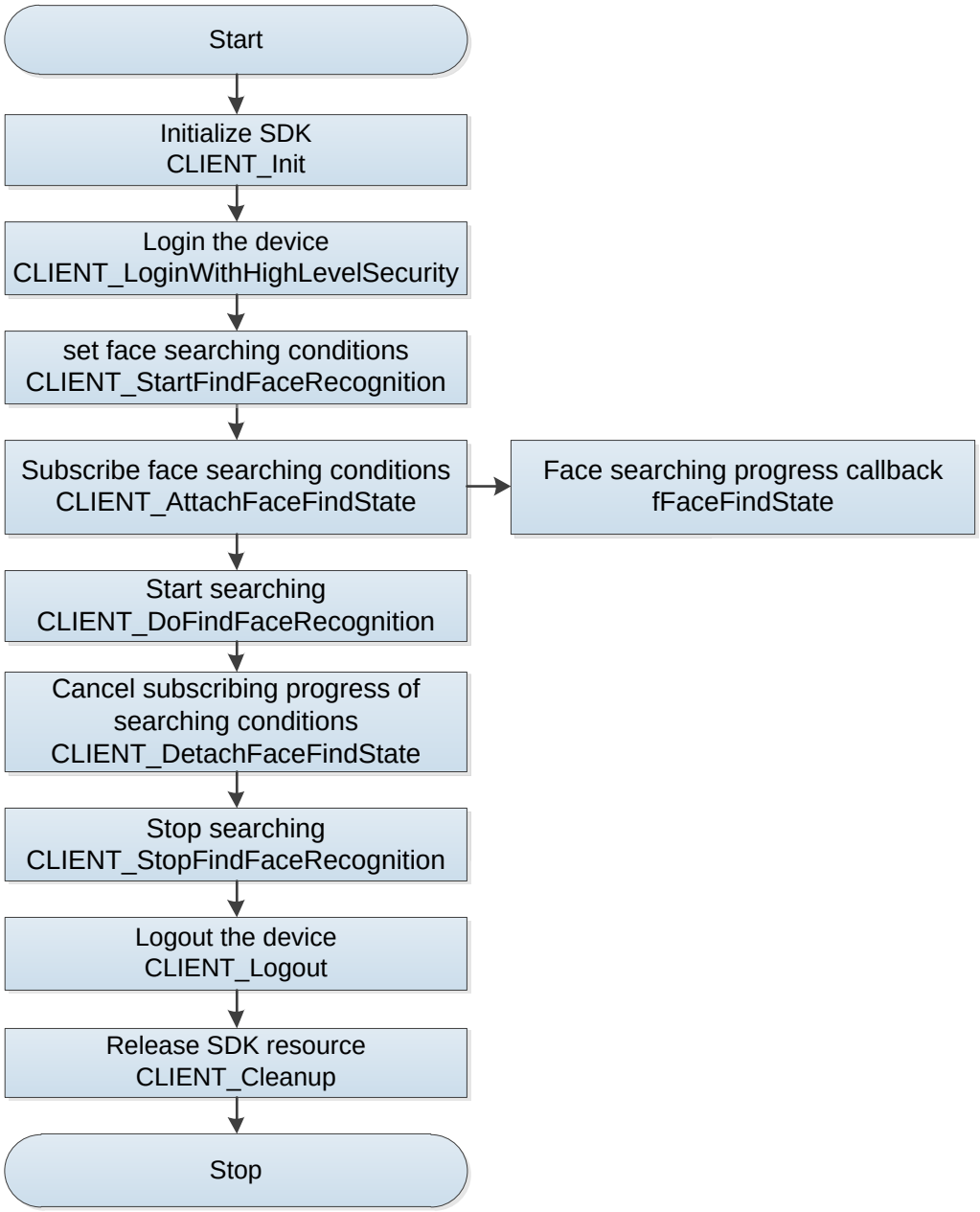
3.5.2 Interface Overview

Table 3-4 Interfaces of searching picture by picture

Interface	Implication
CLIENT_StartFindFaceRecognition	Set the searching conditions of people face.
CLIENT_AttachFaceFindState	Subscribe searching conditions of people face.
CLIENT_DetachFaceFindState	Cancel subscribing the progress of searching conditions.
CLIENT_DoFindFaceRecognition	Start searching.
CLIENT_StopFindFaceRecognition	Stop searching.

3.5.3 Process

Figure 3-4 Process of searching picture by picture



Process Description

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

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

Step 3 Call **CLIENT_StartFindFaceRecognition** to set the searching conditions of people face.

Step 4 Check the return value in the Step3, If the nTotalCount in the output parameter structure returns -1, you need to wait until device searching complete.

Step 5 Call **CLIENT_AttachFaceFindState** to subscribe the status of people face searching. Wait until the return progress of the progress callback function is 100, the searching is complete. Call **CLIENT_DetachFaceFindState** to cancel subscribing the searching progress.

Step 6 Call **CLIENT_DoFindFaceRecognition** to get the searching result.

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

Step 8 After using the function module, call **CLIENT_Logout** to logout the device.

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

3.5.4 Example Code

```
// Search progress callback function
void CALLBACK FaceFindState(LLONG lLoginID, LLONG lAttachHandle, NET_CB_FACE_FIND_STATE* pstStates,
int nStateNum, LDWORD dwUser)
{
    if (pstStates->nProgress== 100) // Means the searching progress is 100%.
    {
        //Stop subscribe the progress of people face searching
        CLIENT_DetachFaceFindState(lAttachHandle);
        // Start searching
        DoFind();
        }
        return;
}

// Configure searching conditions
NET_IN_STARTFIND_FACERECONGNITION stuInParam = { sizeof(stuInParam) };
NET_OUT_STARTFIND_FACERECONGNITION stuOutParam = { sizeof(stuOutParam) };
stuInParam.stFilterInfo.dwSize = sizeof(stuInParam.stFilterInfo);
stuInParam.stMatchOptions.dwSize = sizeof(stuInParam.stMatchOptions);
stuInParam.bPersonExEnable = TRUE;
stuInParam.nChannelID = 0;
stuInParam.stMatchOptions.nSimilarity = 80;
stuInParam.stFilterInfo.stStartTime = startTime;
stuInParam.stFilterInfo.stEndTime = endTime;
stuInParam.nBufferLen = nPicBufLen;
stuInParam.pBuffer = strPicBuf;    // Picture Buffer
stuInParam.stPersonInfoEx.wFacePicNum = 1;
stuInParam.stPersonInfoEx.szFacePicInfo[0].dwOffSet = 0;
stuInParam.stPersonInfoEx.szFacePicInfo[0].dwFileLenth = nLength;

BOOL bRet = CLIENT_StartFindFaceRecognition(m_lLoginId, &stuInParam, &stuOutParam, 5000);
if (!bRet)
{
    printf("CLIENT_StartFindFaceRecognition: failed! Error code %x.\n", CLIENT_GetLastError());
    return -1;
}
```

```
m_lFindHandle = stuOutParam.lFindHandle;

if (-1 == stuOutParam.nTotalCount)
{
    // When the total searching number is -1, it means the device searching is not completed. You need to subscribe the progress of device searching.
    NET_IN_FACE_FIND_STATE stuInFindState = { sizeof(stuInFindState) };
    NET_OUT_FACE_FIND_STATE stuOutFindState = { sizeof(stuOutFindState) };
    stuInFindState.nTokenNum = 1;
    int nToken = stuOutParam.nToken;
    stuInFindState.nTokens = &nToken;
    stuInFindState.cbFaceFindState = FaceFindState; // Progress callback function.
    stuInFindState.dwUser = (DWORD)this;
    m_lAttachHandle = CLIENT_AttachFaceFindState(m_lLoginId, &stuInFindState, &stuOutFindState, 5000);
}
else
{
    // Start searching.
    DoFind();
}

void DoFind()
{
    NET_IN_DOFIND_FACERECONGNITION stuInDoFind = { sizeof(NET_IN_DOFIND_FACERECONGNITION) };
    NET_OUT_DOFIND_FACERECONGNITION stuOutDoFind = { sizeof(NET_OUT_DOFIND_FACERECONGNITION) };
    stuOutDoFind.bUseCandidatesEx = TRUE;
    stuInDoFind.nCount = 20; // Search for 20 information one time, the total searching number is more than 20.
    stuInDoFind.lFindHandle = m_lFindHandle;
    stuInDoFind.emDataType = EM_NEEDED_PIC_TYPE_HTTP_URL;// The returned picture format by specified searching result is http link.
    stuInDoFind.nBeginNum = 0;    // Search from 0.

    BOOL bRet = CLIENT_DoFindFaceRecognition(&stuInDoFind, &stuOutDoFind, 10000);
    if (!bRet)
    {
        printf("CLIENT_DoFindFaceRecognition: failed! Error code %x.\n", CLIENT_GetLastError());
        return ;
        }
```

```
    CLIENT_StopFindFaceRecognition(m_lFindHandle);
}
```

3.6 Searching and Downloading Face Video and Picture

The face intelligent event is one of the intelligent events, so for more details, See "2.5Searching/Playbacking/Downloading Video and Picture".

Call CLIENT_FindFileEx to set searching conditions. The following is about the face searching operation and the value of parameter emType.

- DH_FILE_QUERY_FACE: Search target recognition picture
- DH_FILE_QUERY_FACE_DETECTION: Search face detection picture
- DH_FILE_QUERY_FILE: Search video

The structure pointer of ET_IN_MEDIA_QUERY_FILE, the nEventLists field in the structure is as the following:

- EVENT_IVS_FACERECOGNITION: Target recognition video searching
- EVENT_IVS_FACEDETECT: Face detection video searching

4 Body Detection

4.1 Subscribing Body Event

About more details, see "2.4 Subscribing Intelligent Event". Call fAnalyzerDataCallBack to filter out body detection, which is EVENT_IVS_HUMANTRAIT for body detection events.

4.2 Searching the Body Picture

4.2.1 Introduction

For the code of body detection picture searching, see "2.5 Searching/Playbacking/Downloading Video and Picture". The following section shows the description for body detection picture downloading.

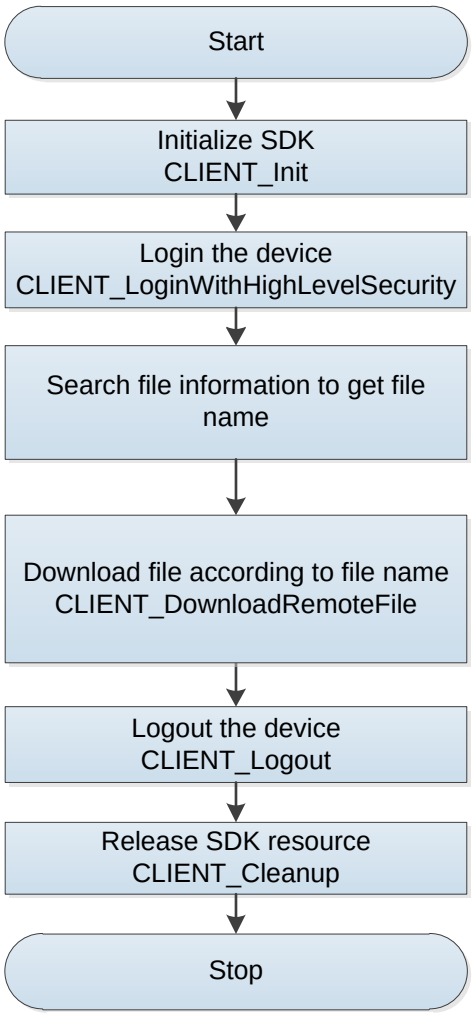
4.2.2 Interface Overview

Table 4-1 Interface of body picture searching

Interface	Implication
CLIENT_DownloadRemoteFile	Download file.

4.2.3 Process

Figure 4-1 Process of body picture searching



Process Description

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

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

Step 3 Call SDK interface to search file information and to get the file name.

Step 4 Using the searched file information, call **CLIENT_DownloadRemoteFile** to download file.

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

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

4.2.4 Example Code

```
DH_IN_DOWNLOAD_REMOTE_FILE stuInDownloadFile    = {sizeof(DH_IN_DOWNLOAD_REMOTE_FILE    )};
stuInDownloadFile.pszFileName = strFileName
stuInDownloadFile.pszFileDst = strDownloadName;

DH_OUT_DOWNLOAD_REMOTE_FILE stuOutDownloadFile = {sizeof(DH_OUT_DOWNLOAD_REMOTE_FILE )};
BOOL bRet = CLIENT_DownloadRemoteFile(m_lLoginHandle, &stuInDownloadFile, &stuOutDownloadFile);
```

```
if (bRet == FALSE)
{
	MessageBox(ConvertString("Download Failed"), ConvertString("Prompt"));
	return;
}
```

5 People Flow Statistics

5.1 Subscribing People Flow Event

5.1.1 Introduction

This is the real-time subscribe of flow statistics data function.

You can install the front-end devices in the specified areas to precise statistics the in and out number of people real-time in each entrance by the intelligent analysis server according to video data collected by the front-end devices.

You can also get the total in and out number of people in one single day and real-time in and out number of people.

