NetSDK_JAVA (Intelligent Event)

Programming Manual

V1.0.0

Foreword

Safety Instructions

The following categorized signs and 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 No.	Revision Content	Release Date
V1.0.0	First release.	October 2020

Glossary

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

Term	Explanation
Face detection	Detect the faces and their feature information (age, gender, and expression) through the intelligent analysis of videos.
Face recognition	Detect whether the faces are in the armed face database through the intelligent analysis of videos, including face detection.
Face database	Detect whether the faces are in the face database in real time by importing some face images into IVSS, NVR, camera IPC, and other devices in advance.
ITC	Intelligent Traffic Camera, which can capture vehicle images and automatically analyze traffic events.
Tripwire detection	Detection of crossing the warning line.
Intrusion detection	Detection of objects intruding into the warning zone, including "Crossing region" and "In the region".
People counting	Number of people in the camera calibration region.

Table of Contents

1 Overview

1.1 General

This manual mainly introduces the reference information for SDK intrefaces , including main functions, interface functions, and callback.

The main functions include general functions, face detection, face recognition, general behavior event, human detection, thermal temperature measurement, access control event, people counting statistics, intelligent traffic, and person and ID card comparison.

- For files included in Windows , see Table 1-1.

Table 1-1 Files in the Windows packaging

Library type	Library file name	Library file description
Function library	dhnetsdk.h	Header file
	dhnetsdk.lib	Lib file
	dhnetsdk.dll	Library file
	avnetsdk.dll	Library file
Configuration library	avglobal.h	Header file
	dhconfigsdk.h	Header file
	dhconfigsdk.lib	Lib file
	dhconfigsdk.dll	Library file
Play (encoding/decoding) auxiliary library	dhplay.dll	Play library
Auxiliary library of avnetsdk.dll	Infra.dll	Function auxiliary library
	json.dll	Function auxiliary library
	NetFramework.dll	Function auxiliary library
	Stream.dll	Function auxiliary library
	StreamSvr.dll	Function auxiliary library

- For files included in the Linux packaging, see Table 1-2.

Table 1-2 Files in the Linux packaging

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 libavnetsdk.so	libInfra.so	Function auxiliary library
	libJson.so	Function auxiliary library
	libNetFramework.so	Function auxiliary library
	libStream.so	Function auxiliary library

- The function library and configuration library of NetSDK are necessary libraries.

- The function library is the main body of NetSDK, which is used for communication interaction between client and products, remote control, search, configuration, acquisition and processing of stream data.
- Configuration library packs and parses according to the structural body of the configuration function.
- It is recommended use play library to parse stream and play.
- If the function library includes avnetsdk.dll or libavnetsdk.so, the corresponding auxiliary library is required.

1.2 Applicability

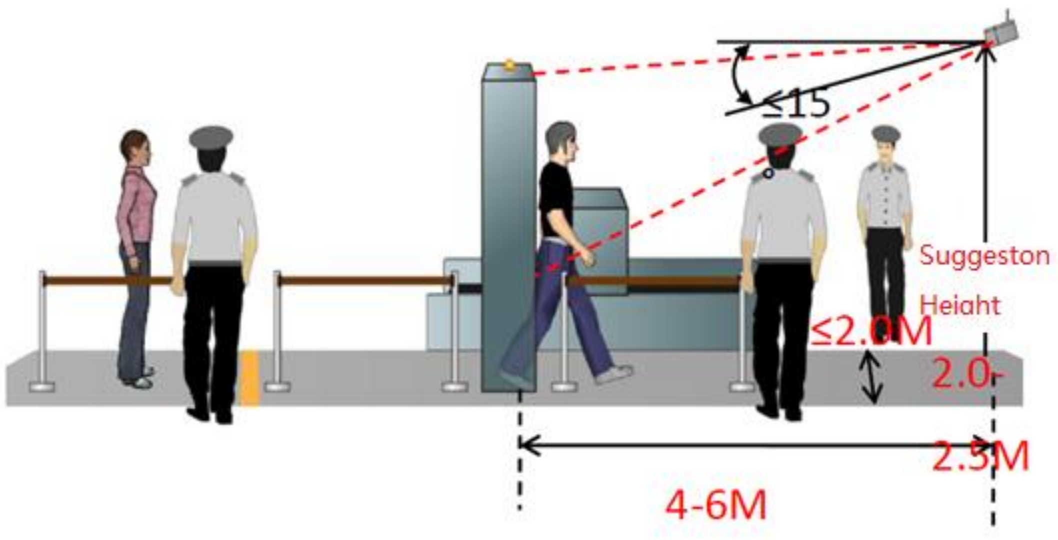
- Recommended memory: No less than 512 M.
- Jdk version: jdk1.6; jdk1.8.
- Systems supported by SDK:
  - ◇ Windows 10/Windows 8.1/Windows 7/ 2000 and Windows Server 2008/2003.
  - ◇ Linux
    General Linux system like Red Hat/SUSE

1.3 Application Scenarios

1.3.1 Face Detection/Face Recognition/Human Detection

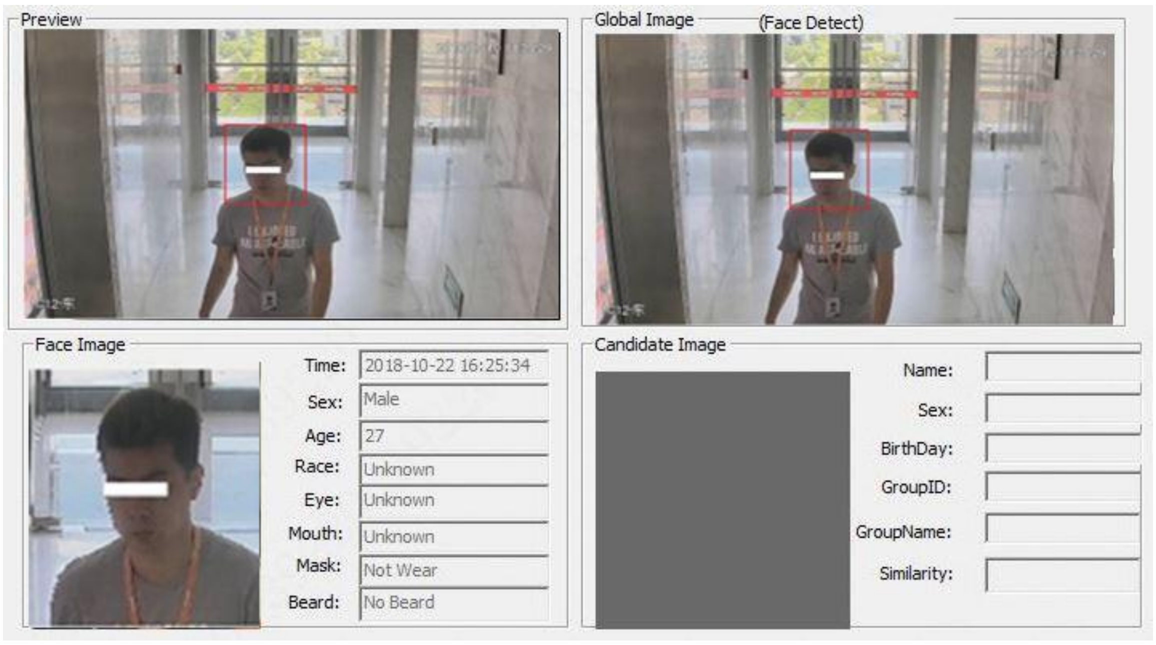
For the application scenarios of face detection, face recognition, and human recognition devices, see Figure 1-1.

Figure 1-1 Face recognition



For the face detection scenario, see Figure 1-2.

Figure 1-2 Face detection scenario

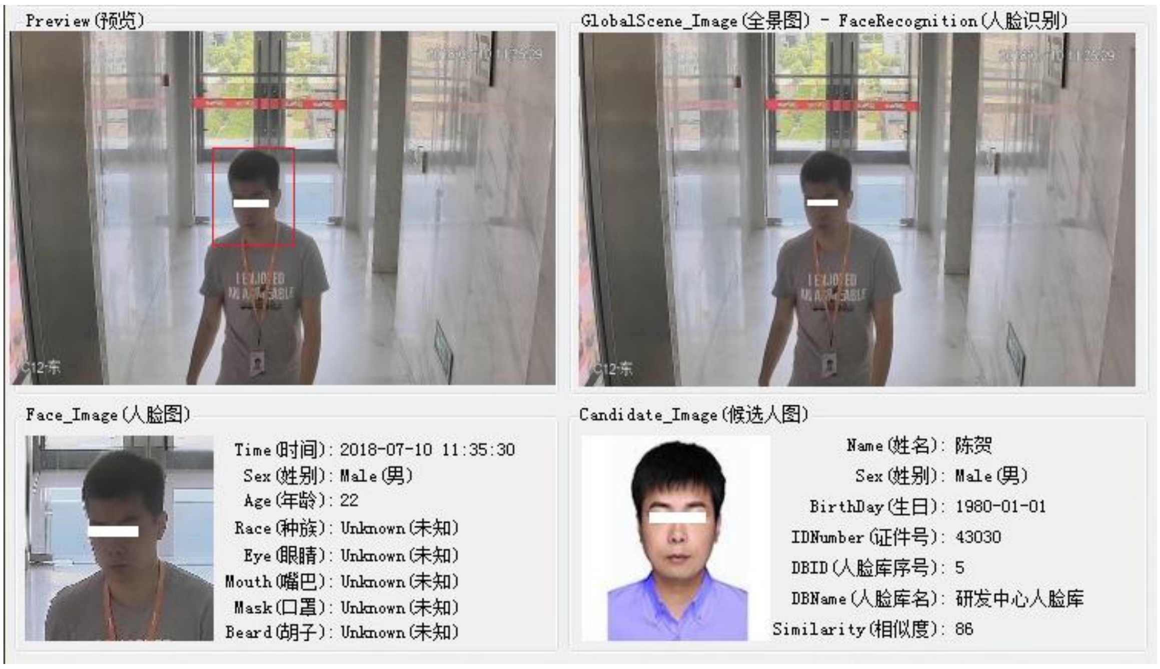


Time:	2018-10-22 16:25:34
Sex:	Male
Age:	27
Race:	Unknown
Eye:	Unknown
Mouth:	Unknown
Mask:	Not Wear
Beard:	No Beard

Name:	
Sex:	
BirthDay:	
GroupID:	
GroupName:	
Similarity:	

For the face recognition scenario, see Figure 1-3.

Figure 1-3 Face recognition scenario

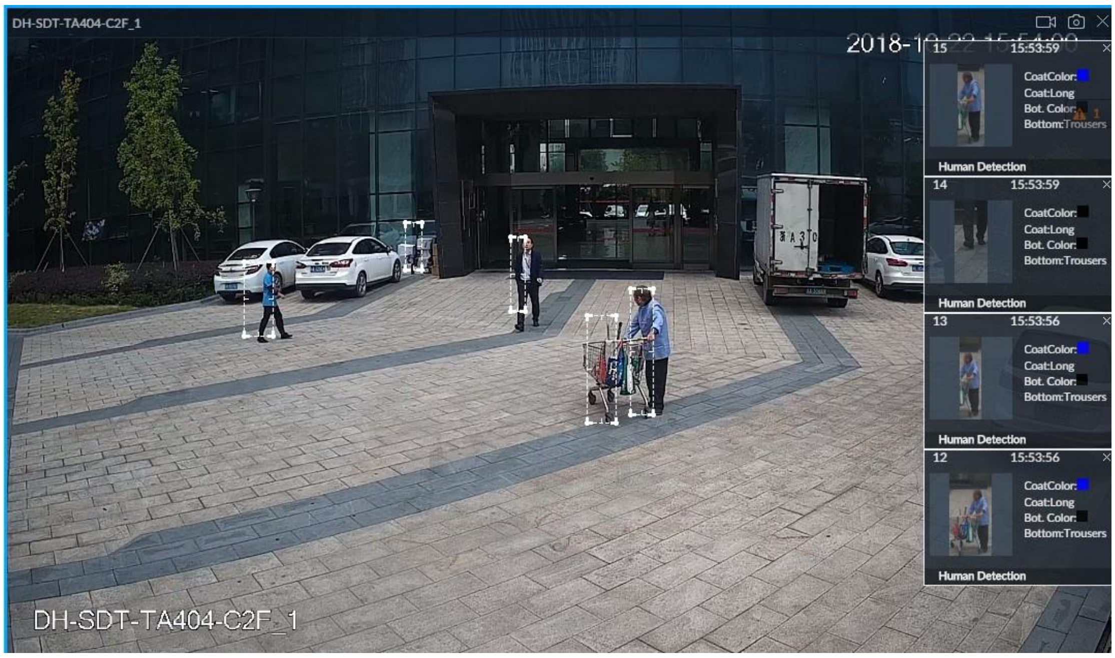

Preview(预览)
GlobalScene_Image(全景图) - FaceRecognition(人脸识别)
Face_Image(人脸图)
Candidate_Image(候选人图)

Time(时间):	2018-07-10 11:35:30
Sex(姓别):	Male(男)
Age(年龄):	22
Race(种族):	Unknown(未知)
Eye(眼睛):	Unknown(未知)
Mouth(嘴巴):	Unknown(未知)
Mask(口罩):	Unknown(未知)
Beard(胡子):	Unknown(未知)

Name(姓名):	陈贺
Sex(姓别):	Male(男)
BirthDay(生日):	1980-01-01
IDNumber(证件号):	43030
DBID(人脸库序号):	5
DBName(人脸库名):	研发中心人脸库
Similarity(相似度):	86

For the human detection scenario, see Figure 1-4.

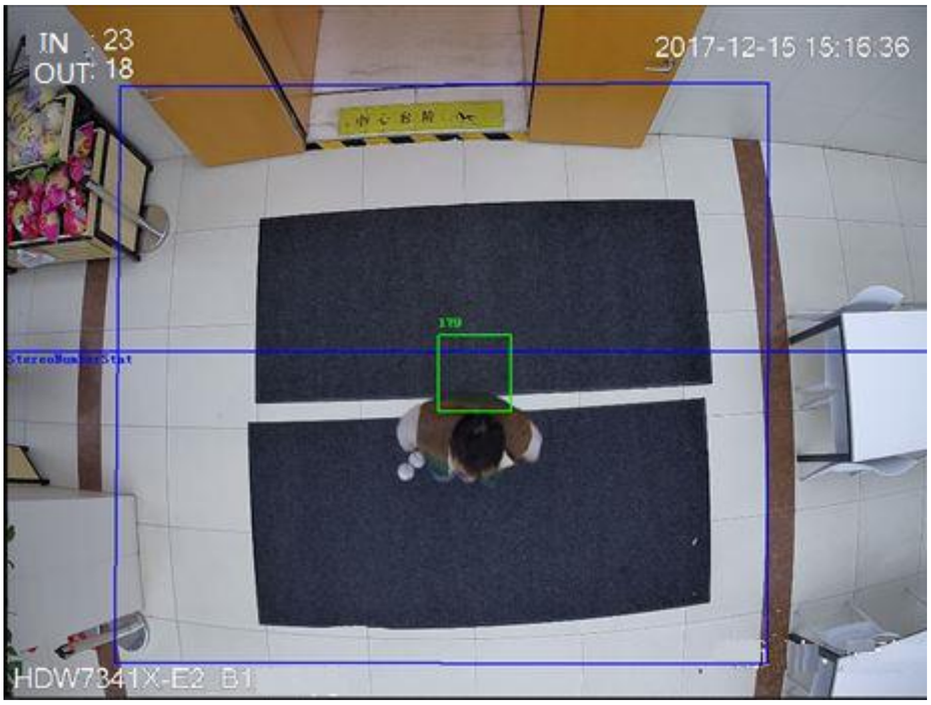
Figure 1-4 Human detection scenario



1.3.2 People Counting

For the application of people counting products in the actual scenario, see Figure 1-5.

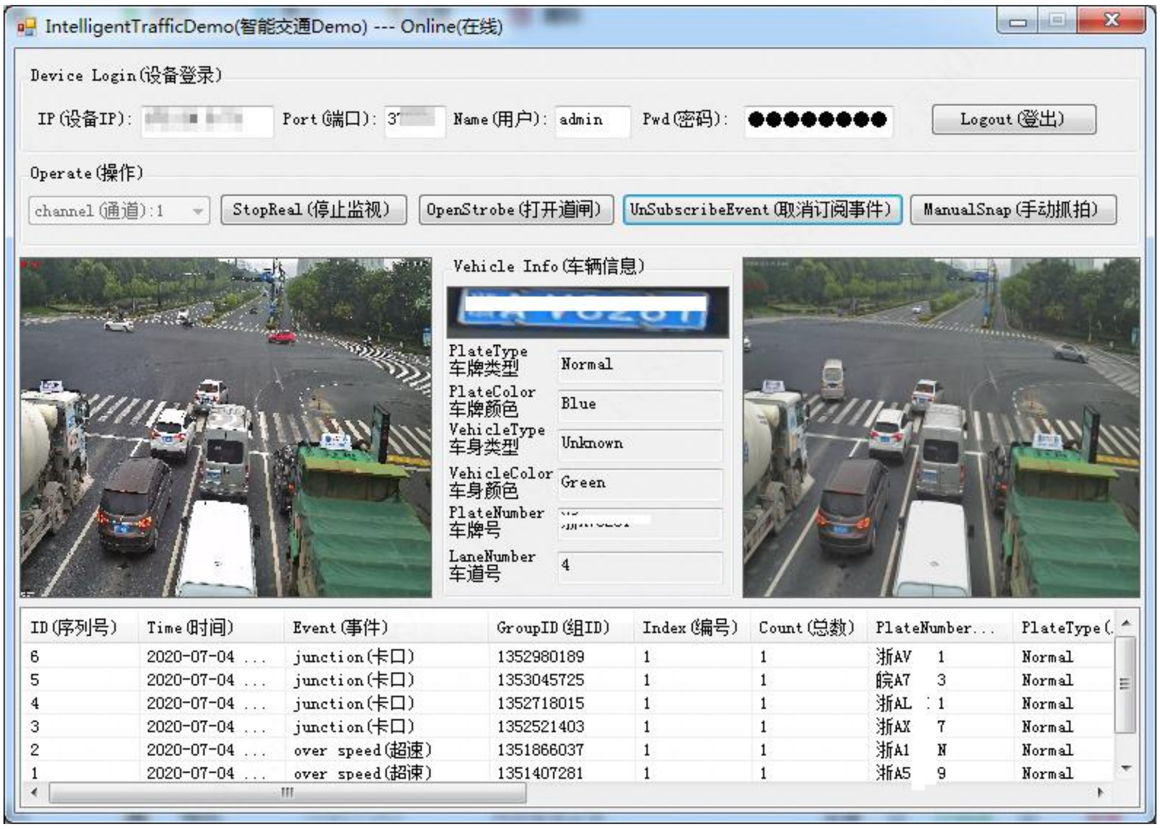
Figure 1-5 People counting scenario



1.3.3 Intelligent Traffic

- ITC at the intersection is used for capturing traffic violations and traffic flow statistics, see Figure 1-6.

Figure 1-6 Applications of ITC at the intersection



IntelligentTrafficDemo(智能交通Demo) --- Online(在线)

Device Login(设备登录)

IP(设备IP): [redacted] Port(端口): 37[redacted] Name(用户): admin Pwd(密码): ●●●●●●●● Logout(登出)

Operate(操作)

channel(通道):1 StopReal(停止监视) OpenStrobe(打开道闸) UnSubscribeEvent(取消订阅事件) ManualSnap(手动抓拍)

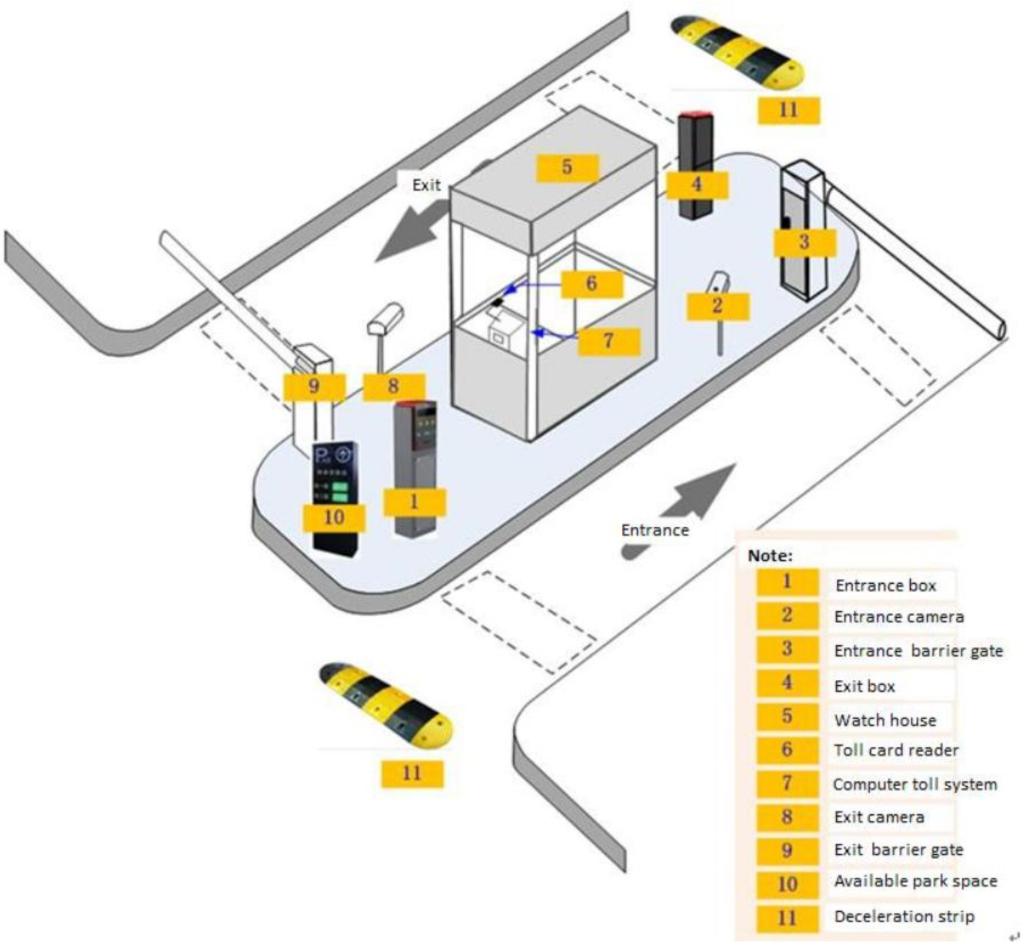
Vehicle Info(车辆信息)

PlateType 车牌类型	Normal
PlateColor 车牌颜色	Blue
VehicleType 车身类型	Unknown
VehicleColor 车身颜色	Green
PlateNumber 车牌号	[redacted]
LaneNumber 车道号	4

ID(序列号)	Time(时间)	Event(事件)	GroupID(组ID)	Index(编号)	Count(总数)	PlateNumber...	PlateType(.
6	2020-07-04 ...	junction(卡口)	1352980189	1	1	浙AV [redacted] 1	Normal
5	2020-07-04 ...	junction(卡口)	1353045725	1	1	皖A7 [redacted] 3	Normal
4	2020-07-04 ...	junction(卡口)	1352718015	1	1	浙AL [redacted] 1	Normal
3	2020-07-04 ...	junction(卡口)	1352521403	1	1	浙AX [redacted] 7	Normal
2	2020-07-04 ...	over speed(超速)	1351866037	1	1	浙A1 [redacted] N	Normal
1	2020-07-04 ...	over speed(超速)	1351407281	1	1	浙A5 [redacted] 9	Normal

- ITC at the entrance and exit of the parking lot is used for controlling vehicles for entering and exiting the parking lot and monitoring whether there are parking spaces available. See Figure 1-7.