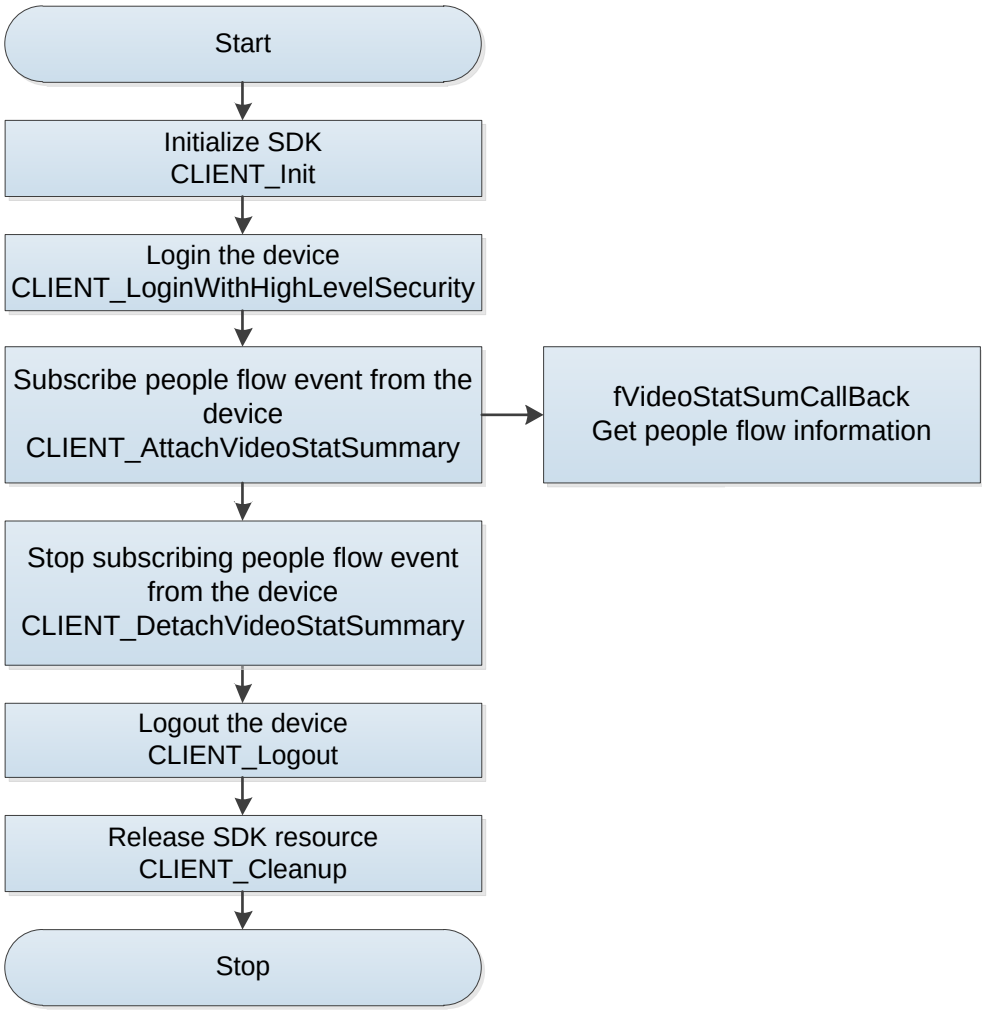
5.1.2 Interface Overview

Table 5-1 Interfaces of subscribing people flow

Interface	Implication
CLIENT_AttachVideoStatSummary	Subscribe people flow event.
CLIENT_DetachVideoStatSummary	Cancel subscribing people flow event.

5.1.3 Process

Figure 5-1 Process of subscribing people flow



Process Description

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

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

Step 3 Call **CLIENT_AttachVideoStatSummary** to subscribe people flow events from the device.

Step 4 After successful subscribe, call fVideoStatSumCallBack to get the face events and notify users.

Step 5 After using the flow statistics event function, call **CLIENT_DetachVideoStatSummary** to stop subscribing people flow events.

Step 6 After using the function module, call **CLIENT_Logout** to logout the device.

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

5.1.4 Example Code

```
void_CALLBACK VideoStatSumCallback(LLONG lAttachHandle, NET_VIDEOSTAT_SUMMARY* pBuf, DWORD dwBufLen, LDWORD dwUser)
{
```

```
    // Produce callback data
}

NET_IN_ATTACH_VIDEOSTAT_SUM InParam = {sizeof(NET_IN_ATTACH_VIDEOSTAT_SUM)};
NET_OUT_ATTACH_VIDEOSTAT_SUM OutParam = {sizeof(NET_OUT_ATTACH_VIDEOSTAT_SUM)};
InParam.nChannel=0;
InParam.cbVideoStatSum=VideoStatSumCallback;    // Subscribe callback function.

/ / Subscribe people flow statistics
LLONG attachHnd = CLIENT_AttachVideoStatSummary(lLoginID,&InParam,&OutParam,5000)
if(0 == attachHnd)
{
    printf("CLIENT_AttachVideoStatSummary failed! Error code %x.\n", CLIENT_GetLastError());
    return;
}

/ / Cancel subscribing people flow statistics
CLIENT_DetachVideoStatSummary(attachHnd);
```

5.2 Alarm of People Flow Event

About more details, see "2.4 Subscribing Intelligent Event". Call fAnalyzerDataCallBack to filter out people flow event, which is EVENT_IVS_NUMBERSTAT for people counting events and EVENT_IVS_MAN_NUM_DETECTION for area people counting events.

5.3 Searching History Data of People Flow Statistics

5.3.1 Introduction

You can specify the start time and the end time of people flow information, and then the device end will send back the searching data to the SDK.

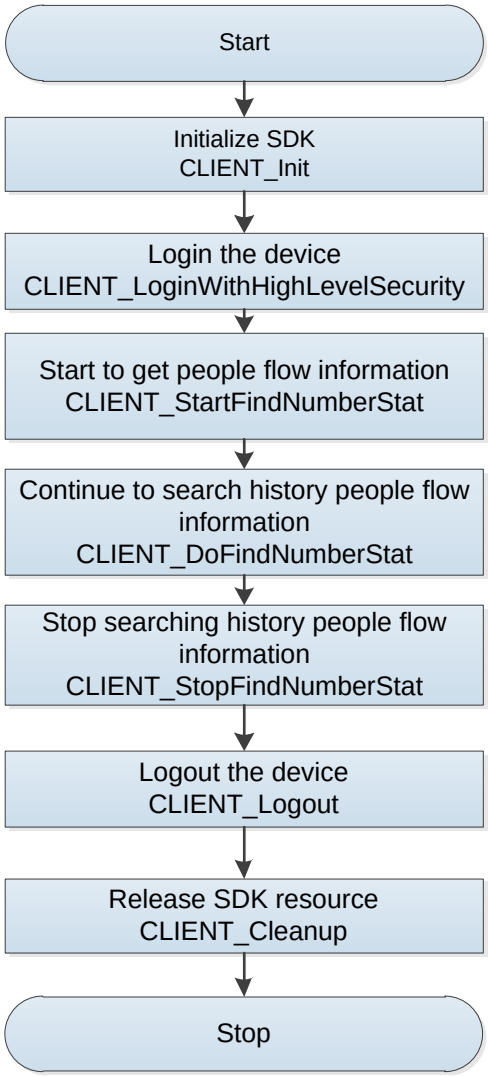
5.3.2 Interface Overview

Table 5-2 Interfaces of searching history data of people flow statistics

Interface	Implication
CLIENT_StartFindNumberStat	Start searching history people flow information.
CLIENT_DoFindNumberStat	Continue to search history people flow information.
CLIENT_StopFindNumberStat	Stop searching history people flow information.

5.3.3 Process

Figure 5-2 Searching process of people flow video and picture



Process Description

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

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

Step 3 Call **CLIENT_StartFindNumberStat** to get people flow statistics information.

Step 4 Call **CLIENT_DoFindNumberStat** to continue searching people flow statistics information at some period.

Step 5 Call **CLIENT_StopFindNumberStat** to stop searching records.

Step 6 After using the function module, call **CLIENT_Logout** to logout the device.

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

5.3.4 Example Code

```
// Set searching conditions
NET_IN_FINDNUMBERSTAT inParam = { sizeof(NET_IN_FINDNUMBERSTAT)};
inParam.nChannelID = nChannelID; // The channel number you want to search
```

```
inParam.nGranularityType = 1; // Searching unit, 0: minute, 1: hour, 2: day, 3: week, 4: month, 5: season, 6: year
inParam.nWaittime = 5000; // Time-out for waiting received data
NET_OUT_FINDNUMBERSTAT outParam { sizeof(NET_OUT_FINDNUMBERSTAT)};
LLONG findHnd = CLIENT_StartFindNumberStat(pLoginHandle, &inParam, &outParam);
if (findHand == 0)
{
    printf("CLIENT_StartFindNumberStat failed! Error code %x.\n", CLIENT_GetLastError());
    return ;
}

NET_IN_DOFINDNUMBERSTAT inDoFind = {sizeof(NET_IN_DOFINDNUMBERSTAT)};
NET_OUT_DOFINDNUMBERSTAT outDoFind = {sizeof(NET_OUT_DOFINDNUMBERSTAT)};
inDoFind.nBeginNumber = 0; // Search from 0
stuInDoFind.nCount = 10; // Search for 10 information one time.
inDoFind.nWaittime = 5000; // The time-out of interface is 5s.

outDoFind.pstuNumberStat = new DH_NUMBERSTAT[10];
outDoFind.nBufferLen = 10 * sizeof(DH_NUMBERSTAT);
for (int i = 0; i < 10 ; i++)
{
    outDoFind.pstuNumberStat[i].dwSize = sizeof(DH_NUMBERSTAT);
}

// Search
BOOL bRet = CLIENT_DoFindNumberStat(findHand, &inDoFind, &outDoFind)
if (FALSE == bRet)
{
    printf("CLIENT_DoFindNumberStat failed! Error code %x.\n", CLIENT_GetLastError());
    delete[] outDoFind.pstuNumberStat;
    return ;
}

// Stop searching people flow statistics
CLINET_StopFindNumberStat(findHand);
delete[] outDoFind.pstuNumberStat;
```

6 General Behavior Event

6.1 Subscribing General Behavior Event

For more details, see "2.4 Subscribing Intelligent Event". Call fAnalyzerDataCallBack to filter out general behavior events, which are EVENT_IVS_HUMANTRAIT for tripwire events and EVENT_IVS_CROSSREGIONDETECTION for intrusion events.

6.2 Video Searching and Downloading of General Behavior Event

The general behavior is one of the intelligent events. For more details, see "2.5 Searching/Playbacking/Downloading Video and Picture".

Call CLIENT_FindFileEx to set the searching conditions when searching. For the searching operation of general behavior video, the value of emType is as the follow:

DH_FILE_QUERY_FILE, searching video, and the interface parameter pQueryCondition is the structure pointer of type NET_IN_MEDIA_QUERY_FILE. The segments of nEventLists in the structure are as the follow:

- ◇ EVENT_IVS_CROSSLINEDETECTION, video searching of tripwire.
- ◇ EVENT_IVS_CROSSREGIONDETECTION, video searching of intrusion.

7 Intelligent Traffic

7.1 Subscribing Intelligent Traffic Event

About more details, see "2.4 Subscribing Intelligent Event". Call fAnalyzerDataCallBack to filter out the intelligent traffic event, which is as the following:

- EVENT_IVS_TRAFFICJUNCTION: Traffic junction event.
- EVENT_IVS_TRAFFICJAM: Traffic jam event.
- EVENT_IVS_TRAFFIC_OVERSPEED: Over speed event.
- EVENT_IVS_TRAFFIC_UNDERSPEED: Low speed event.
- EVENT_IVS_TRAFFIC_PEDESTRAIN: Passerby event.
- EVENT_IVS_TRAFFIC_FLOWSTATE: Vehicle flow event.

7.2 Searching History Data of Vehicle Flow Statistics

7.2.1 Introduction

Search the history data of vehicle flow statistics.

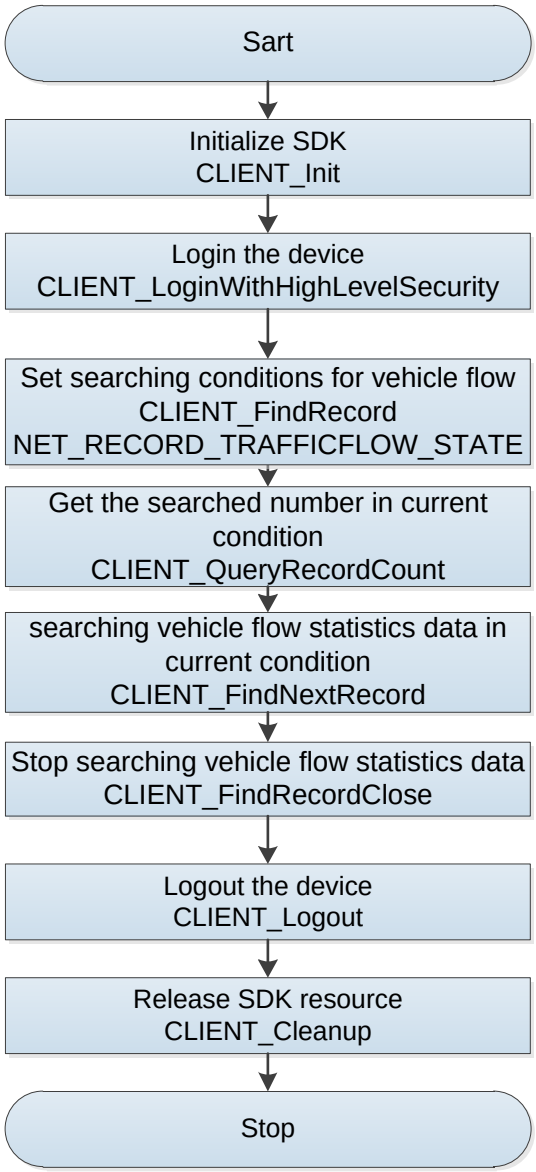
7.2.2 Interface Overview

Table 7-1 Interfaces of searching history data of vehicle flow statistics

Interface	Implication
CLIENT_FindRecord	Set searching conditions.
CLIENT_QueryRecordCount	Get searching number.
CLIENT_FindNextRecord	Search data in the current searching condition.
CLIENT_FindRecordClose	Stop searching.