Figure 1-7 Applications of ITC at the entrance and exit of the parking lot



1.3.4 General Behavior

Corresponding alarm event is triggered when a person or vehicle crosses the rule line (tripwire) or intrudes into the warning zone (intrusion), while the target object (person or vehicle) can be distinguished.

For the application scenarios of general behaviors, see Figure 1-8 and Figure 1-9.

Figure 1-8 General behavior scenario—tripwire

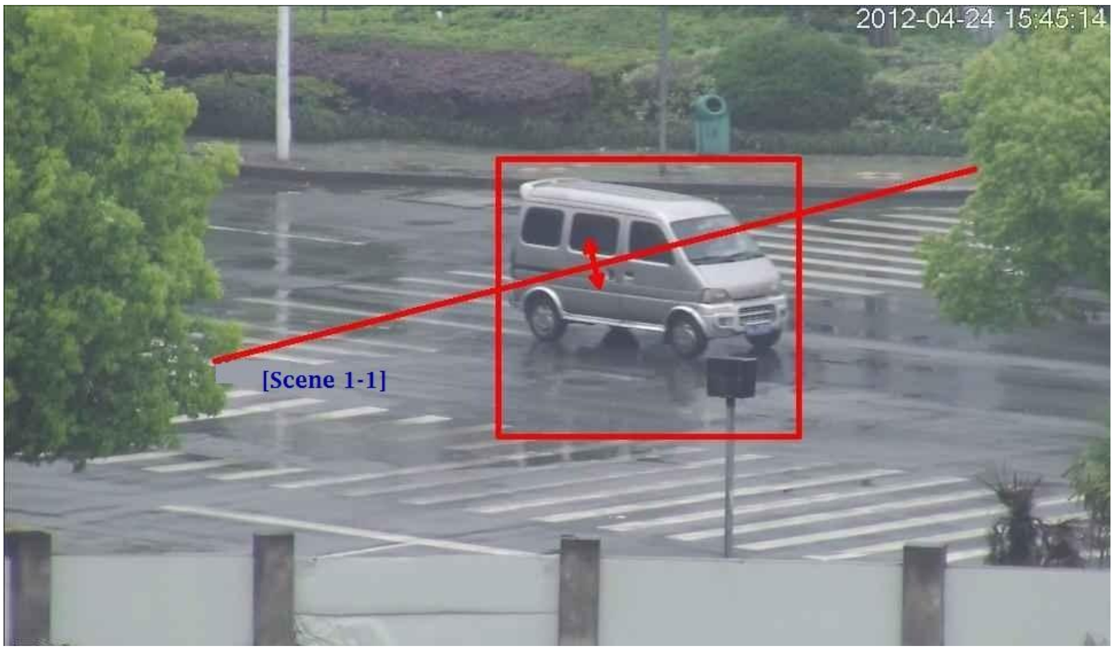
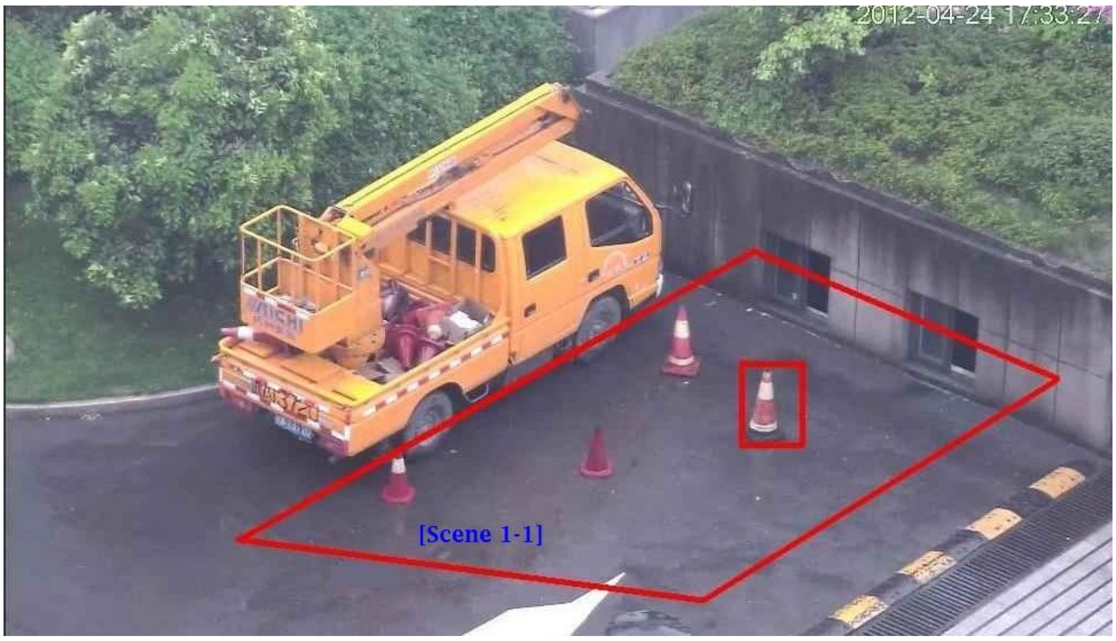


Figure 1-9 General behavior scenario—intrusion

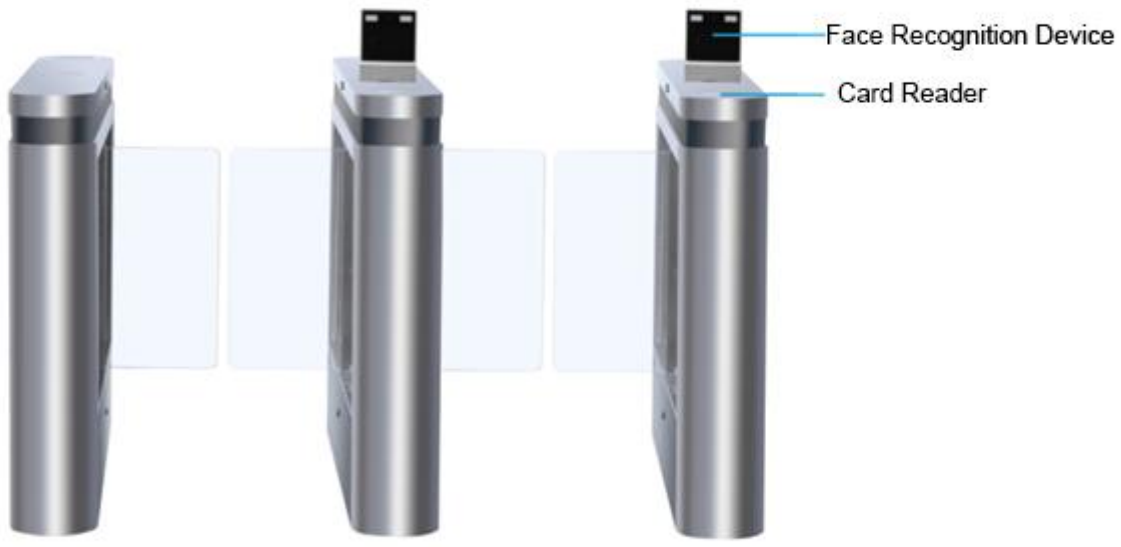


1.3.5 Turnstile

The access control turnstile is mainly applied in parks, scenic areas, schools, residence areas, and office buildings. After the collected face images and personnel information is uploaded to the platform, the platform issues the data to the turnstile system.

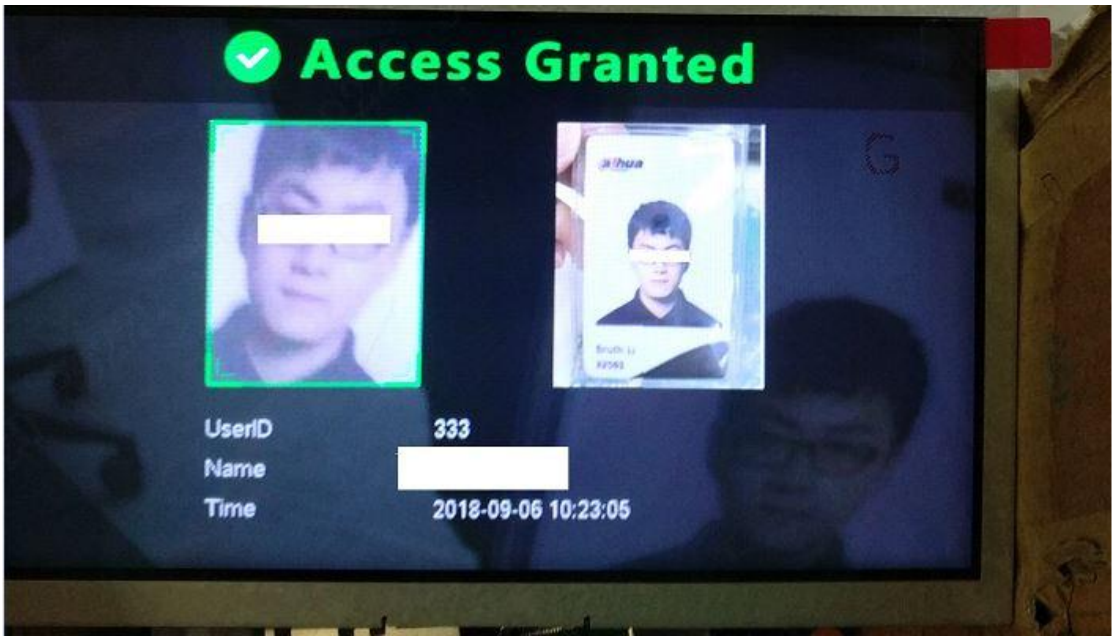
For the appearance of the access control turnstile, see Figure 1-10.