7.2.3 Process

Figure 7-1 Searching history data of vehicle flow statistics



Process Description

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

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

Step 3 Call **CLIENT_FindRecord** to set searching conditions for vehicle floe statistics. The enumeration is NET_RECORD_TRAFFICFLOW_STATE.

Step 4 Call **CLIENT_QueryRecordCount** to get the total number in the current searching conditions.

Step 5 Call **CLIENT_FindNextRecord** to search the specifed number in the current searching conditions.

Step 6 After searching, call **CLIENT_FindRecordClose** to clean up the searching resource.

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

- During searching the vehicle flow, at first, the device must support this function and have the vehicle flow data during the searching time. In addition, the device should have an SD card to save the vehicle flow data.
- The segment of the average speed is -1, which means that no vehicle has passed this period. If it is more than 1, it means that the average speed of the vehicle. If it is equal to 0, it means that the average speed is 0.

7.2.4 Example Code

```
// Start searching and set searching conditions
FIND_RECORD_TRAFFICFLOW_CONDITION stTrafficFlow = {sizeof(FIND_RECORD_TRAFFICFLOW_CONDITION)};
stTrafficFlow.abChannelId =TRUE;
stTrafficFlow.nChannelId = 0;
stTrafficFlow.abLane = FALSE;
stTrafficFlow.bStartTime= TRUE;
stTrafficFlow.bEndTime= TRUE;
stTrafficFlow.stStartTime = startTime;
stTrafficFlow.stEndTime = endTime;
stTrafficFlow.bStatisticsTime = TRUE;

NET_IN_FIND_RECORD_PARAM stuFindInParam = {sizeof(NET_IN_FIND_RECORD_PARAM)};
stuFindInParam.emType = NET_RECORD_TRAFFICFLOW_STATE;
stuFindInParam.pQueryCondition = &stTrafficFlow;

NET_OUT_FIND_RECORD_PARAM stuFindOutParam = {sizeof(NET_OUT_FIND_RECORD_PARAM)};
bool bRet = CLIENT_FindRecord(m_lLoginHandle, &stuFindInParam, &stuFindOutParam, MAX_TIMEOUT);
if (!bRet)
{
    return;
}

// Search the total number
NET_IN_QUEYT_RECORD_COUNT_PARAM inQueryCountParam =
{ sizeof(NET_IN_QUEYT_RECORD_COUNT_PARAM)};
inQueryCountParam.lFindeHandle = stuFindOutParam.lFindeHandle;
NET_OUT_QUEYT_RECORD_COUNT_PARAM outQueryCountParam =
{ sizeof(NET_OUT_QUEYT_RECORD_COUNT_PARAM) };

bRet = CLIENT_QueryRecordCount(&inQueryCountParam, &outQueryCountParam , MAX_TIMEOUT);
```

```
if (!bRet)
{
    Printf("Query record count failed!\n");
    return;
}


// Search 100 information.
int nQueryCount = 100;
NET_RECORD_TRAFFIC_FLOW_STATE* pRecordList = new NET_RECORD_TRAFFIC_FLOW_STATE[nQueryCount];
memset(pRecordList, 0, sizeof(NET_RECORD_TRAFFIC_FLOW_STATE) * nQueryCount);
for (int unIndex = 0; unIndex < nQueryCount; ++unIndex)
{
    pRecordList[unIndex].dwSize = sizeof(NET_RECORD_TRAFFIC_FLOW_STATE);
}

NET_IN_FIND_NEXT_RECORD_PARAM stuFindNextInParam = {sizeof(NET_IN_FIND_NEXT_RECORD_PARAM)};
stuFindNextInParam.lFindeHandle = stuFindOutParam.lFindeHandle;
stuFindNextInParam.nFileCount = nQueryCount;

NET_OUT_FIND_NEXT_RECORD_PARAM stuFindNextOutParam = {sizeof(NET_OUT_FIND_NEXT_RECORD_PARAM)};
stuFindNextOutParam.pRecordList = pRecordList;
stuFindNextOutParam.nMaxRecordNum = nQueryCount;
bRet = CLIENT_FindNextRecord(&stuFindNextInParam, &stuFindNextOutParam, MAX_TIMEOUT);
if (!bRet)
{
    printf("Query record count failed!");
}

// Stop searching.
CLIENT_FindRecordClose(stuFindOutParam.lFindeHandle);
delete[] pRecordList;
```

7.3 Adding/deleting/modifying/searching Blocklist and Trusted List of Vehicle

7.3.1 Introduction

You can add, delete, modify and search blocklist and trusted list of vehicle.

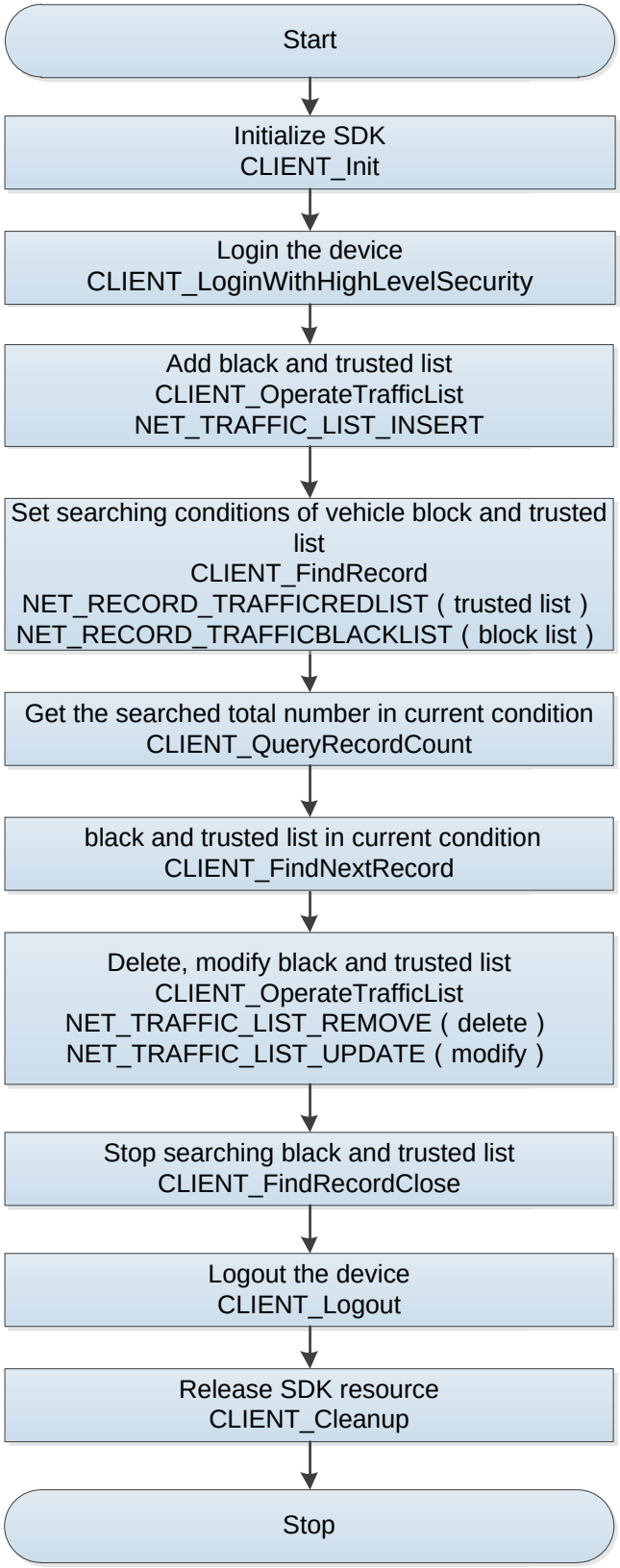
7.3.2 Interface Overview

Table 7-2 Interfaces of adding, deleting, modifying and searching blocklist and trusted list of vehicle,

Interface	Implication
CLIENT_OperateTrafficList	Add, delete, modify and search the blocklist and trusted list.
CLIENT_FindRecord	Set searching conditions.
CLIENT_QueryRecordCount	Get searching number.
CLIENT_FindNextRecord	Search data in the current searching condition.
CLIENT_FindRecordClose	Close searching.

7.3.3 Process

Figure 7-2 Adding/deleting/modifying/searching blocklist and trusted list of vehicle



Process Description

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

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

Step 3 Call **CLIENT_OperateTrafficList** to add the blocklist and trusted list. The enumeration type is NET_TRAFFIC_LIST_INSERT.

Step 4 Call **CLIENT_FindRecord** to set searching conditions for blocklist and trusted list. The enumeration is NET_RECORD_TRAFFICREDLIST (trusted list) and NET_RECORD_TRAFFICBLACKLIST (blocklist).

Step 5 Call **CLIENT_QueryRecordCount** to get the total number in the current searching conditions.

Step 6 Call **CLIENT_FindNextRecord** to search the specified number of the blocklist and trusted list in the current searching conditions.

Step 7 Using the blocklist and trusted list you have searched, call **CLIENT_OperateTrafficList** blocklist and trusted list. The enumeration is NET_TRAFFIC_LIST_REMOVE (delete) and NET_TRAFFIC_LIST_UPDATE (modify).

Step 8 After searching, call **CLIENT_FindRecordClose** to clean up the searching resource.

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

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

Notes for Process

Call **CLIENT_OperateTrafficList** to add blocklist and trusted list. When the interface returns -1, it means that do not generate a record list number but not means the interface failed. At first, the device saves the blocklist and trusted list to the cache, and then saves the data in the cache to the database. The device returns -1 because at this time the lists have not been added to the database.

7.3.4 Example Code

To check the code of blocklist and trusted list, see "7.2.4 Example Code". The following shows the example code of adding/deleting/modifying/searching blocklist and trusted list.

```
// Add block list and trusted list.
NET_IN_OPERATE_TRAFFIC_LIST_RECORD stInParam = { sizeof(NET_IN_OPERATE_TRAFFIC_LIST_RECORD) };
NET_OUT_OPERATE_TRAFFIC_LIST_RECORD stOutParam = { sizeof(NET_OUT_OPERATE_TRAFFIC_LIST_RECORD) };
stInParam.emOperateType = NET_TRAFFIC_LIST_INSERT;
stInParam.emRecordType = NET_RECORD_TRAFFICBLACKLIST; // Blocklist
//stInParam.emRecordType = NET_RECORD_TRAFFICREDLIST; // Trusted list

NET_TRAFFIC_LIST_RECORD stTrafficListRecord = { sizeof(NET_TRAFFIC_LIST_RECORD) };
stTrafficListRecord.stBeginTime = startTime;
stTrafficListRecord.stCancelTime = endTime;
strncpy(stTrafficListRecord.szPlateNumber, strPlateNumber.GetBuffer(), DH_MAX_PLATE_NUMBER_LEN-1);
strncpy(stTrafficListRecord.szMasterOfCar, strOwner.GetBuffer(), DH_MAX_NAME_LEN-1);

NET_INSERT_RECORD_INFO stInsertInfo = { sizeof( NET_INSERT_RECORD_INFO ) };
stInsertInfo.pRecordInfo = &stTrafficListRecord;
```

```
stInParam.pstOpreateInfo = &stInsertInfo;



bool bRet = CLIENT_OperateTrafficList(m_lLoginHandle, &stInParam, &stOutParam, MAX_TIMEOUT);
if (!bRet)
{
    return;
}

// Modify blocklist.
NET_IN_OPERATE_TRAFFIC_LIST_RECORD stInParam = { sizeof(NET_IN_OPERATE_TRAFFIC_LIST_RECORD) };
NET_OUT_OPERATE_TRAFFIC_LIST_RECORD stOutParam =
{ sizeof(NET_OUT_OPERATE_TRAFFIC_LIST_RECORD) };

stInParam.emOperateType = NET_TRAFFIC_LIST_UPDATE;
stInParam.emRecordType = NET_RECORD_TRAFFICBLACKLIST; // Blocklist
//stInParam.emRecordType = NET_RECORD_TRAFFICREDLIST; // Trusted list

NET_TRAFFIC_LIST_RECORD stTrafficListRecord = { sizeof(NET_TRAFFIC_LIST_RECORD) };
stTrafficListRecord.stBeginTime = startTime;
stTrafficListRecord.stCancelTime = endTime;
strncpy(stTrafficListRecord.szPlateNumber, strPlateNumber.GetBuffer(), DH_MAX_PLATE_NUMBER_LEN-1);
strncpy(stTrafficListRecord.szMasterOfCar, strOwner.GetBuffer(), DH_MAX_NAME_LEN-1);
stTrafficListRecord.nRecordNo = m_stTrafficListInfo.nRecordNo;    // Recording list number

NET_UPDATE_RECORD_INFO stModifyRecord = {sizeof(NET_UPDATE_RECORD_INFO)};
stModifyRecord.pRecordInfo = &stTrafficListRecord;
stInParam.pstOpreateInfo = &stModifyRecord;

bool bRet = CLIENT_OperateTrafficList(m_lLoginHandle, &stInParam, &stOutParam, MAX_TIMEOUT);
if (!bRet)
{
    return;
}

// Modify blocklist and trusted list.
NET_REMOVE_RECORD_INFO stRemoveRecord = { sizeof(NET_REMOVE_RECORD_INFO) };
stRemoveRecord.nRecordNo = m_vecTrafficListInfo[nSelect]->nRecordNo; // Recording list number

NET_IN_OPERATE_TRAFFIC_LIST_RECORD stInParam = { sizeof(NET_IN_OPERATE_TRAFFIC_LIST_RECORD) };
```

```
stInParam.emOperateType = NET_TRAFFIC_LIST_REMOVE;
stInParam.emRecordType = NET_RECORD_TRAFFICBLACKLIST; // Blocklist
//stInParam.emRecordType = NET_RECORD_TRAFFICREDLIST; // Trusted list

stInParam.pstOpreateInfo = &stRemoveRecord;
NET_OUT_OPERATE_TRAFFIC_LIST_RECORD stOutParam =
{ sizeof(NET_OUT_OPERATE_TRAFFIC_LIST_RECORD) };

bool bRet = CLIENT_OperateTrafficList(m_lLoginHandle, &stInParam, &stOutParam, MAX_TIMEOUT);
if (!bRet)
{
    return;
}
```

7.4 Searching and Downloading Vehicle Picture

7.4.1 Introduction

This function will save the capture picture for intelligent event to the storage in the device. You can search pictures through the plate number and corresponding intelligent events. This function also supports picture downloading.

For the vehicle picture searching, see "2.5 Searching/Playbacking/Downloading Video and Picture". The interface **CLIENT_FindFileEx** searching type is DH_FILE_QUERY_TRAFFICCAR_EX.

The following shows the interface and example code.

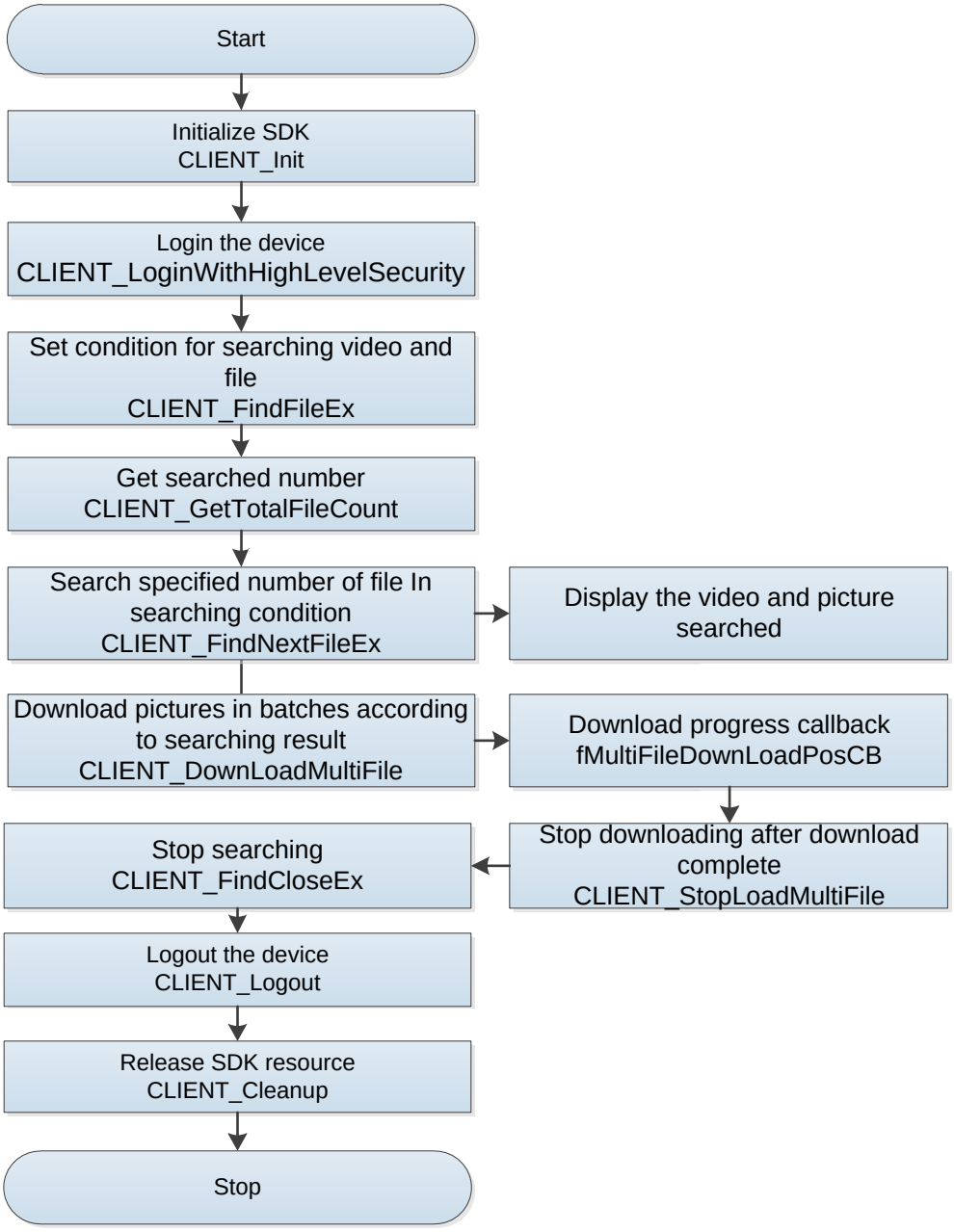
7.4.2 Interface Overview

Table 7-3 Interfaces of searching and downlosding picture

Interface	Implication
CLIENT_DownLoadMultiFile	Download files in batches.
CLIENT_StopLoadMultiFile	Stop downloading files in batches.

7.4.3 Process

Figure 7-3 Process of searching and downloading picture



Process Description

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

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

Step 3 Call **CLIENT_FindFileEx** to set the searching conditions. After successfully setting, return the searching handle. To judge the right searching type according to the different values of emType.

Step 4 Call **CLIENT_GetTotalFileCount** to get the total number of video and picture searched.

Step 5 Call **CLIENT_FindNextFileEx** to search the specified number of video and picture. Save the video and picture and do the playing back and downloading operation to the video and picture.

Step 6 Using the searched file information, call **CLIENT_DownLoadMultiFile** to download files in batches.

Step 7 Call **CLIENT_StopLoadMultiFile** to stop downloading the picture when the value of dwDownLoadSize in fMultiFileDownLoadPosCB is the maximum value.

Step 8 Call **CLIENT_FindCloseEx** to stop the searching.

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

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

Notes for Process

- The parameter pQueryCondition in **CLIENT_FindFileEx** is requested and released by the user. The spacific type is defined by the enumeration type of emType.
- If **CLIENT_FindFileEx** sucessfully search, the searching handle will be returned. **CLIENT_FindNextFileEx** will take the searching handle as a parameter to search specific video and picture. You should call **CLIENT_FindCloseEx** to close the searching handle.
- **Call CLIENT_FindNextFileEx** to set the searching number. If the number is more than 1, then the parameter pMediaFileInfo should be taken as a data pointer.

7.4.4 Example Code

```
// Download progress callback function in batches
void CALLBACK DownloadTrafficPicture(LLONG lDownLoadHandle, DWORD dwID, DWORD dwFileTotalSize, DWORD dwDownLoadSize, int nError, LDWORD dwUser, void* pReserved)
{
    if (nError != 0 || dwDownLoadSize == UINT_MAX)
    {
        // Download error or download completed
        CLIENT_StopLoadMultiFile(m_lDownloadHandle);
        delete[] pDownLoadInfo;
        }
}

NET_DOWNLOADFILE_INFO* pDownLoadInfo = new NET_DOWNLOADFILE_INFO[10]; // Download 10 information
memset(pDownLoadInfo, 0, 10*sizeof(NET_DOWNLOADFILE_INFO));
for(int i =0; i++; i<10)
{
    pDownLoadInfo[i].dwFileID = 1;

    // The data of stTrafficPicture is the file searched from CLIENT_FindFileEx. The type is MEDIAFILE_TRAFFICCAR_INFO_EX.
    pDownLoadInfo[i].nFileSize = stTrafficPicture.stuInfo.sizeEx / 1024;
    strncpy(pDownLoadInfo[i].szSourceFilePath, stTrafficPicture.stuInfo.szFilePath, MAX_PATH-1);
```

```
        // The save path after downloading
        strncpy(pDownLoadInfo[i].szSavedFileName, szFilePathName[i], MAX_PATH-1);
}

NET_IN_DOWNLOAD_MULTI_FILE stInDownloadFile = {sizeof(NET_IN_DOWNLOAD_MULTI_FILE )};
NET_OUT_DOWNLOAD_MULTI_FILE stOutDownloadFile= {sizeof(NET_OUT_DOWNLOAD_MULTI_FILE )};
stInDownloadFile.emDownloadType = EM_DOWNLOAD_BY_FILENAME;
stInDownloadFile.cbPosCallBack = DownloadTrafficPicture;  // Download in batches progress callback function
stInDownloadFile.dwUserData =  (LDWORD)this;
stInDownloadFile.nFileCount = 10; // The number of downloading file
stInDownloadFile.pFileInfos = pDownLoadInfo; // It means the file information pointer that you need to download. If you need to download multiple file information pointers, it means the array first address.

// Download picture files in batches
BOOL nRet = CLIENT_DownLoadMultiFile(m_lLoginHandle, &stInDownloadFile, &stOutDownloadFile, MAX_TIMEOUT);
if (!nRet)
{
        printf("CLIENT_DownLoadMultiFile failed! Error code %x.\n", CLIENT_GetLastError());
        delete[] pDownLoadInfo;
        return;
}
m_lDownloadHandle = stOutDownloadFile.lDownLoadHandle;
```

8 Barrier

8.1 Subscribing Access Control Event

About more details, see "2.4 Subscribing Intelligent Event". Call fAnalyzerDataCallBack to filter out access control event, which is as the following:
EVENT_IVS_ACCESS_CTL: Access control event

8.2 Manager Information of Access Control Card

8.2.1 Introduction

You can add, delete, modify and search the information of access control card such as the card number, user ID and card name.

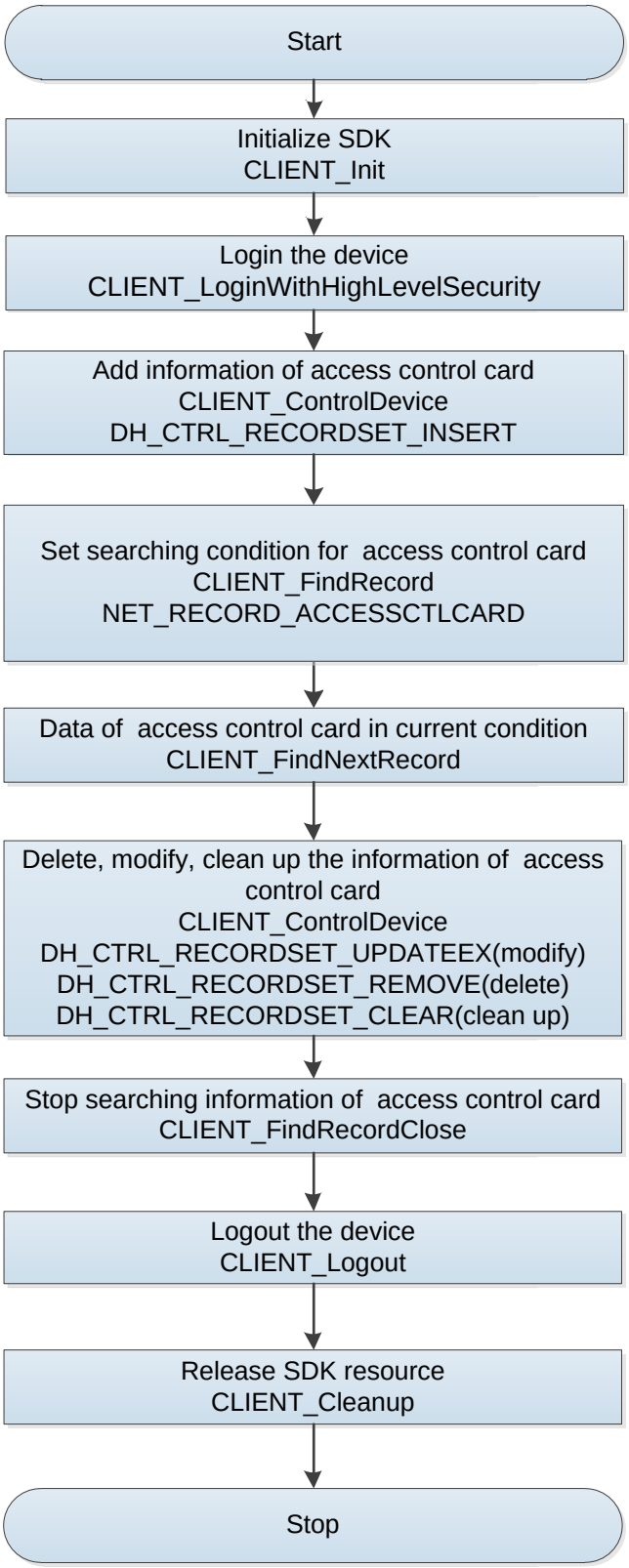
8.2.2 Interface Overview

Table 8-1 Interfaces of manager information of access control card

Interface	Implication
CLIENT_FindRecord	Set searching conditions.
CLIENT_FindNextRecord	Search data in the current searching condition.
CLIENT_FindRecordClose	Close searching.
CLIENT_ControlDevice	Add, delete, modify and search the information of access control card.