Figure 1-10 Swing turnstile appearance



You can unlock the turnstile by face or swiping card. For unlocking by face, see Figure 1-11.

Figure 1-11 Unlocking by face


Access Granted

UserID 333
Name [redacted]
Time 2018-09-06 10:23:05

2 General Functions

2.1 NetSDK 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 initialization, call the SDK cleaning up interface to release resource.

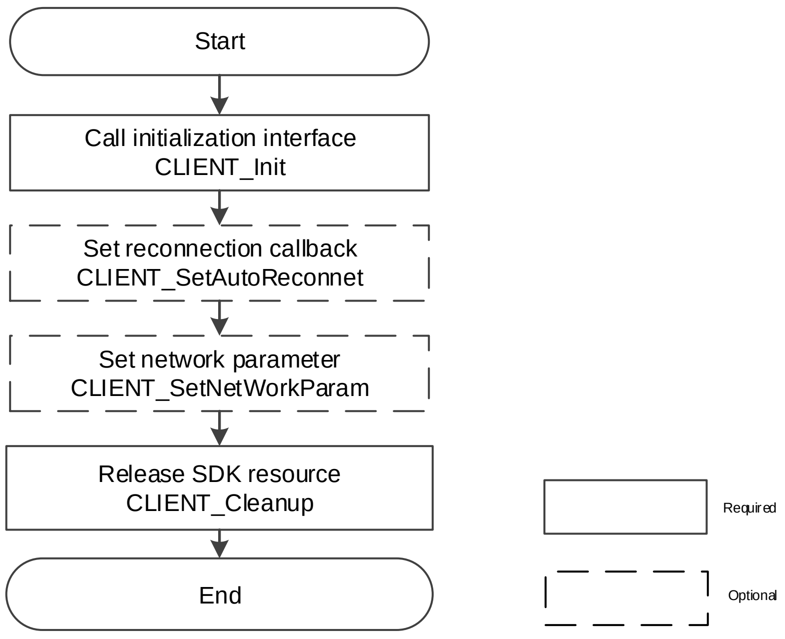
2.1.2 Interface Overview

Table 2-1 Description of SDK initialization interfaces

Interface	Description
CLIENT_Init	SDK initialization interface
CLIENT_Cleanup	SDK cleaning up interface
CLIENT_SetAutoReconnect	Setting of reconnection callback interface
CLIENT_SetNetworkParam	Setting of login network environment interface

2.1.3 Process Description

Figure 2-1 Process of SDK initialization



Process Description

<u>Step 1</u> Call **CLIENT_Init** to initialize SDK.

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

Step 3 (Optional) Call **CLIENT_SetNetworkParam** to set network login parameter that includes the timeout period for device login and the number of attempts.

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

Notes

- You need to call the interfaces CLIENT_Init and CLIENT_Cleanup in pairs. It supports single-thread multiple calling in pairs, but it is suggested to call for only once overall.
- Initialization: Internally calling the interface 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 and playback function modules can be restored after reconnection.

2.1.4 Sample Code

```
import java.io.File;

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

import com.sun.jna.ptr.IntByReference;

/**
 * login interface realization
 * mainly includes: initialization, login, logout
 */
public class LoginModule {

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

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

                        private static boolean bInit       = false;
                        private static boolean bLogopen = false;

                        //initialization
```

```
		public static boolean init(NetSDKLib.fDisConnect disConnect,
		NetSDKLib.fHaveReConnect haveReConnect) {
			bInit = netsdk.CLIENT_Init(disConnect, null);
			if(!bInit) {
				System.out.println("Initialize SDK failed");
				return false;
			}

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

			// configure reconnection callback interface and if device s are
		disconnected, SDK will connect the devices again automatically
			// it is optional but we recommending configuring this for your
		device
			netsdk.CLIENT_SetAutoReconnect(haveReConnect, null);

			//configure login timeout duration and attempts, optional
			int waitTime = 5000; //login request respose time is configured
		5 s.
			int tryTimes = 1;	//try to establish connection once during
		login
			netsdk.CLIENT_SetConnectTime(waitTime, tryTimes);

			// configure more network parameters, nWaittime of
		NET_PARAM, nConnectTryNum member and
		CLIENT_SetConnectTime
			// the meaning of device loin timeout duration config and
		attempt config are the same, optional
```

```
                NetSDKLib.NET_PARAM netParam = new
        NetSDKLib.NET_PARAM();
                netParam.nConnectTime = 10000;          //timeout duration of
        tringy to establish connection when login.
                netParam.nGetConnInfoTime = 3000;     // timeout duration of
        configuring sub connection.
                netsdk.CLIENT_SetNetworkParam(netParam);

                return true;
        }

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

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

2.2 Device Login

2.2.1 Introduction

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

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

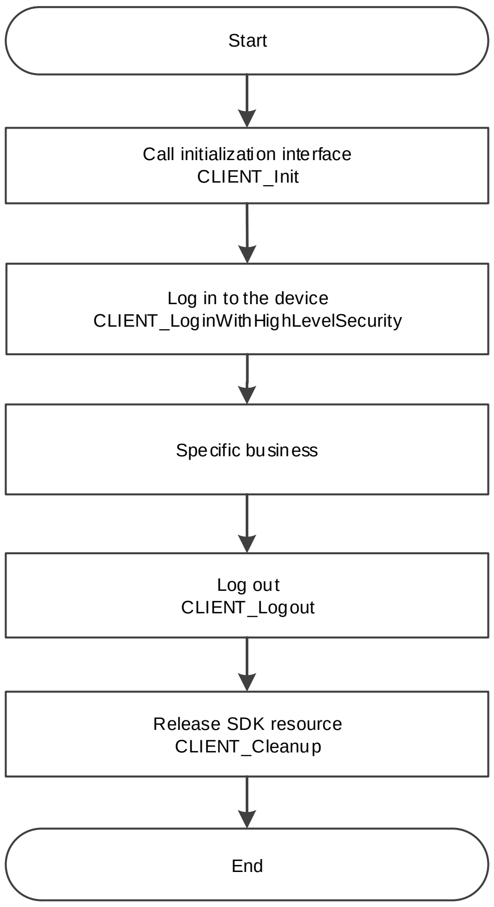
2.2.2 Interface Overview

Table 2-2 Description of device login interfaces

Interface	Description
CLIENT_LoginWithHighLevelSecurity	High security level login interface
CLIENT_Logout	Logout interface

2.2.3 Process Description

Figure 2-2 Login process



Process Description

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

Notes

- Login handle: When the login is successful, the returned value of the interface is not 0 (even the handle is smaller than 0, the login is also successful). One device can log in multiple times with different handle at each login. If there is not special function module, it is suggested to log in only one time. The login handle can be repeatedly used on other function modules.
- Handle repetition: The login handle might be the same as an existing handle, which is normal. For example, if you log in to Device A and get loginIDA, then cancel loginIDA. When you log in again, you might get LoginIDA again. However, throughout the life cycle of a handle, the same handle will not appear.

- Logout: The interface will release the opened functions in the login session internally, but it is not suggested to rely on the cleaning up function of the logout interface. For example, if you enable the monitoring function, you should call the interface that disables the monitoring function when it is no longer required.
- Use login and logout in pairs: The login consumes some memory and socket information and releases sources once logged out.
- Login failure: It is suggested to check the failure through the error parameter (login error code) of the login interface. For common error codes, see Table 2-3.
- Log in to multiple devices: After SDK initialization, you can log in to multiple devices, but the corresponding login handle and information need to be adjusted.

Table 2-3 Common error codes and meanings

Error Code	Meanings
1	Incorrect password.
2	User name does not exist.

3	Login timeout. The example code to avoid this error is as follows:
	`NET_PARAM stuNetParam = new NET_PARAM();`
	`stuNetParam.nWaittime = 8000; // unit ms`
	`CLIENT_SetNetworkParam (stuNetParam);`

4	The account has been logged in.
5	The account has been locked.
6	The account is listed in blocklist.
7	Out of resources, the system is busy.
8	Sub-connection failed.
9	Primary connection failed.
10	Exceeded the maximum number of user connections.
11	Lack of avnetsdk or avnetsdk dependent library
12	USB flash disk is not inserted into device, or the USB flash disk information error.
13	The client IP address is not authorized with login.

2.2.4 Sample Code

```
import java.io.File;

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

import com.sun.jna.ptr.IntByReference;

public class LoginModule {

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

```
                              //SDK initialization，SDK cleaning up omitting

                              // device info
                              public static NetSDKLib.NET_DEVICEINFO_Ex m_stDeviceInfo =
                              new NetSDKLib.NET_DEVICEINFO_Ex();

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

                              //log in ti the device
                              public static boolean login(String m_strIp, int m_nPort, String
                              m_strUser, String m_strPassword) {
                              //input parameter
NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY pstInParam=
new NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY();
pstInParam. szIP= m_strIp;
pstInParam.nport= m_nPort;
pstInParam.szUserName= m_strUser;
pstInParam.szPassword= m_strPassword;
//Input parameter
NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY pstOutParam=
new NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY();
                              m_hLoginHandle =
                              netsdk.CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WIT
                              H_HIGHLEVEL_SECURITY pstInParam,
                              NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY  pstOutParam);

                              if(m_hLoginHandle.longValue() == 0) {
                                        System.err.printf("Login Device[%s] Port[%d]Failed. %s\n",
                              m_strIp, m_nPort, ToolKits.getErrorCodePrint());
                                   } else {
                                        System.out.println("Login Success ");
                                   }

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

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

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