8.2.3 Process

Figure 8-1 Adding/deleting/modifying/searching the information of access control card



Process Description

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

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

Step 3 Call **CLIENT_ControlDevice** to add the information of access control card. The enumeration is DH_CTRL_RECORDSET_INSERT.

Step 4 Call **CLIENT_FindRecord** to set searching conditions for the information of access control card. The enumeration is NET_RECORD_ACCESSCTLCARD.

Step 5 Call **CLIENT_FindNextRecord** to search the specified number of the information data of access control card in the current searching conditions.

Step 6 After searching, call **CLIENT_FindRecordClose** to clean up the searching resource.

Step 7 Call **CLIENT_ControlDevice** to modify the information of access control card. The enumeration is NET_RECORD_ACCESSCTLCARD.

Step 8 Call **CLIENT_ControlDevice** to delete the information of access control card. The enumeration is DH_CTRL_RECORDSET_REMOVE.

Step 9 Call **CLIENT_ControlDevice** to clean up the information of access control card. The enumeration is DH_CTRL_RECORDSET_CLEAR.

Step 10 After using the function module, call **CLIENT_Logout** to logout the device.

Step 11 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- When adding the access control card, make sure the CardNo and UserID are different with the information of access control card which saved in the device before.
- When deleting the access control card, and if the access control card binding with people face picture information, you should delete the information of access control card before deleting people face picture.
- During searching the information of access control card, firstly the device should support this function, and the device itself should have the information data of access control card.

8.2.4 Example Code

8.2.4.1 Searching the Information of Access Control Card

```
NET_OUT_FIND_RECORD_PARAM stuOutParam= sizeof(NET_OUT_FIND_RECORD_PARAM);
NET_IN_FIND_RECORD_PARAM stuInParam= sizeof(stuInParam);

stuInParam.emType = NET_RECORD_ACCESSCTLCARD;
FIND_RECORD_ACCESSCTLCARD_CONDITION *pStuCardInfo = NEW FIND_RECORD_ACCESSCTLCARD_CONDITION;
pStuCardInfo->dwSize = sizeof(FIND_RECORD_ACCESSCTLCARD_CONDITION);
stuInParam.pQueryCondition = (void*)pStuCardInfo;

// Start searching and set searching conditions
bRet = CLIENT_FindRecord(m_lLoginID, &stuInParam, &stuOutParam, DEFAULT_WAIT_TIME);
if (bRet == FALSE)
{
    printf("CLIENT_FindRecord fail \n");
```

```
}

if (pStuCardInfo)
{
	delete pStuCardInfo;
	pStuCardInfo = NULL;
}

// Search 100 information.
NET_IN_FIND_NEXT_RECORD_PARAM stuInParam= sizeof(stuInParam);
NET_OUT_FIND_NEXT_RECORD_PARAM	stuOutParam = sizeof(stuOutParam);
stuInParam.lFindeHandle = m_lFindHandle;
stuInParam.nFileCount = 100;
memset(pAccessCardInfo, 0, stuInParam.nFileCount * sizeof(NET_RECORDSET_ACCESS_CTL_CARD));

for (int i = 0; i < stuInParam.nFileCount; i++)
{
	pAccessCardInfo[i].dwSize = sizeof(NET_RECORDSET_ACCESS_CTL_CARD);
}
stuOutParam.pRecordList = (void*)pAccessCardInfo;
stuOutParam.nMaxRecordNum = stuInParam.nFileCount;

int nRet = CLIENT_FindNextRecord(&stuInParam, &stuOutParam, DEFAULT_WAIT_TIME);

if (!nRet)
{
	printf("CLIENT_FindNextRecord failed \n");
}
// Stop searching.
CLIENT_FindRecordClose(m_lFindHandle);
```

8.2.4.2 Adding/Deleting/Modifying/Searching the Information of Access Control Card

```
/ / Add the information of access control card
NET_CTRL_RECORDSET_INSERT_PARAM stuInParam = sizeof(stuInParam);
stuInParam.stuCtrlRecordSetInfo.dwSize = sizeof(NET_CTRL_RECORDSET_INSERT_IN);
stuInParam.stuCtrlRecordSetResult.dwSize = sizeof(NET_CTRL_RECORDSET_INSERT_OUT);
stuInParam.stuCtrlRecordSetInfo.emType = NET_RECORD_ACCESSCTLCARD;
```

```
NET_RECORDSET_ACCESS_CTL_CARD *pStrCardInfo = NEW NET_RECORDSET_ACCESS_CTL_CARD;

pStrCardInfo->dwSize = sizeof(NET_RECORDSET_ACCESS_CTL_CARD);
strncpy(pStrCardInfo->szCardNo, m_StuAddCardInfo.szCardNo, DH_MAX_CARDNO_LEN - 1);
strncpy(pStrCardInfo->szCardName, m_StuAddCardInfo.szCardName, DH_MAX_CARDNAME_LEN - 1);
strncpy(pStrCardInfo->szUserID, m_StuAddCardInfo.szUserID, DH_MAX_USERID_LEN - 1);
strncpy(pStrCardInfo->szPsw, m_StuAddCardInfo.szPsw, DH_MAX_CARDPWD_LEN - 1);
pStrCardInfo->emStatus = NET_ACCESSCTLCARD_STATE_NORMAL;
pStrCardInfo->emType = NET_ACCESSCTLCARD_TYPE_GENERAL;
pStrCardInfo->nUserTime = m_StuAddCardInfo.nUserTime;
pStrCardInfo->bFirstEnter = TRUE;
pStrCardInfo->bIsValid = TRUE
pStrCardInfo->stuValidStartTime = m_StuAddCardInfo.stuValidStartTime;
pStrCardInfo->stuValidEndTime = m_StuAddCardInfo.stuValidEndTime;

// DoorNum is 2, it indicates the two doors of the gate.
pStrCardInfo->nDoorNum = 2;
pStrCardInfo->sznDoors[0] = 0;
pStrCardInfo->sznDoors[1] = 1;
// Control the valid time of the opening door, 255 indicates all day.
pStrCardInfo->nTimeSectionNum = 2;
pStrCardInfo->sznTimeSectionNo[0] = 255;
pStrCardInfo->sznTimeSectionNo[1] = 255;

stuInParam.stuCtrlRecordSetInfo.nBufLen = sizeof(NET_RECORDSET_ACCESS_CTL_CARD);
stuInParam.stuCtrlRecordSetInfo.pBuf = (void*)pStrCardInfo;

BOOL bRet = CLIENT_ControlDevice(m_lLoginID, DH_CTRL_RECORDSET_INSERT, &stuInParam,
DEFAULT_WAIT_TIME);
if (bRet == FALSE)
{
    Printf("CLIENT_ControlDevice insert card fail \n");
    return;
}
/ / Modify the information of access control card
NET_CTRL_RECORDSET_PARAM stuInParam = sizeof(stuInParam);
stuInParam.emType = NET_RECORD_ACCESSCTLCARD;

NET_RECORDSET_ACCESS_CTL_CARD *pStrCardInfo = NEW NET_RECORDSET_ACCESS_CTL_CARD;
```

```
memset(pStrCardInfo, 0, sizeof(NET_RECORDSET_ACCESS_CTL_CARD));

pStrCardInfo->dwSize = sizeof(NET_RECORDSET_ACCESS_CTL_CARD);
strncpy(pStrCardInfo->szCardNo, m_CardInfo.szCardNo, DH_MAX_CARDNO_LEN - 1);
strncpy(pStrCardInfo->szCardName, m_CardInfo.szCardName, DH_MAX_CARDNAME_LEN - 1);
strncpy(pStrCardInfo->szUserID, m_CardInfo.szUserID, DH_MAX_USERID_LEN - 1);
strncpy(pStrCardInfo->szPsw, m_CardInfo.szPsw, DH_MAX_CARDPWD_LEN - 1);
pStrCardInfo->emStatus = NET_ACCESSCTLCARD_STATE_NORMAL;
pStrCardInfo->emType = NET_ACCESSCTLCARD_TYPE_GENERAL;
pStrCardInfo->nUserTime = m_CardInfo.nUserTime;
pStrCardInfo->bFirstEnter = TRUE;
pStrCardInfo->bIsValid =TRUE;
pStrCardInfo->stuValidStartTime = m_CardInfo.stuValidStartTime;
pStrCardInfo->stuValidEndTime = m_CardInfo.stuValidEndTime;
pStrCardInfo->nRecNo = m_CardInfo.nRecNo;

// DoorNum is 2, it indicates the two doors of the gate.
pStrCardInfo->nDoorNum = 2;
pStrCardInfo->sznDoors[0] = 0;
pStrCardInfo->sznDoors[1] = 1;
// Control the valid time of the opening door, 255 indicates all day.
pStrCardInfo->nTimeSectionNum = 2;
pStrCardInfo->sznTimeSectionNo[0] = 255;
pStrCardInfo->sznTimeSectionNo[1] = 255;

stuInParam.nBufLen = sizeof(NET_RECORDSET_ACCESS_CTL_CARD);
stuInParam.pBuf = (void*)pStrCardInfo;

BOOL bRet = CLIENT_ControlDevice(m_lLoginID, DH_CTRL_RECORDSET_UPDATEEX, &stuInParam,
DEFAULT_WAIT_TIME);
if (FALSE == bRet)
{
    printf ("CLIENT_ControlDevice    modify cardinfo fail");
    return;
}

/ / Delete the information of access control card
NET_CTRL_RECORDSET_PARAM stuInParam = sizeof(stuInParam);
stuInParam.emType = NET_RECORD_ACCESSCTLCARD;
stuInParam.pBuf = &pCardInfo->nRecNo;
```

```
stuInParam.nBufLen = sizeof(int);
BOOL bRet = CLIENT_ControlDevice(m_lLoginID, DH_CTRL_RECORDSET_REMOVE, &stuInParam, DEFAULT_WAIT_TIME);
if (FALSE == bRet)
{
    printf(("CLIENT_ControlDevice    delete cardinfo fail\n");
    return;
}

/ / Clean up the information of access control card (Delete all information of access control card)
NET_CTRL_RECORDSET_PARAM stuInParam = sizeof(stuInParam);
stuInParam.emType = NET_RECORD_ACCESSCTLCARD;

BOOL bRet = CLIENT_ControlDevice(m_lLoginID, DH_CTRL_RECORDSET_CLEAR, &stuInParam, DEFAULT_WAIT_TIME);
if (FALSE == bRet)
{
    printf ("CLIENT_ControlDevice    clear cardinfo fail\n");
    return;
}
```

8.3 Face Management

8.3.1 Introduction

Add, modify, delete and clean up the face picture.

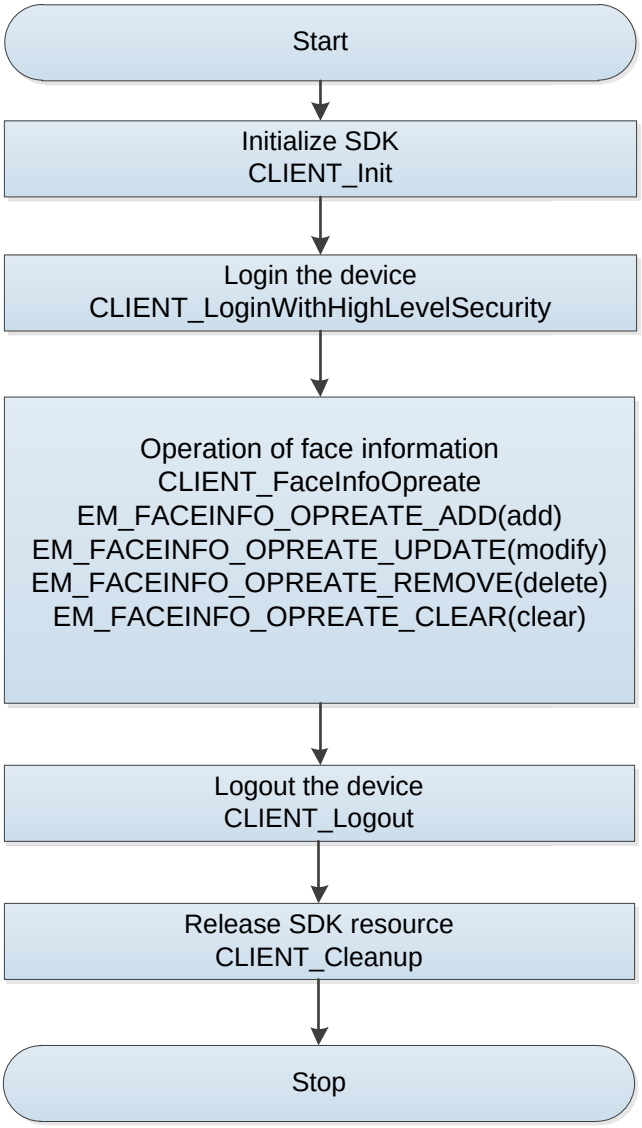
8.3.2 Interface Overview

Table 8-2 Interfaces of face management

Interface	Implication
CLIENT_FaceInfoOpreate	Add, delete, modify and clean up the information of face picture.
CLIENT_FindRecord	Set searching conditions.
CLIENT_FindNextRecord	Search data in the current searching condition.
CLIENT_FindRecordClose	Close searching.
CLIENT_ControlDevice	Add, delete, modify and clean up the people information.