```
                                    }

                                    return bRet;
                               }
}
```

2.3 Real-Time Monitoring

2.3.1 Introduction

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

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

- Pass in the window handle for SDK to directly decode and play the stream (Windows system only).
- Call back the real-time stream data to users for independent treatment.
- Save the real-time record to the specific file through saving the callback stream or calling the SDK interface.

2.3.2 Interface Overview

Table 2-4 Description of real-time monitoring interfaces

Interface	Description
CLIENT_RealPlayEx	Extension interface for starting the real-time monitoring
CLIENT_StopRealPlayEx	Extension interface for stopping the 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	Extension interface for setting the real-time monitoring data callback

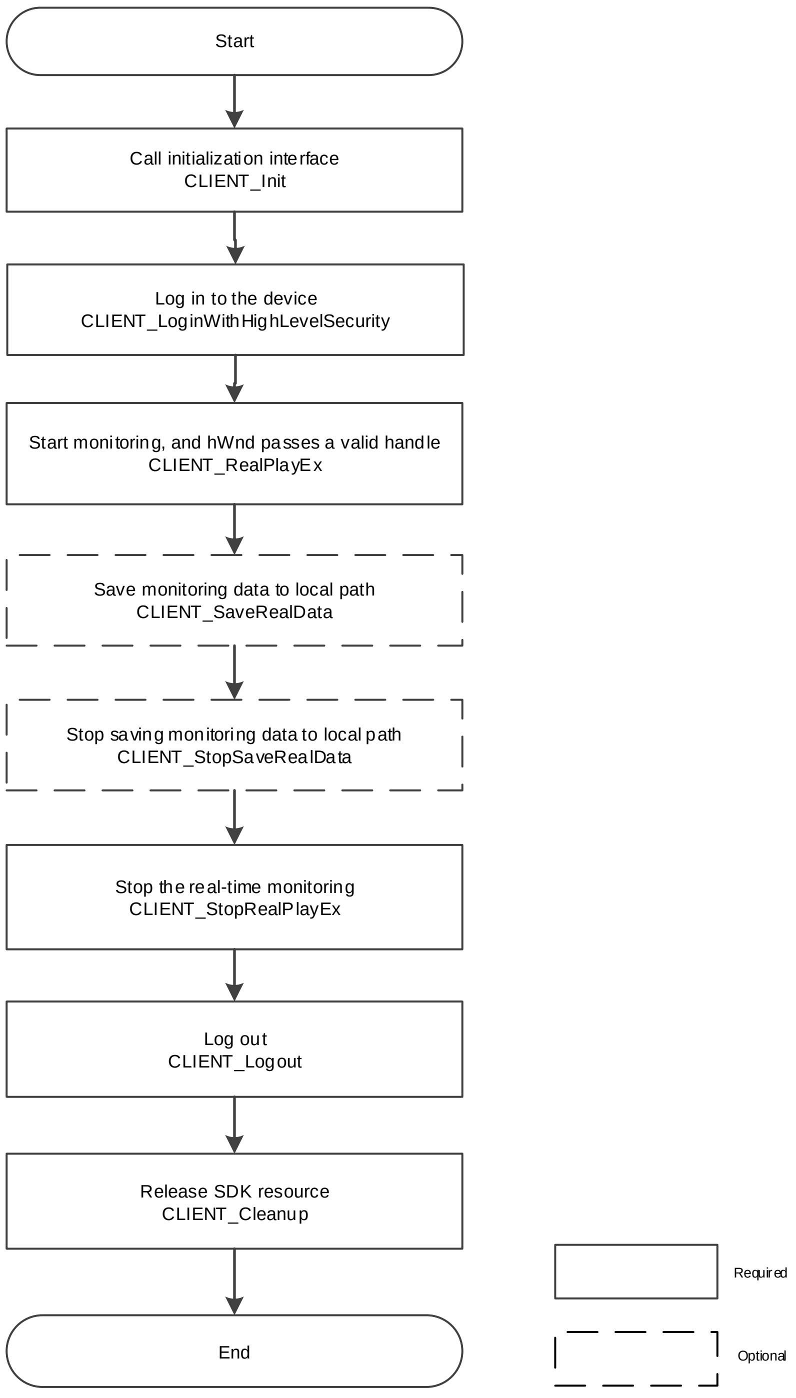
2.3.3 Process Description

You can realize the real-time monitoring through SDK integrated play 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. For the process of SDK decoding play, see Figure 2-3.

Figure 2-3 SDK decoding play flowchart


Start
Call initialization interface
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Start monitoring, and hWnd passes a valid handle
CLIENT_RealPlayEx
Save monitoring data to local path
CLIENT_SaveRealData
Stop saving monitoring data to local path
CLIENT_StopSaveRealData
Stop the real-time monitoring
CLIENT_StopRealPlayEx
Log out
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
End
Required
Optional

Process Description

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

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

Step 3 Call **CLIENT_RealPlayEx** to start 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 a local video file.

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

Step 7 After using the function module, call **CLIENT_Logout** to log out of the device.

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

Notes

- SDK decoding play only supports Windows system. You need to call the decoding after getting the stream for display 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 application for monitoring resources in the interface should make some agreements with the device before requesting the monitoring data. There are some timeout settings (see "NET_PARAM structure"), and the field related to monitoring is **nGetConnInfoTime**. If there is timeout due to the reasons such as poor 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 the CLIENT_Init function.

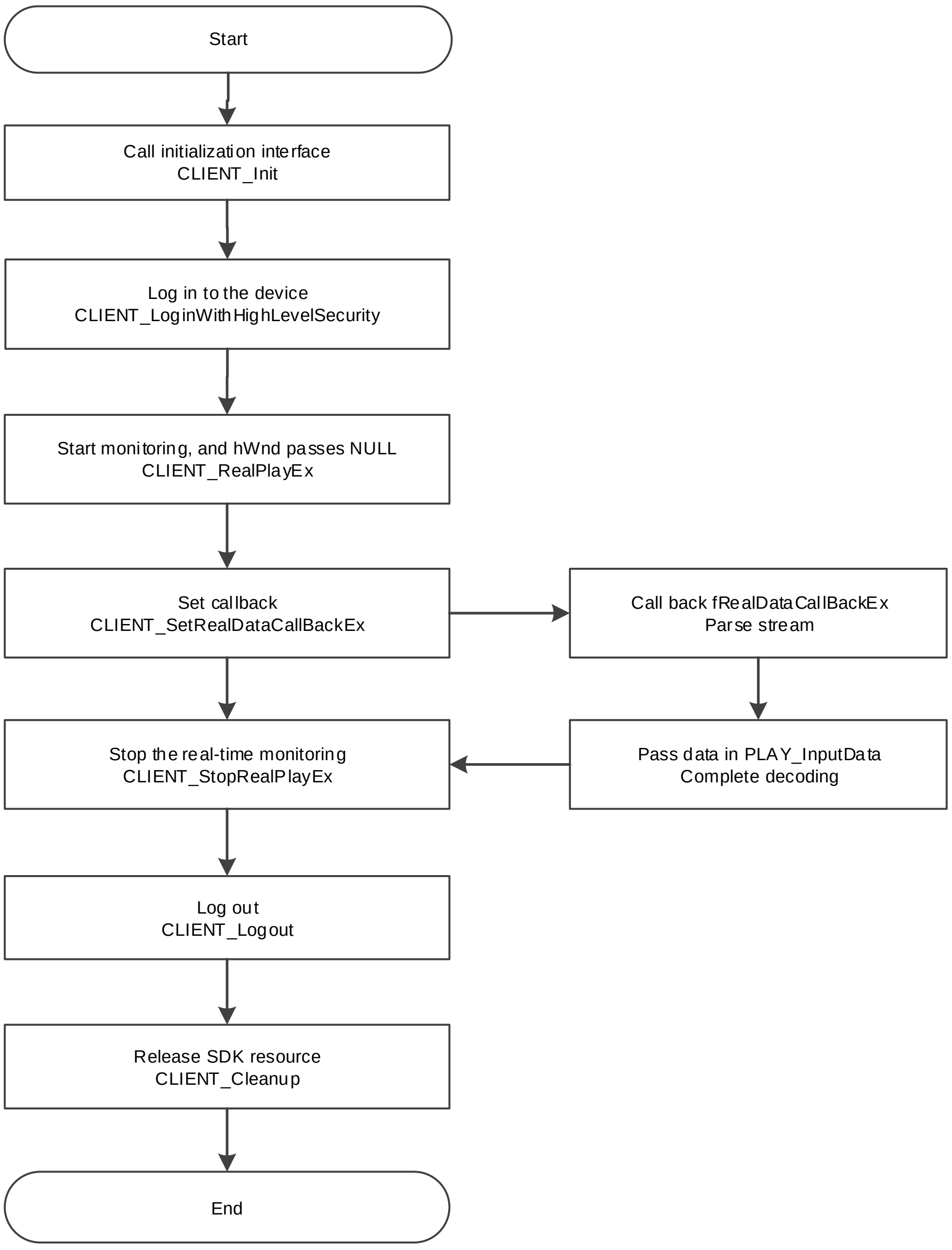
```
NET_PARAM stuNetParam = new NET_PARAM();
stuNetParam. nGetConnInfoTime = 5000; if the value is 0, it is 1000 ms by default
CLIENT_SetNetworkParam (stuNetParam);
```

- Failed to repeat opening: Because some devices do not support opening the monitoring function on the same channel for multiple times, these devices might fail from the second opening. In this case, you can try the following:
  - ◇ Close the opened channel first. For example, if you already opened the main stream video on the channel 1 and still want to open the sub stream video on the same channel, you can close the main stream video first and then open the sub stream video.
  - ◇ Log in 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.
- If the system resource is insufficient, the device might return error instead of recovering 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.

2.3.3.2 Calling Private Play Library

SDK calls back the real-time monitoring stream to you and then you call PlaySDK to perform decoding play. For the process of calling the private play library for decoding play, see Figure 2-4.

Figure 2-4 Third-party decoding play flowchart



Start
Call initialization interface
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Start monitoring, and hWnd passes NULL
CLIENT_RealPlayEx
Set callback
CLIENT_SetRealDataCallBackEx
Call back fRealDataCallBackEx
Parse stream
Stop the real-time monitoring
CLIENT_StopRealPlayEx
Pass data in PLAY_InputData
Complete decoding
Log out
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
End

Process Description

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

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

Step 3 After successful login, call **CLIENT_RealPlayEx** to start 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 using the real-time monitoring, call **CLIENT_StopRealPlayEx** to stop it.

Step 7 After using the function module, call **CLIENT_Logout** to log out of the device.

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

Notes

- 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 as 3 M.
  - ◇ SDK callback can only call back the next video data after returning from you. It is not recommended for you to consume time for unnecessary operations; otherwise the performance will be affected.

2.3.4 Sample Code

2.3.4.1 SDK Decoding Play

```
import java.awt.Panel;

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

import com.sun.jna.Native;

/**
 * functions realized when real-time live view
 * Mainly includes: Start pulling stream and stop pulling stream.
 */
public class RealPlayModule {
                              // start live view
                              public static LLong startRealPlay(int channel, int stream, Panel
                              realPlayWindow) {
                                   LLong m_hPlayHandle =
                              LoginModule.netsdk.CLIENT_RealPlayEx(LoginModule.m_hLogin
                              Handle, channel, Native.getComponentPointer(realPlayWindow),
                              stream);

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

//custom stream saving file, optional. Use it when need to save videos.
String outFile="example/outputfile";
LoginModule.netsdk.CLIENT_SaveRealData(m_hPlayHandle,outFile);
                                   }
```

```
				return m_hPlayHandle;
			}

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

2.3.4.2 Calling Play Library

```
public class RealPlayModule {
			class DataCallBackEx implements
			NetSDKLib.fRealDataCallBackEx{
				@Override
				public void invoke(LLong lRealHandle, int dwDataType, Pointer
			pBuffer,
						int dwBufSize, int param, Pointer dwUser) {
					// TODO


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


				LoginModule.netsdk.CLIENT_SetRealDataCallBackEx(m_hPla
			yHandle,m_DataCallBackEx, null, 0x00000001);

				if(m_hPlayHandle.longValue() == 0) {
```

```
                    System.err.println("start real-time monitoring failed, error
code" + ToolKits.getErrorCodePrint());
            } else {
                    System.out.println("Success to start realplay");
            }

            return m_hPlayHandle;
    }

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

2.4 Subscription to Intelligent Event

2.4.1 Introduction

Intelligent subscription: Based on the analysis of real-time streams, when detecting the preset event, the intelligence device will send the event to users. Intelligent events include traffic violations, whether there is any park space in the parking lot, and other events.

Intelligent subscription implementation: SDK automatically connects to the device and subscribes to the intelligent event function from the device. When the device detects an intelligent event, it will send the event to SDK immediately.

For supported intelligent subscription events, see the constants starting with EVENT_IVS_ in NetSDKLib.java, including general events such as occupied lane and vehicle violations.

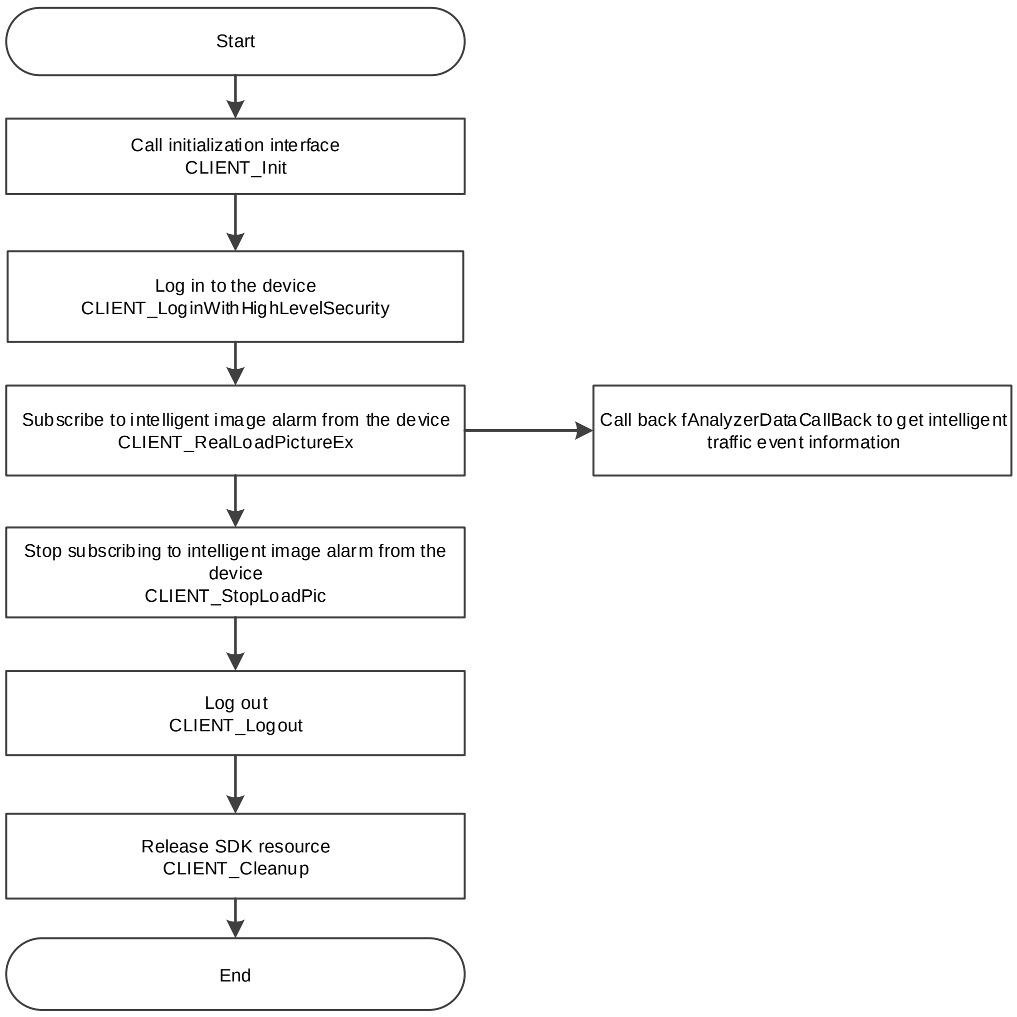
2.4.2 Interface Overview

Table 2-5 Description of interfaces for reporting intelligent traffic events

Interface	Description
CLIENT_RealLoadPictureEx	Subscribe to intelligent event.
CLIENT_StopLoadPic	Unsubscribe from intelligent event.
fAnalyzerDataCallBack	For callback to get intelligent event information

2.4.3 Process Description

Figure 2-5 Process of reporting intelligent subscription events



Start
Call initialization interface
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Subscribe to intelligent image alarm from the device
CLIENT_RealLoadPictureEx
Call back fAnalyzerDataCallBack to get intelligent traffic event information
Stop subscribing to intelligent image alarm from the device
CLIENT_StopLoadPic
Log out
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
End

Process Description

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

Step 2 After successful initialization, call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.

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

Step 4 After successful subscription, use the callback set by fAnalyzerDataCallBack to inform the user of intelligent event reported by the device.

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

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

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

Notes

- Subscription event type: If you need to report different intelligent events at the same time, you can subscribe to all intelligent events (EVENT_IVS_ALL) or a single intelligent event.

- Set whether to receive images: Because some devices are in 3G or 4G network environment, when SDK connects to the device, if you do not need to receive images, you can set the parameter bNeedPicFile in CLIENT_ RealLoadPictureEx interface to False, for only receiving intelligent traffic event information without images.
- Sending -1 through channel will subscribe all channels. Some intelligent trafficproducts do not support subscribing all channels. If the subscription failed by sending -1, please subscribe one channel.

2.4.4 Sample Code

```
//take access control event as an example
//omit SDK initialization and access control device login
// handle subscription
public static LLong m_hAttachHandle = new LLong(0);
private AnalyzerDataCB analyzerCallback = new AnalyzerDataCB();
private boolean isAttach = false;
// Listening
private void setOnClickListener() {
                        // subscribe intelligent event for asscess control devices
                        attachBtn.addActionListener(new ActionListener() {
                            @Override
                            public void actionPerformed(ActionEvent arg0) {
                                m_hAttachHandle =
                        GateModule.realLoadPic(chnComboBox.getSelectedIndex(),
                        analyzerCallback);
                                if(m_hAttachHandle.longValue() != 0) {
                                    isAttach = true;
                                    attachBtn.setEnabled(false);
                                    detachBtn.setEnabled(true);
                                } else {
                                    JOptionPane.showMessageDialog(null,
                        ToolKits.getErrorCodeShow(), Res.string().getErrorMessage(),
                        JOptionPane.ERROR_MESSAGE);
                                }
                            }
                        });

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

```
                                    clearPanel();
                                          }
                                    });
}
//access control system intelligent callback, inherit fAnalyzerDataCallBack and use its own logic
private class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack {
                                    private BufferedImage gateBufferedImage = null;
                                    @Override
      public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
                                                      Pointer pAlarmInfo, Pointer pBuffer, int
                                    dwBufSize,
                                                      Pointer dwUser, int nSequence, Pointer
                                    reserved)
      {
            if (lAnalyzerHandle.longValue() == 0 || pAlarmInfo == null) {
                  return -1;
            }

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

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

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

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

```
                    //image and access control info display
                    EventQueue eventQueue = Toolkit.getDefaultToolkit().getSystemEventQueue();
                    if (eventQueue != null) {
                    eventQueue.postEvent( new AccessEvent(target, gateBufferedImage, msg));


                                                                                                                              }
                                            }

                                            return 0;
      }
}
```

3 Face Detection

3.1 Subscription to Event

3.1.1 Introduction

When the camera detects the appearance of faces in the specified region, an intelligent event message is generated and reported to NetSDK.

3.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see “2.4 Subscribing Intelligent Event”.

3.1.3 Enumeration and Structure

- Enumerated value corresponding to the event: EVENT_IVS_FACEDETECT
- Structure corresponding to the event: DEV_EVENT_FACEDETECT_INFO

3.2 Sample Code

```
                        * static to avoid recycle
                      */
                     private static class AnalyzerDataCB implements
                     NetSDKLib.fAnalyzerDataCallBack {
                         private AnalyzerDataCB() {}

                         private static class AnalyzerDataCBHolder {
                             private static final AnalyzerDataCB instance = new
                     AnalyzerDataCB();
                         }

                         public static AnalyzerDataCB getInstance() {
                             return AnalyzerDataCBHolder.instance;
                         }

        public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
                                           Pointer pAlarmInfo, Pointer pBuffer, int
                     dwBufSize,
                                           Pointer dwUser, int nSequence, Pointer
                     reserved)
        {
            if (lAnalyzerHandle.longValue() == 0 || pAlarmInfo == null) {
                return -1;
            }

                        switch(dwAlarmType)
            {
                            case NetSDKLib.EVENT_IVS_FACEDETECT:    ///<
                    face detection
                            {
                                DEV_EVENT_FACEDETECT_INFO msg = new
                    DEV_EVENT_FACEDETECT_INFO();
                                ToolKits.GetPointerData(pAlarmInfo, msg);

                                // save image , get image buffer
                                try {
                                    saveFaceDetectPic(pBuffer, dwBufSize,
                    msg);
                                } catch (FileNotFoundException e) {
                                    e.printStackTrace();
                                }

                                // list and image display
                    EventQueue.invokeLater(new
                    FaceDetectRunnable(globalBufferedImage, personBufferedImage,
                    msg));

                                // release memory
                            msg = null;
                            System.gc();

                                break;
                            }
}
//stop subscription
if(m_hAttachHandle.longValue() != 0) {

                        LoginModule.netsdk.CLIENT_StopLoadPic(m_hAttachHandle)
                    ;
                        m_hAttachHandle.setValue(0);
                    }
```

4 Face Recognition

4.1 Subscription to Event

4.1.1 Introduction

Face recognition: When the server compares the faces detected in the video with face images in its internal database and the faces match, the server will report the event to the platform.

Information in the face recognition event includes: Recognized person information, image files of each person, and similarity with the current face.

4.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see "2.4 Subscribing Intelligent Event".

4.1.3 Enumeration and Structure

- Enumerated value corresponding to the event: EVENT_IVS_FACERECOGNITION
- Structure corresponding to the event: DEV_EVENT_FACERECOGNITION_INFO

4.2 Sample Code

```
                         * static to avoid recycle

                          */
                         private static class AnalyzerDataCB implements
                         NetSDKLib.fAnalyzerDataCallBack {
                             private AnalyzerDataCB() {}

                             private static class AnalyzerDataCBHolder {
                                 private static final AnalyzerDataCB instance = new
                         AnalyzerDataCB();
                             }

                             public static AnalyzerDataCB getInstance() {
                                 return AnalyzerDataCBHolder.instance;
                             }

        public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
                                                     Pointer pAlarmInfo, Pointer pBuffer, int
                         dwBufSize,
                                                     Pointer dwUser, int nSequence, Pointer
                         reserved)
        {
            if (lAnalyzerHandle.longValue() == 0 || pAlarmInfo == null) {
                return -1;
            }

                                    switch(dwAlarmType)
            {
                                        case NetSDKLib.EVENT_IVS_FACERECOGNITION:
                        ///< face recognition event
                                        {
                        // DEV_EVENT_FACERECOGNITION_INFO   structural body
                        toolarge，new object will consume too much time，
                        ToolKits.GetPointerData content copy does not consume time
                        // if too many devices, change static
                        DEV_EVENT_FACERECOGNITION_INFO msg = new
                        DEV_EVENT_FACERECOGNITION_INFO(); change to global
                        // change to global because new each time takes too much time,
                        ifchanged to global, you need to lock process iof the case

                                            // why lock, because shared an object to avoid
                        data error

                                            // takes about 800ms
                        DEV_EVENT_FACERECOGNITION_INFO msg = new
                        DEV_EVENT_FACERECOGNITION_INFO();

                                            // takes about 20ms
                            ToolKits.GetPointerData(pAlarmInfo, msg);

                    // save image, get image buffer
                            // takes about 20ms
                                        try {
                                                saveFaceRecognitionPic(pBuffer, dwBufSize,
                        msg);
                                            } catch (FileNotFoundException e) {
                                                e.printStackTrace();
                                            }