8.3.3 Process

Figure 8-2 Adding/deleting/modifying/cleaning up the information of face picture



Process Description

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

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

Step 3 Call **CLIENT_FaceInfoOpreate** to add, modify and delete the face according to enumeration type.

Step 4 After using the function module, call **CLIENT_Logout** to logout the device.

Step 5 After using all SDK functions, call **CLIENT_Cleanup** to release SDK resource.

Notes for Process

- Face information adding: Use EM_FACEINFO_OPREATE_ADD as the enumeration.
- Face information modifying: Use EM_FACEINFO_OPREATE_UPDATE as the enumeration.
- Face information deleting: Use EM_FACEINFO_OPREATE_REMOVE as the enumeration.

- Face information cleaning up: Use EM_FACEINFO_OPREATE_CLEAR as the enumeration.
- The face information modifying and deleting are according to the UserID of the access control card.

8.3.4 Example Code

```
/ / Add the information of face picture
NET_IN_ADD_FACE_INFO stuInParam = sizeof(NET_IN_ADD_FACE_INFO);
strncpy(stuInParam.szUserID, m_StuAddCardInfo.szUserID, DH_MAX_USERID_LEN - 1);
stuInParam.stuFaceInfo.nFacePhoto = 1;

FILE *fPic = fopen(m_szFilePath, "rb");
fseek(fPic, 0, SEEK_END);
int nLength = ftell(fPic);
rewind(fPic);

stuInParam.stuFaceInfo.nFacePhotoLen[0] = nLength;
stuInParam.stuFaceInfo.pszFacePhoto[0] = new char[nLength];

memset(stuInParam.stuFaceInfo.pszFacePhoto[0], 0, nLength);
int nReadLen = fread(stuInParam.stuFaceInfo.pszFacePhoto[0], 1, nLength, fPic);
fclose(fPic);
fPic = NULL;

NET_OUT_ADD_FACE_INFO stuOutParam = sizeof(stuOutParam);

BOOL bRet = CLIENT_FaceInfoOpreate(m_lLoginID, EM_FACEINFO_OPREATE_ADD, (void*)&stuInParam, (void*)&stuOutParam, DEFAULT_WAIT_TIME);
if (bRet == FALSE)
{
    printf("CLIENT_FaceInfoOpreate add face fail");
}

if (fPic)
{
    fclose(fPic);
    fPic = NULL;
}
if (stuInParam.stuFaceInfo.pszFacePhoto[0])
{
    delete[] stuInParam.stuFaceInfo.pszFacePhoto[0];
```

```
    stuInParam.stuFaceInfo.pszFacePhoto[0] = NULL;
}
/ / Modify the information of face picture
NET_IN_UPDATE_FACE_INFO stuInParam = sizeof(stuInParam);
strncpy(stuInParam.szUserID, m_CardInfo.szUserID, sizeof(m_CardInfo.szUserID) - 1);
stuInParam.stuFaceInfo.nFacePhoto = 1;

FILE *fPic = fopen(m_szFilePath, "rb");

fseek(fPic, 0, SEEK_END);
int nLength = ftell(fPic);
rewind(fPic);

stuInParam.stuFaceInfo.nFacePhotoLen[0] = nLength;
stuInParam.stuFaceInfo.pszFacePhoto[0] = new char[nLength];

memset(stuInParam.stuFaceInfo.pszFacePhoto[0], 0, nLength);
int nReadLen = fread(stuInParam.stuFaceInfo.pszFacePhoto[0], 1, nLength, fPic);
fclose(fPic);
fPic = NULL;

NET_OUT_UPDATE_FACE_INFO stuOutParam;
memset(&stuOutParam, 0, sizeof(stuOutParam));
stuOutParam.dwSize = sizeof(stuOutParam);

BOOL bRet = CLIENT_FaceInfoOpreate(m_lLoginID, EM_FACEINFO_OPREATE_UPDATE, (void*)&stuInParam,
(void*)&stuOutParam, DEFAULT_WAIT_TIME);
if (bRet == FALSE)
{
    printf ("CLIENT_FaceInfoOpreate modify face fail");
}

if (fPic)
{
    fclose(fPic);
    fPic = NULL;
}
if (stuInParam.stuFaceInfo.pszFacePhoto[0])
{
    delete[] stuInParam.stuFaceInfo.pszFacePhoto[0];
```

```
    stuInParam.stuFaceInfo.pszFacePhoto[0] = NULL;
}

/ / Delete the face picture
NET_IN_REMOVE_FACE_INFO stuInParam = sizeof(stuInParam);
// Delete the face picture according to the UserID of the information of access control card.
strncpy(stuInParam.szUserID, cardInfo.szUserID, DH_MAX_USERID_LEN - 1);

NET_OUT_REMOVE_FACE_INFO stuOutParam;
memset(&stuOutParam, 0, sizeof(stuOutParam));
stuOutParam.dwSize = sizeof(stuOutParam);

bRet = CLIENT_FaceInfoOpreate(m_lLoginID, EM_FACEINFO_OPREATE_REMOVE, (void*)&stuInParam, (void*)&stuOutParam, DEFAULT_WAIT_TIME);
if (bRet == FALSE)
{
    printf ("CLIENT_FaceInfoOpreate delete face fail");
}

/ / Clean up the information of face picture(Delete all information of face picture)
NET_IN_CLEAR_FACE_INFO stuInParam = sizeof(stuInParam);
NET_OUT_CLEAR_FACE_INFO stuOutParam;
memset(&stuOutParam, 0, sizeof(stuOutParam));
stuOutParam.dwSize = sizeof(stuOutParam);

BOOL bRet = CLIENT_FaceInfoOpreate(m_lLoginID, EM_FACEINFO_OPREATE_CLEAR, (void*)&stuInParam, (void*)&stuOutParam, DEFAULT_WAIT_TIME);
if (bRet == FALSE)
{
    printf ("CLIENT_FaceInfoOpreate clear face fail");
}
```

8.4 Searching the Record of In-Out the Door

8.4.1 Introduction

Search the record list of access control. The searching information includes card No., user serial number, the status of opening the door, card type, the mode of opening the door and time.

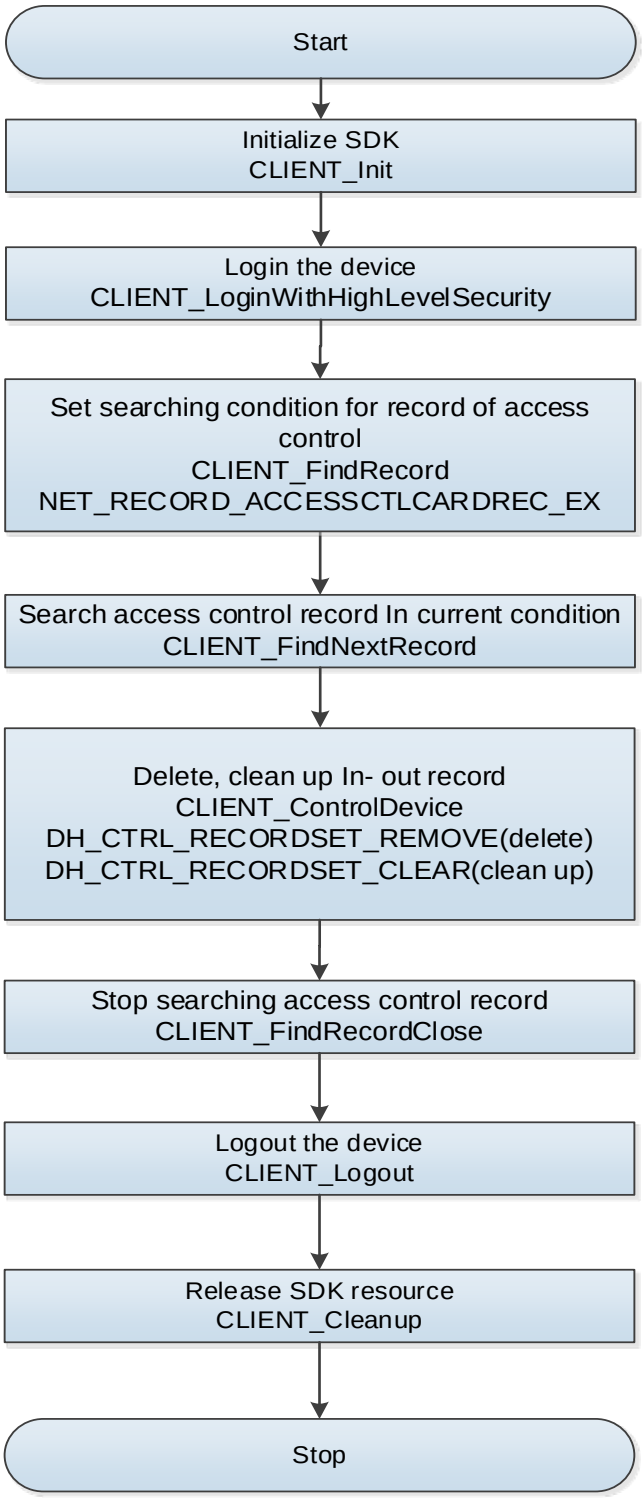
8.4.2 Interface Overview

Table 8-3 Interfaces of searching the record of in-out the door

Interface	Implication
CLIENT_FindRecord	Set searching conditions.
CLIENT_FindNextRecord	Search data in the current searching condition.
CLIENT_FindRecordClose	Close searching.
CLIENT_ControlDevice	Delete and clean up the record of access control.

8.4.3 Process

Figure 8-3 Searching/deleting/cleaning up the record of access control



Process Description

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

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

Step 3 Call **CLIENT_FindRecord** to set searching conditions for the record of access control.

Step 4 Call **CLIENT_FindNextRecord** to search the specified number of the record of access control in the current searching conditions.

Step 5 Call **CLIENT_ControlDevice** to delete and clean up the record of access control.

Step 6 After searching, call **CLIENT_FindRecordClose** to clean up the searching resource Information.

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

- Access control record deleting: Use DH_CTRL_RECORDSET_REMOVE as the enumeration.
- Access control record cleaning up: Use DH_CTRL_RECORDSET_CLEAR as the enumeration.

8.4.4 Example Code

About the codes, see "8.2.4.1 Searching the Information of Access Control Card".

The example code for deleting and cleaning up the access control records, see "8.2.4.2 Adding/Deleting/Modifying/Searching the Information of Access Control Card".

9 Interface Definition

9.1 SDK Initialization

9.1.1 SDK CLIENT_Init

Table 9-1 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.	

9.1.2 CLIENT_Cleanup

Table 9-2 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.

9.1.3 CLIENT_SetAutoReconnect

Table 9-3 Set reconnection callback

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

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

9.1.4 CLIENT_SetNetworkParam

Table 9-4 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.	

9.2 Device Login

9.2.1 CLIENT_LoginWithHighLevelSecurity

Table 9-5 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 9-6 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 blocklisted.
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	Lack avnetsdk or the dependent libraries of avnetsdk.
12	USB flash disk is not inserted into device, or the USB flash disk information error.
13	The IP at client is not authorized for login.

9.2.2 CLIENT_Logout

Table 9-7 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.	

9.3 Real-time Monitoring

9.3.1 CLIENT_RealPlayEx

Table 9-8 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.

Item	Description	
	[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 9-9 Live view type and meaning

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

9.3.2 CLIENT_StopRealPlayEx

Table 9-10 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.	

9.3.3 CLIENT_SaveRealData

Table 9-11 Save the real-time monitoring data as file

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

Item	Description	
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.	

9.3.4 CLIENT_StopSaveRealData

Table 9-12 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.	

9.3.5 CLIENT_SetRealDataCallBackEx

Table 9-13 Set the callback of real-time monitoring data

Item	Description	
Name	Set the callback of real-time monitoring data.	
Function	BOOL CLIENT_SetRealDataCallBackEx(LLONG lRealHandle, fRealDataCallBackEx 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 9-14 dwFlag type and parameter

dwFlag	Meaning
0x00000001	The original data of device.
0x00000004	YUV data.

9.4 Subscribing Intelligent Event

9.4.1 CLIENT_RealLoadPictureEx

Table 9-15 Subscribe intelligent event interface

Item	Description	
Name	Subscribe intelligent event interface.	
Function	LLONG CLIENT_RealLoadPictureEx(LLONG lLoginID, int nChannelID, DWORD dwAlarmType, BOOL bNeedPicFile, fAnalyzerDataCallBack cbAnalyzerData, LDWORD dwUser, void* Reserved);	
Parameter	[in] lLoginID	Login handle.
	[in] nChannelID	Channel number.
	[in] dwAlarmType	Alarm type.
	[in] bNeedPicFile	Whether to subscribe picture file, 1-yes, return intelligent picture information in the callback function; 0-no, do not return intelligent picture information(in this case, it can reduce the network flow).
	[in] cbAnalyzerData	Intelligent data analysis callback function.
	[in] dwUser	The user parameters.
	[in] Reserved	Reserve parameter.
Return value	Success: the subscribe handle of LLONG type. Failure: 0.	
Note	If the interface return failed, call CLIENT_GetLastError to get error code.	

Table 9-16 Preview type and meaning

Preview type.	Meaning
EVENT_IVS_FACEDETECT	Face Detection.
EVENT_IVS_FACERECOGNITION	Target Recognition.
EVENT_IVS_TRAFFICJUNCTION	Traffic junction event.
EVENT_IVS_TRAFFICJAM	Traffic jam event.
EVENT_IVS_TRAFFIC_OVERSPEED	Over speed event.
EVENT_IVS_TRAFFIC_UNDERSPEED	Low speed.
EVENT_IVS_TRAFFIC_FLOWSTATE	Vehicle flow event.
EVENT_IVS_HUMANTRAIT	Body detection event.
EVENT_IVS_CROSSLINEDETECTION	Tripwire event.
EVENT_IVS_CROSSREGIONDETECTION	Intrusuion event.
EVENT_IVS_NUMBERSTAT	People counting event.
EVENT_IVS_MAN_NUM_DETECTION	People countingin event in the eara.
EVENT_IVS_ACCESS_CTL	Access control event.

9.4.2 CLIENT_StopLoadPic

Table 9-17 Stop subscribing intelligent event.

Item	Description	
Name	Stop subscribing intelligent event.	
Function	BOOL CLIENT_StopLoadPic(LLONG lAnalyzerHandle);	
Parameter	[in] lAnalyzerHandle	Event subscribing handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.5 Searching and Downloading Intelligent Video and Picture

9.5.1 CLIENT_FindFileEx

Table 9-18 Search files by conditions

Item	Description	
Name	Search files by conditions.	
Function	LLONG CLIENT_FindFileEx(LLONG lLoginID, EM_FILE_QUERY_TYPE emType, void* pQueryCondition, void* reserved, int waittime);	
Parameter	[in] lLoginID	Login handle.
	[in] emType	Searched file type.
	[in] pQueryCondition	Searching conditions.
	[in] reserved	Reserve bytes.
	[in] waittime	Waiting time.
Return value	Success: the searching handle of LLONG type. Failure: 0.	
Note	None.	

9.5.2 CLIENT_GetTotalFileCount

Table 9-19 Get the number of searched files

Item	Description
Name	Get the number of searched files.

Item	Description	
Function	BOOL CLIENT_GetTotalFileCount(LLONG lFindHandle, int* pTotalCount, void * reserved, int waittime);	
Parameter	[in] lFindHandle	Searching handle.
	[out] pTotalCount	The searched number.
	[in] reserved	Reserve bytes.
	[in] waittime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.5.3 CLIENT_FindNextFileEx

Table 9-20 Search a file

Item	Description	
Name	Search a file.	
Function	int CLIENT_FindNextFileEx(LLONG lFindHandle, int nFilecount, void* pMediaFileInfo, int maxlen, void* reserved, int waittime);	
Parameter	[in] lFindHandle	Searching handle.
	[in] nFilecount	The searched file number.
	[out] pMediaFileInfo	File cache area.
	[in] maxlen	Search for cache size of file array.
	[in] reserved	Reserve bytes.
	[in] waittime	Timeout.
Return value	Success: File number. Failure: -1. Return 0 means search complete.	
Note	None.	

9.5.4 CLIENT_FindCloseEx

Table 9-21 Stop searching files

Item	Description	
Name	Stop searching files.	
Function	BOOL CLIENT_FindCloseEx(LLONG lFindHandle);	
Parameter	[in] lFindHandle	Searching handle.
Return value	Success: TRUE. Failure: FALSE.	

Item	Description
Note	None.

9.5.5 CLIENT_PlayBackByTimeEx2

Table 9-22 Start playing back the video

Item	Description	
Name	Start playing back the video.	
Function	LLONG CLIENT_PlayBackByTimeEx2(LLONG lLoginID, Int nChannelID, NET_IN_PLAY_BACK_BY_TIME_INFO* pstNetIn, NET_OUT_PLAY_BACK_BY_TIME_INFO* pstNetOut);	
Parameter	[in] lLoginID	Login handle.
	[in] nChannelID	Channel number.
	[in] pstNetIn	Playback input parameter.
	[out] pstNetOut	Playback output parameter.
Return value	Success: the playback handle of LLONG type. Failure: 0.	
Note	None.	

9.5.6 CLIENT_StopPlayBack

Table 9-23 Stop playing back the video

Item	Description	
Name	Stop playing back the video.	
Function	BOOL CLIENT_StopPlayBack(LLONG lPlayHandle);	
Parameter	[in] lPlayHandle	Playback handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.5.7 CLIENT_DownloadByTimeEx

Table 9-24 Start downloading video

Item	Description	
Name	Start downloading video.	
Function	LLONG CLIENT_PlayBackByTimeEx2(LLONG lLoginID, int nChannelID, NET_IN_PLAY_BACK_BY_TIME_INFO* pstNetIn, NET_OUT_PLAY_BACK_BY_TIME_INFO* pstNetOut);	
Parameter	[in] lLoginID	Login handle.
	[in] nChannelID	Channel number.

Item	Description	
	[in] pstNetIn	Playback input parameter.
	[out] pstNetOut	Playback output parameter.
Return value	Success: the playback handle of LLONG type. Failure: 0.	
Note	None.	

9.5.8 CLIENT_StopDownload

Table 9-25 Stop downloading the video

Item	Description	
Name	Stop downloading the video.	
Function	BOOL CLIENT_StopDownload(LLONG lFileHandle);	
Parameter	[in] lFileHandle	Playback handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.5.9 CLIENT_DownloadRemoteFile

Table 9-26 Download files by file name

Item	Description	
Name	Download files by file name.	
Function	BOOL CLIENT_DownloadRemoteFile(LLONG lLoginID, const DH_IN_DOWNLOAD_REMOTE_FILE* pInParam, DH_OUT_DOWNLOAD_REMOTE_FILE* pOutParam, int nWaitTime);	
Parameter	[in] lFindHandle	Playback handle.
	[in] pInParam	Download file input parameter.
	[out] pOutParam	Download file output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.6 Subscribing Face Event

For subscribing face event, see "9.4 Subscribing Intelligent Event."

9.7 Adding/Deleting/Modifying/Searching the Face Library

9.7.1 CLIENT_OperateFaceRecognitionGroup

Table 9-27 Add, delete and modify the face library

Item	Description	
Name	Add, delete and modify the face library.	
Function	BOOL CLIENT_OperateFaceRecognitionGroup(LLONG lLoginID, const NET_IN_OPERATE_FACERECONGNITION_GROUP* pstInParam, NET_OUT_OPERATE_FACERECONGNITION_GROUP *pstOutParam, int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.7.2 CLIENT_FindGroupInfo

Table 9-28 Searching the of face library

Item	Description	
Name	Searching the of face library.	
Function	BOOL CLIENT_FindGroupInfo(LLONG lLoginID, const NET_IN_FIND_GROUP_INFO* pstInParam, NET_OUT_FIND_GROUP_INFO *pstOutParam, int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8 Adding/Deleting/Modifying/Searching People Face

9.8.1 CLIENT_OperateFaceRecognitionDB

Table 9-29 Add, delete and modify the people face

Item	Description	
Name	Add, delete and modify the people face.	
Function	BOOL CLIENT_OperateFaceRecognitionDB(LLONG lLoginID, const NET_IN_OPERATE_FACERECONGNITIONDB* pstInParam, NET_OUT_OPERATE_FACERECONGNITIONDB *pstOutParam, Int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.2 CLIENT_OperateFaceRecognitionDB

Table 9-30 Set the searching conditions of people face

Item	Description	
Name	Set the searching conditions of people face.	
Function	BOOL CLIENT_StartFindFaceRecognition(LLONG lLoginID, const NET_IN_STARTFIND_FACERECONGNITION* pstInParam, NET_OUT_STARTFIND_FACERECONGNITION *pstOutParam, int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.3 CLIENT_DoFindFaceRecognition

Table 9-31 Search the face information

Item	Description
Name	Search the face information.

Item	Description	
Function	BOOL CLIENT_DoFindFaceRecognitionRecord(const NET_IN_DOFIND_FACERECONGNITIONRECORD* pstInParam, NET_OUT_DOFIND_FACERECONGNITIONRECORD *pstOutParam, int nWaitTime);	
Parameter	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.4 CLIENT_StopFindFaceRecognition

Table 9-32 Stop searching face information

Item	Description	
Name	Stop searching face information.	
Function	BOOL CLIENT_StopFindFaceRecognition(LLONG lFindHandle);	
Parameter	[in] lFindHandle	Searching handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.5 CLIENT_FaceRecognitionPutDisposition

Table 9-33 Arm by library

Item	Description	
Name	Arm by library.	
Function	BOOL CLIENT_FaceRecognitionPutDisposition(LLONG lLoginID, const NET_IN_FACE_RECOGNITION_PUT_DISPOSITION_INFO* pstInParam, NET_OUT_FACE_RECOGNITION_PUT_DISPOSITION_INFO *pstOutParam, int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.6 CLIENT_FaceRecognitionDelDisposition

Table 9-34 Disarm by library

Item	Description	
Name	Disarm by library.	
Function	BOOL CLIENT_FaceRecognitionDelDisposition(LLONG lLoginID, const NET_IN_FACE_RECOGNITION_DEL_DISPOSITION_INFO* pstInParam, NET_OUT_FACE_RECOGNITION_DEL_DISPOSITION_INFO *pstOutParam, int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.7 CLIENT_SetGroupInfoForChannel

Table 9-35 Arm by channel

Item	Description	
Name	Arm by channel.	
Function	BOOL CLIENT_SetGroupInfoForChannel(LLONG lLoginID, const NET_IN_SET_GROUPINFO_FOR_CHANNEL* pstInParam, NET_OUT_SET_GROUPINFO_FOR_CHANNEL *pstOutParam, int WaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.8.8 CLIENT_AttachFaceFindState

Table 9-36 Subscribe searching progress of people face

Item	Description
Name	Subscribe searching progress of people face.

Item	Description	
Function	LLONG CLIENT_AttachFaceFindState(LLONG lLoginID, const NET_IN_FACE_FIND_STATE* pstInParam, NET_OUT_FACE_FIND_STATE *pstOutParam, Int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input parameter.
	[out] pstOutParam	Output parameter.
	[in] nWaitTime	Timeout.
Return value	Success: Face progress handle. Failure: 0.	
Note	None.	

9.8.9 CLIENT_DetachFaceFindState

Table 9-37 Cancel subscribing the searching progress of people face

Item	Description	
Name	Cancel Subscribing the searching progress of people face.	
Function	BOOL CLIENT_DetachFaceFindState(LLONG lAttachHandle);	
Parameter	[in] lAttachHandle	The handle returned by CLIENT_AttachFaceFindState.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.9 Body Detection

9.9.1 CLIENT_DownloadRemoteFile

Table 9-38 Download the picture

Item	Description	
Name	Download the picture.	
Function	BOOL CLIENT_DownloadRemoteFile(LLONG lLoginID, const DH_IN_DOWNLOAD_REMOTE_FILE* pInParam, DH_OUT_DOWNLOAD_REMOTE_FILE* pOutParam, int nWaitTime = 1000);	
Parameter	[in] lLoginID	The handle returned by CLIENT_AttachFaceFindState.
	[in] pInParam	Input parameter.
	[out] pOutParam	Output parameter.
	[in] nWaitTime	Timeout.

Item	Description
Return value	Success: TRUE. Failure: FALSE.
Note	None.

9.10 People Flow Statistics

9.10.1 CLIENT_AttachVideoStatSummary

Table 9-39 Subscribe flow statistics event

Item	Description	
Name	Subscribe flow statistics event.	
Function	LLONG CLIENT_AttachVideoStatSummary(LLONG lLoginID, const NET_IN_ATTACH_VIDEOSTAT_SUM* pInParam, NET_OUT_ATTACH_VIDEOSTAT_SUM* pOutParam, int nWaitTime);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Subscribe input parameter of people flow.
	[out] pOutParam	Subscribe output parameter of people flow.
	[in] nWaitTime	Timeout.
Return value	Flow statistics subscribing handle.	
Note	None.	

9.10.2 CLIENT_DetachVideoStatSummary

Table 9-40 Cancel subscribing flow statistics event

Item	Description	
Name	Cancel subscribing flow statistics event.	
Function	BOOL CLIENT_DetachVideoStatSummary(LLONG lAttachHandle);	
Parameter	[in] lAttachHandle	Flow statistics subscribing handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.10.3 CLIENT_StartFindNumberStat

Table 9-41 Start searching people history data (set searching conditions)

Item	Description
Name	Start searching people history data (set searching conditions).

Item	Description	
Function	LLONG CLIENT_StartFindNumberStat(LLONG lLoginID, NET_IN_FINDNUMBERSTAT* pstInParam, NET_OUT_FINDNUMBERSTAT* pstOutParam);	
Parameter	[in] lLoginID	Login handle.
	[in] pstInParam	Input searching conditions.
	[out] pstOutParam	Output query result.
Return value	Searching handle.	
Note	None.	