                                        // list and image display
                                        // callback is sub thread. The following is UI thread,
                        used to refreash UI
                        EventQueue.invokeLater(new
                        FaceRecognitionRunnable(globalBufferedImage,
                        personBufferedImage, candidateBufferedImage, msg, index));
```

```
                                        // release memory
                                        msg = null;
                                        System.gc();

                                            break;
                                        }
}
//stop subscription
if(m_hAttachHandle.longValue() != 0) {

                               LoginModule.netsdk.CLIENT_StopLoadPic(m_hAttachHandle)
                           ;
                                   m_hAttachHandle.setValue(0);
                               }
```

5 General Behavior

5.1 Subscription to Event

5.1.1 Introduction

General behaviors mainly include intrusion and tripwire. Intrusion indicates that an alarm is triggered when a person who intrudes into the specified region is detected. Tripwire indicates that an alarm is triggered when a person who crosses the line set by the camera is detected.

5.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see “2.4 Subscribing Intelligent Event”.

5.1.3 Enumeration and Structure

- Tripwire event
  - ◇ Enumerated value corresponding to the tripwire event: EVENT_IVS_CROSSLINEDETECTION
  - ◇ Structure corresponding to the tripwire event: DEV_EVENT_CROSSLINE_INFO
- Intrusion event
  - ◇ Enumerated value corresponding to the intrusion event: EVENT_IVS_CROSSREGIONDETECTION
  - ◇ Structure corresponding to the intrusion event: DEV_EVENT_CROSSREGION_INFO

5.2 Sample Code

```
/**
 * IVS callback
 */
public class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack{

                        private File picturePath;

    private AnalyzerDataCB() {
                        picturePath = new File("./AnalyzerPicture/");
                            if (!picturePath.exists()) {
                                picturePath.mkdir();
                            }
   }

    private static class CallBackHolder {
        private static AnalyzerDataCB instance = new AnalyzerDataCB();
	}

	public static AnalyzerDataCB getInstance() {
		return CallBackHolder.instance;
	}

	// callback
	public int invoke(NetSDKLib.LLong lAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo,
	Pointer pBuffer, int dwBufSize, Pointer dwUser, int nSequence, Pointer reserved)
	{
		if (lAnalyzerHandle == null || lAnalyzerHandle.longValue() == 0) {
			return -1;
		}

		NetSDKLib.NET_EVENT_FILE_INFO stuFileInfo = null;
		NetSDKLib.NET_PIC_INFO stPicInfo = null;

		switch(dwAlarmType)
		{
						case NetSDKLib.EVENT_IVS_CROSSLINEDETECTION :
					// warning line event
						{
					NetSDKLib.DEV_EVENT_CROSSLINE_INFO msg = new
					NetSDKLib.DEV_EVENT_CROSSLINE_INFO();
							ToolKits.GetPointerData(pAlarmInfo, msg);
							stuFileInfo = msg.stuFileInfo;
							stPicInfo = msg.stuObject.stPicInfo;
					System.out.printf("【warning line event】 time (UTC):%s channel
					No.:%d start time:%s end time:%s event occurrence times:%d
					event source device ID:%s \n",
					msg.UTC, msg.nChannelID, msg.stuObject.stuStartTime,
					msg.stuObject.stuEndTime,
					msg.nOccurrenceCount, new String(msg.szSourceDevice));
							break;
						}
case NetSDKLib.EVENT_IVS_CROSSREGIONDETECTION: ///<   warning area event
			{
					NetSDKLib.DEV_EVENT_CROSSREGION_INFO msg = new
					NetSDKLib.DEV_EVENT_CROSSREGION_INFO();
					ToolKits.GetPointerData(pAlarmInfo, msg);
					String Picture = picturePath + "\\" + System.currentTimeMillis() +
					".jpg";
					ToolKits.savePicture(pBuffer, 0, dwBufSize, Picture);
					System.out.println("warning area event time(UTC)：" + msg.UTC + "
					channel No.:" + msg.nChannelID + "start time:" +
					msg.stuObject.stuStartTime + "End time:" +
					msg.stuObject.stuEndTime);
//                              PrintStruct.print(msg);
                                break;
                }
                }
                                     }
```

6 Human Detection

6.1 Subscription to Event

6.1.1 Introduction

When the camera detects human features in the specified region, an intelligent event message is generated and reported to NetSDK.

6.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see "2.4 Subscribing Intelligent Event".

6.1.3 Enumeration and Structure

- Enumerated value corresponding to the event: EVENT_IVS_HUMANTRAIT
- Structure corresponding to the event: DEV_EVENT_HUMANTRAIT_INFO

6.2 Sample Code

```
/**
 * human detetion callback
 */
public class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack{

                        private File picturePath;

    private AnalyzerDataCB() {
                        picturePath = new File("./AnalyzerPicture/");
                            if (!picturePath.exists()) {
                                picturePath.mkdir();
                            }
   }

    private static class CallBackHolder {
        private static AnalyzerDataCB instance = new AnalyzerDataCB();
    }

    public static AnalyzerDataCB getInstance() {
        return CallBackHolder.instance;
    }

    // callback
```

```
public int invoke(NetSDKLib.LLong lAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo,
Pointer pBuffer, int dwBufSize, Pointer dwUser, int nSequence, Pointer reserved)
{
        if (lAnalyzerHandle == null || lAnalyzerHandle.longValue() == 0) {
                return -1;
        }

        NetSDKLib.NET_EVENT_FILE_INFO stuFileInfo = null;
        NetSDKLib.NET_PIC_INFO stPicInfo = null;

        switch(dwAlarmType)
        {
                                                        case NetSDKLib.EVENT_IVS_HUMANTRAIT:      //
                                body feature event
                                                        {
                                DEV_EVENT_HUMANTRAIT_INFO msg = new
                                DEV_EVENT_HUMANTRAIT_INFO();
                                                                ToolKits.GetPointerData(pAlarmInfo, msg);
                                                                PrintStruct.print(msg);

                                                                //save panoramaic image
                                                                if(msg.stuSceneImage.nLength>0)
                                                                {
                                String strFileName = path + "\\" + System.currentTimeMillis() +
                                "HumanTrait_ panoramaic image.jpg";
                                ToolKits.savePicture(pBuffer, msg.stuSceneImage.nOffSet,
                                msg.stuSceneImage.nLength, strFileName);
                                                                }
                                                                else
                                                                {
                                                                        System.out.println("no panoramaic image");
                                                                }

                                                                //save face image
                                                                if(msg.stuFaceImage.nLength>0)
                                                                {
                                String strFileName = path + "\\" + System.currentTimeMillis() +
                                "HumanTrait_face image.jpg";
                                ToolKits.savePicture(pBuffer, msg.stuFaceImage.nOffSet,
                                msg.stuFaceImage.nLength, strFileName);
                                                                }
                                                                else
                                                                {
                                                                        System.out.println("no face image");
                                                                }

                                                                //save face panoramaic image
                                if(msg.stuFaceSceneImage.nLength>0)
                                {
                    String strFileName = path + "\\" + System.currentTimeMillis() +
                    "HumanTrait_face panoramaic image.jpg";
                    ToolKits.savePicture(pBuffer, msg.stuFaceSceneImage.nOffSet,
                    msg.stuFaceSceneImage.nLength, strFileName);
                                }
                                else
                                {
                                    System.out.println("no panoramaic face
                    image");
                                }

                                //save human image
                                if(msg.stuHumanImage.nLength>0)
                                {
                    String strFileName = path + "\\" + System.currentTimeMillis() +
                    "HumanTrait_human image.jpg";
                    ToolKits.savePicture(pBuffer, msg.stuHumanImage.nOffSet,
                    msg.stuHumanImage.nLength, strFileName);
                                }
                                else
                                {
                                    System.out.println("no human image);
                                }

                                break;
                            }
```

7 Thermal Temperature Measurement Event

7.1 Subscription to Event

7.1.1 Introduction

When the thermal camera detects human in the specified region, it will report body temperature through the thermal technology.

7.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see “2.4 Subscribing Intelligent Event”.

7.1.3 Enumeration and Structure

- Enumerated value corresponding to the event: EVENT_IVS_ANATOMY_TEMP_DETECT
- Structure corresponding to the event: DEV_EVENT_ANATOMY_TEMP_DETECT_INFO

7.2 Sample Code

```
/**
 * Human detection callback
 */
public class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack{

                        private File picturePath;

    private AnalyzerDataCB() {
                        picturePath = new File("./AnalyzerPicture/");
                            if (!picturePath.exists()) {
                                picturePath.mkdir();
                            }
   }

    private static class CallBackHolder {
        private static AnalyzerDataCB instance = new AnalyzerDataCB();
    }

    public static AnalyzerDataCB getInstance() {
        return CallBackHolder.instance;
    }

    // Callback
    public int invoke(NetSDKLib.LLong lAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo,
   Pointer pBuffer, int dwBufSize, Pointer dwUser, int nSequence, Pointer reserved)
    {
        if (lAnalyzerHandle == null || lAnalyzerHandle.longValue() == 0) {
            return -1;
        }

        NetSDKLib.NET_EVENT_FILE_INFO stuFileInfo = null;
        NetSDKLib.NET_PIC_INFO stPicInfo = null;

        switch(dwAlarmType)
        {
case NetSDKLib.EVENT_IVS_ANATOMY_TEMP_DETECT :
 // Intelligent human temperature measurement event
                                    {
                        NetSDKLib.DEV_EVENT_ANATOMY_TEMP_DETECT_INFO msg
                        = new
                        NetSDKLib.DEV_EVENT_ANATOMY_TEMP_DETECT_INFO();
                                ToolKits.GetPointerData(pAlarmInfo, msg);

                        System.out.printf("[Intelligent huamn temperature measurement
                        event] time (UTC): %s Channel number: %d nAction: %d
                        szName: %s nPresetID: %d \n",
                        msg.UTC, msg.nChannelID, msg.nAction, new
                        String(msg.szName).trim(), msg.nPresetID);

                        System.out.printf("[Human temperature information in the region]
                        nObjectID"+msg.stManTempInfo.nObjectID+"dbHighTemp"+msg.st
                        ManTempInfo.dbHighTemp+"
                        nTempUnit"+msg.stManTempInfo.nTempUnit+"bIsOverTemp"+msg.
                        stManTempInfo.bIsOverTemp+"bIsUnderTemp"+msg.stManTempInf
                        o.bIsUnderTemp+"\n");
                                //Visible image panorama
                                if(msg.stVisSceneImage!=null &&
                        msg.stVisSceneImage.nLength> 0){
                        String bigPicture = picturePath + "\\" + System.currentTimeMillis() +
                        ".jpg";
                        ToolKits.savePicture(pBuffer, msg.stVisSceneImage.nOffset,
                        msg.stVisSceneImage.nLength, bigPicture);
                                }
                                //Thermal imaging panorama
                            if(msg.stThermalSceneImage!=null &&
                        msg.stThermalSceneImage.nLength> 0){
            String smallPicture = picturePath + "\\" + System.currentTimeMillis() + "small.jpg";
                        ToolKits.savePicture(pBuffer, msg.stThermalSceneImage.nOffset,
                        msg.stThermalSceneImage.nLength, smallPicture);
                            }
```