9.10.4 CLIENT_DoFindNumberStat

Table 9-42 Start searching people history data (Set searching conditions)

Item	Description	
Name	Start searching people history data (Set searching conditions).	
Function	int CLIENT_DoFindNumberStat(LLONG lFindHandle, NET_IN_DOFINDNUMBERSTAT* pstInParam, NET_OUT_DOFINDNUMBERSTAT* pstOutParam);	
Parameter	[in] lLoginID	Login handle.
	[in] plnParam	Searching input parameter.
	[out] pstOutParam	Searching output parameter.
Return value	Searching number.	
Note	None.	

9.10.5 CLIENT_StopFindNumberStat

Table 9-43 Stop searching history data

Item	Description	
Name	Stop searching history data.	
Function	BOOL CLIENT_StopFindNumberStat(LLONG lFindHandle);	
Parameter	[in] lFindHandle	Searching handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.11 Intelligent Traffic

9.11.1 CLIENT_FindRecord

Table 9-44 Start searching data (Set searching conditions)

Item	Description	
Name	Start searching data (Set searching conditions).	
Function	BOOL CLIENT_FindRecord(LLONG lLoginID, NET_IN_FIND_RECORD_PARAM* pInParam, NET_OUT_FIND_RECORD_PARAM* pOutParam, int waittime=1000);	
Parameter	[in] lLoginID	Login handle.
	[in] pInParam	Input searching conditions.
	[out] pOutParam	Output query result.
Return value	Searching handle.	
Note	None.	

9.11.2 CLIENT_QueryRecordCount

Table 9-45 Search the total number of data

Item	Description	
Name	Search the total number of data.	
Function	BOOL CLIENT_QueryRecordCount(NET_IN_QUEYT_RECORD_COUNT_PARAM* pInParam, NET_OUT_QUEYT_RECORD_COUNT_PARAM* pOutParam, int waittime=1000);	
Parameter	[in] pInParam	Searching input parameter.
	[out] pOutParam	Searching output parameter.
	[in] waittime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.11.3 CLIENT_FindNextRecord

Table 9-46 Search the data of specified number

Item	Description
Name	Search the data of specified number.
Function	int CLIENT_FindNextRecord(NET_IN_FIND_NEXT_RECORD_PARAM* pInParam, NET_OUT_FIND_NEXT_RECORD_PARAM* pOutParam, int waittime=1000);

Item	Description	
Parameter	[in] pstInParam	Searching input parameter.
	[out] pstOutParam	Searching output parameter.
	[in] waittime	Timeout.
Return value	Searching number.	
Note	None.	

9.11.4 CLIENT_FindRecordClose

Table 9-47 Stop searching vehicle flow

Item	Description	
Name	Stop searching vehicle flow.	
Function	BOOL CLIENT_FindRecordClose(LLONG lFindHandle);	
Parameter	[in] lFindHandle	Searching handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.11.5 CLIENT_OperateTrafficList

Table 9-48 Add, delete and modify the blocklist and trusted list

Item	Description	
Name	Add, delete and modify the blocklist and trusted list.	
Function	BOOL CLIENT_OperateTrafficList(LLONG lLoginID , NET_IN_OPERATE_TRAFFIC_LIST_RECORD* pstInParam , NET_OUT_OPERATE_TRAFFIC_LIST_RECORD *pstOutParam , int waittime)	
Parameter	[in] lLoginID	Login handle.
	[in] pstInParam	Input parameter for blocklist and trusted list operation.
	[out] pstOutParam	Output parameter for blocklist and trusted list operation.
	[in] waittime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	NET_TRAFFIC_LIST_INSERT// Add record NET_TRAFFIC_LIST_UPDATE// Update record NET_TRAFFIC_LIST_REMOVE// Delete record	

9.11.6 CLIENT_DownLoadMultiFile

Table 9-49 Download files in batches

Item	Description
Name	Download files in batches.

Item	Description	
Function	BOOL CLIENT_DownLoadMultiFile(LLONG lLoginID, NET_IN_DOWNLOAD_MULTI_FILE *pstInParam, NET_OUT_DOWNLOAD_MULTI_FILE *pstOutParam, int waittime=1000);	
Parameter	[in] lLoginID	Login handle.
	[in] pstInParam	Searching input parameter.
	[out] pstOutParam	Searching output parameter.
	[in] waittime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.11.7 CLIENT_StopLoadMultiFile

Table 9-50 Stop downloading files in batches

Item	Description	
Name	Stop downloading files in batches.	
Function	BOOL CLIENT_StopLoadMultiFile(LLONG lDownLoadHandle);	
Parameter	[in] lDownLoadHandle	Download handle in batches.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.12 Access Control

9.12.1 CLIENT_FindRecord

Table 9-51 Start searching data (Set searching conditions)

Item	Description	
Name	Start searching data (Set searching conditions).	
Function	BOOL CLIENT_FindRecord(LLONG lLoginID, NET_IN_FIND_RECORD_PARAM* pInParam, NET_OUT_FIND_RECORD_PARAM* pOutParam, int waittime=1000);	
Parameter	[in] lLoginID	Login handle.
	[int] pInParam	Input searching conditions.
	[out] pOutParam	Output query result.
Return value	Searching handle.	
Note	None.	

9.12.2 CLIENT_FindNextRecord

Table 9-52 Search the data of specified number

Item	Description	
Name	Search the data of specified number.	
Function	int CLIENT_FindNextRecord(NET_IN_FIND_NEXT_RECORD_PARAM* pInParam, NET_OUT_FIND_NEXT_RECORD_PARAM* pOutParam, int waittime=1000);	
Parameter	[int] pstInParam	Searching input parameter.
	[out] pstOutParam	Searching output parameter.
	[in] waittime	Timeout.
Return value	Searching number.	
Note	None.	

9.12.3 CLIENT_FindRecordClose

Table 9-53 Stop searching

Item	Description	
Name	Stop searching.	
Function	BOOL CLIENT_FindRecordClose(LLONG lFindHandle);	
Parameter	[in] lFindHandle	Searching handle.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.12.4 CLIENT_FindRecordClose

Table 9-54 Operate records of personnel and access conmtrol

Item	Description	
Name	You can add, delete, modify, search and clean up the people information. You can also delete and clean up access control record.	
Function	BOOL CLIENT_ControlDevice(LLONG lLoginID , CtrlType type , void *param , int waittime = 1000);	
Parameter	[in] lLoginID	Login handle.
	[in] type	Control type.
	[in] param	Control parameter, according to different types.
	[in] waittime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

9.12.5 CLIENT_FindRecordClose

Table 9-55 Add, delete, modify and clean up the information of face picture

Item	Description	
Name	Add, delete, modify and clean up the information of face picture.	
Function	BOOL CLIENT_FaceInfoOpreate(LLONG lLoginID, EM_FACEINFO_OPREATE_TYPE emType, void* pInParam, void* pOutParam, int nWaitTime = 1000);	
Parameter	[in] lLoginID	Login handle.
	[in] emType	Control type.
	[in] pInParam	Control parameter, select different structures according to different types.
	[out] pOutParam	Return parameter, select different structures according to different types.
	[in] waittime	Timeout.
Return value	Success: TRUE. Failure: FALSE.	
Note	None.	

10 Callback Function Definition

10.1 fDisConnect

Table 10-1 Disconnection callback

Item	Description	
Name	Disconnection callback.	
Function	typedef void (CALLBACK *fDisConnect)(LLONG lLoginID, char* pchDVRIP, LONG nDVRPort, LDWORD dwUser);	
Parameter	[out] lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity.
	[out] pchDVRIP	Device IP.
	[out] nDVRPort	Device port.
	[out] dwUser	User parameter of callback.
Return value	None.	
Note	None.	

10.2 fHaveReConnect

Table 10-2 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	Device IP.
	[out] nDVRPort	Device port.
	[out] dwUser	User parameter of callback.
Return value	None.	
Note	None.	

10.3 fRealDataCallBackEx

Table 10-3 Callback of real-time monitoring data

Item	Description	
Name	The callback of real-time monitoring data.	
Function	typedef void (CALLBACK *fRealDataCallBackEx)(LLONG lRealHandle, DWORD dwDataType, BYTE* pBuffer, DWORD dwBufSize, LONG param, LDWORD dwUser);	
Parameter	[out] lRealHandle	Return value of CLIENT_RealPlayEx.
	[out] dwDataType	Data type, 0-original data, 2-YUV data.
	[out] pBuffer	Monitor block data address.
	[out] dwBufSize	Monitor the length of block data, unit: byte.
	[out] param	Callback data parameter structure, the types are different according to different value of dwDataType. ● When dwDataType is 0, param is the empty pointer. ● When dwDataType is 2, param is the tagCBYUVDataParam structure pointer.
	[out] dwUser	User parameter of callback.
Return value	None.	
Note	None.	

10.4 fAnalyzerDataCallBack

Table 10-4 Intelligent event callback

Item	Description	
Name	Intelligent event callback.	
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	Intelligent event type.
	[out] pAlarmInfo	Event information cache.

Item	Description	
	[out] pBuffer	Picture cache.
	[out] dwBufSize	Picture cache size.
	[out] dwUser	User data.
	[out] nSequence	● nSequence is 0, it means the first time appears. ● nSequence is 2, it means only appears once or the last time appears. ● nSequence is 1, it means after this time there is always one or multiple times.
	[out] reserved	Reserve.
Return value	None.	
Note	None.	

10.5 fDownLoadPosCallBack

Table 10-5 Callback of playback and download by file

Item	Description	
Name	Playback or download progress calling back function by files.	
Function	typedef void (CALLBACK *fDownLoadPosCallBack)(LLONG lPlayHandle, DWORD dwTotalSize, DWORD dwDownLoadSize, LDWORD dwUser);	
Parameter	[out]lPlayHandle	Playback or download return value of interface.
	[out]dwTotalSize	Total size, unit: KB.
	[out]dwDownLoadSize	Downloaded size, unit: KB. ● -1: This time playback complete. ● -2: Write file failed.
	[out]dwUser	User data.
Return value	None.	
Note	None.	

10.6 fDataCallBack

Table 10-6 Callback of playback and download data

Item	Description
Name	Playback or download data callback function.

Item	Description	
Function	typedef int (CALLBACK *fDataCallBack)(LLONG lRealHandle, DWORD dwDataType, BYTE *pBuffer, DWORD dwBufSize, LDWORD dwUser);	
Parameter	[out]lPlayHandle	Playback or download return value of interface.
	[out] dwDataType	Is 0 (original data).
	[out] pBuffer	Data cache.
	[out] dwBufSize	cache length, unit: byte.
	[out] dwUser	User data.
Return value	None.	
Note	None.	

10.7 fFaceFindState

Table 10-7 Callback of face searching progress

Item	Description	
Name	Face searching progress callback function.	
Function	typedef void (CALLBACK *fFaceFindState)(LLONG lLoginID, LLONG lAttachHandle, NET_CB_FACE_FIND_STATE* pstStates, int nStateNum, LDWORD dwUser);	
Parameter	[out] lLoginID	Return login handle.
	[out] lAttachHandle	Event subscribing handle.
	[out] pstStates	Status information of searching the people face.
	[out] nStateNum	Searching progress of people face.
	[out] dwUser	User data.
Return value	None.	
Note	None.	

10.8 fVideoStatSumCallBack

Table 10-8 Callback of subscribing people flow event

Item	Description
Name	People flow event subscribing callback.

Item	Description	
Function	typedef void (CALLBACK *fVideoStatSumCallBack) (LLONG lAttachHandle, NET_VIDEOSTAT_SUMMARY* pBuf, DWORD dwBufLen, LDWORD dwUser);	
Parameter	[out] lAttachHandle	Flow statistics subscribing handle.
	[out] pBuf	Flow statistics return data.
	[out] dwBufLen	Data Length returned.
	[out] dwUser	User data.
Return value	None.	
Note	None.	

10.9 fMultiFileDownLoadPosCB

Table 10-9 Download file progress callback function in batches.

Item	Description	
Name	Download file progress callback function in batches.	
Function	typedef void (CALLBACK *fMultiFileDownLoadPosCB)(LLONG lDownLoadHandle, DWORD dwID, DWORD dwFileTotalSize, DWORD dwDownLoadSize, int nError, LDWORD dwUser, void* pReserved);	
Parameter	[out] lDownLoadHandle	Download file handle in batches.
	[out] dwID	ID is dwFileID set by user.
	[out] dwFileTotalSize	Total size of downloaded file.
	[out] dwDownLoadSize	File size that have downloaded, when this value is the max value, it means download completed.
	[out] nError	Download error: 1-cache lack, 2-check error for return data, 3-download the current file failed, 4-create file failed.
	[out] dwUser	User data.
	[out] pReserved	Reserve bytes.
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 device network security:

1. **Use Strong Passwords**

   Please refer to the following suggestions to set passwords:

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

2. **Update Firmware and Client Software in Time**

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

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

1. **Physical Protection**

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

2. **Change Passwords Regularly**

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

3. **Set and Update Passwords Reset Information Timely**

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

4. **Enable Account Lock**

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

5. **Change Default HTTP and Other Service Ports**

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

6. **Enable HTTPS**

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

7. **MAC Address Binding**

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

8. **Assign Accounts and Privileges Reasonably**

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

9. **Disable Unnecessary Services and Choose Secure Modes**

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

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

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

10. **Audio and Video Encrypted Transmission**

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

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

11. **Secure Auditing**

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

12. **Network Log**

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

13. **Construct a Safe Network Environment**

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

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