```
                                        break;
                                    }
}
```

8 Access Control Event

8.1 Event Subscription

8.1.1 Introduction

When the access control device is unlocked, the unlocking related event information is reported, including event, unlocking mode, unlocking personnel, and other corresponding information.

8.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see "2.4 Subscribing Intelligent Event".

8.1.3 Enumeration and Structure

- Enumerated value corresponding to the event: EVENT_IVS_ACCESS_CTL
- Structure corresponding to the event: DEV_EVENT_ACCESS_CTL_INFO

8.2 Sample Code

```
private class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack {
                                        private BufferedImage gateBufferedImage = null;

          public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
                                                                Pointer pAlarmInfo, Pointer pBuffer, int
                                  dwBufSize,
                                                                Pointer dwUser, int nSequence, Pointer
                                  reserved)
           {
                 if (lAnalyzerHandle.longValue() == 0 || pAlarmInfo == null) {
                       return -1;
                 }

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

                 ///< access control event
                                          if(dwAlarmType ==
                                  NetSDKLib.EVENT_IVS_ACCESS_CTL) {
                                                DEV_EVENT_ACCESS_CTL_INFO msg = new
                                  DEV_EVENT_ACCESS_CTL_INFO();
```

```
                ToolKits.GetPointerData(pAlarmInfo, msg);

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

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

                                    // image and access control info displayed on the
                        interface
                        EventQueue eventQueue =
                        Toolkit.getDefaultToolkit().getSystemEventQueue();
                                    if (eventQueue != null) {
                                    eventQueue.postEvent( new
                        AccessEvent(target,gateBufferedImage,msg));
                                    }
                                }

                                return 0;
        }
```

9 People Counting

9.1 Introduction

A camera is installed in the business region, and the intelligent analysis server accurately counts the number of people entering and exiting each entrance in real time according to the video data collected by the camera. Such products are widely used in large-scale business, tourism, public safety, cultural industry expo, chain stores and other industries.

Through real-time subscription to people counting data, you can get reports related to total number of people entered and exited in real time.

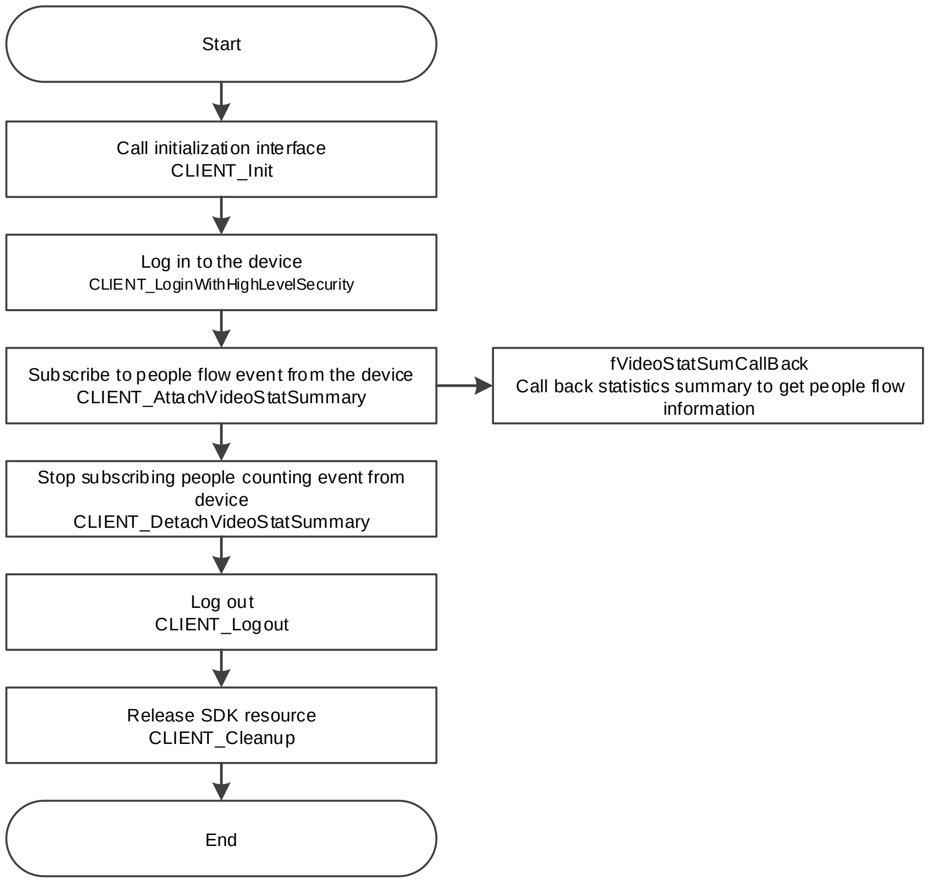
9.2 Interface Overview

Table 9-1 Description of people counting interface

Interface	Description
CLIENT_AttachVideoStatSummary	Subscribe to the people counting event.
CLIENT_DetachVideoStatSummary	Unsubscribe from the people counting event.

9.3 Process Description

Figure 9-1 Process of subscribing to people counting


Start
Call initialization interface
CLIENT_Init
Log in to the device
CLIENT_LoginWithHighLevelSecurity
Subscribe to people flow event from the device
CLIENT_AttachVideoStatSummary
fVideoStatSumCallBack
Call back statistics summary to get people flow information
Stop subscribing people counting event from device
CLIENT_DetachVideoStatSummary
Log out
CLIENT_Logout
Release SDK resource
CLIENT_Cleanup
End

Process Description

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

Step 2 After initialization, call **CLIENT_LoginWithHighLevelSecurity** to log in to the device.

Step 3 Call **CLIENT_AttachVideoStatSummary** to subscribe to the people counting event from the device.

Step 4 After successful subscription, use the callback set by **fVideoStatSumCallBack** to inform the user of people counting events reported by the device.

Step 5 After using the reporting function of the people counting event, call **CLIENT_DetachVideoStatSummary** to stop subscribing to the people counting event.

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

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

9.4 Sample Code

```
/**
 * Subscribe
 */
public void attachVideoStatSummary() {
        if (loginHandle.longValue() == 0) {
            return;
        }

        NET_IN_ATTACH_VIDEOSTAT_SUM videoStatIn = new
    NET_IN_ATTACH_VIDEOSTAT_SUM();
        videoStatIn.nChannel = 1;
        videoStatIn.cbVideoStatSum =
    VideoStatSumCallback.getInstance();

        NET_OUT_ATTACH_VIDEOSTAT_SUM videoStatOut = new
    NET_OUT_ATTACH_VIDEOSTAT_SUM();

        videoStatHandle =
    netsdkApi.CLIENT_AttachVideoStatSummary(loginHandle,
    videoStatIn, videoStatOut, 5000);
        if( videoStatHandle.longValue() == 0 ) {
            System.err.printf("Attach Failed!LastError = %x\n",
    netsdkApi.CLIENT_GetLastError());
            return;
        }
        System.out.printf("Attach Success!Wait Device Notify
    Information\n");
    }

    /**
     * Stop the subscription
     */
    public void detachVideoStatSummary() {
        if (videoStatHandle.longValue() != 0) {

        netsdkApi.CLIENT_DetachVideoStatSummary(videoStatHandl
    e);
            videoStatHandle.setValue(0);
        }
    }
  }
/*
  * People counting callback
*/
    private static class VideoStatSumCallback implements
    NetSDKLib.fVideoStatSumCallBack {
        private static VideoStatSumCallback instance = new
    VideoStatSumCallback();
        private VideoStatSumCallback() {}
        public static VideoStatSumCallback getInstance() {
            return instance;
        }

        public void invoke(LLong lAttachHandle,
    NET_VIDEOSTAT_SUMMARY stVideoState, int dwBufLen, Pointer
    dwUser){
            System.out.printf("Channel[%d] GetTime[%s]
    RuleName[%s]\n" +
                                    "People In    Information[Total[%d] Hour[%d]
    Today[%d]]\n" +
                                    "People Out Information[Total[%d] Hour[%d]
    Today[%d]]\n",
                                    stVideoState.nChannelID ,
    stVideoState.stuTime.toStringTime() ,
                                    new String(stVideoState.szRuleName).trim(),
                                    stVideoState.stuEnteredSubtotal.nToday ,
                                    stVideoState.stuEnteredSubtotal.nHour ,
                                    stVideoState.stuEnteredSubtotal.nTotal ,
                                    stVideoState.stuExitedSubtotal.nToday ,
                                    stVideoState.stuExitedSubtotal.nHour ,
                                    stVideoState.stuExitedSubtotal.nTotal
                                    );
        }
    }
```

10 Intelligent Traffic Event

10.1 Subscription to Event

10.1.1 Introduction

Intelligent traffic event sending: Based on the analysis of real-time streams, when detecting the preset traffic event, the intelligent traffic device will send the event to users. Intelligent traffic events include traffic violations, parking space, and other events.

Intelligent traffic event sending: SDK automatically connects to the device and subscribes to the intelligent event function from the device. When the device detects an intelligent event, it will send the event to SDK immediately.

10.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see "2.4 Subscribing Intelligent Event".

10.1.3 Enumeration and Structure

- Intersection event
  Enumerated value corresponding to the intersection event: EVENT_IVS_TRAFFICJUNCTION
  Structure corresponding to the intersection event: DEV_EVENT_TRAFFICJUNCTION_INFO
- The event of traffic violation—driving on lane
  Enumerated value corresponding to the event of traffic violation—driving on lane: EVENT_IVS_TRAFFIC_OVERLINE
  Structure corresponding to the event of traffic violation—driving on lane: DEV_EVENT_TRAFFIC_OVERLINE_INFO
- The event of traffic violation—wrong-way driving
  Enumerated value corresponding to the event of traffic violation—wrong-way driving: EVENT_IVS_TRAFFIC_RETROGRADE
  Structure corresponding to the event of traffic violation—wrong-way driving: DEV_EVENT_TRAFFIC_RETROGRADE_INFO
- The event of traffic—running a red light
  Enumerated value corresponding to the event of traffic—running a red light: EVENT_IVS_TRAFFIC_RUNREDLIGHT
  Structure corresponding to the event of traffic—running a red light: DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO
- The event of traffic violation—illegal left turn
  Enumerated value corresponding to the event of traffic violation—illegal left turn: EVENT_IVS_TRAFFIC_TURNLEFT

Structure corresponding to the event of traffic violation—illegal left turn: DEV_EVENT_TRAFFIC_TURNLEFT_INFO

- The event of traffic violation—illegal right turn

  Enumerated value corresponding to the event of traffic violation—illegal right turn: EVENT_IVS_TRAFFIC_TURNRIGHT

  Structure corresponding to the event of traffic violation—illegal right turn: DEV_EVENT_TRAFFIC_TURNRIGHT_INFO
- The event of traffic violation—illegal U turn

  Enumerated value corresponding to the event of traffic violation—illegal U turn: EVENT_IVS_TRAFFIC_UTURN

  Structure corresponding to the event of traffic violation—illegal U turn: DEV_EVENT_TRAFFIC_UTURN_INFO
- The event of traffic violation—underspeed

  Enumerated value corresponding to the event of traffic violation—underspeed: EVENT_IVS_TRAFFIC_UNDERSPEED

  Structure corresponding to the event of traffic violation—underspeed: DEV_EVENT_TRAFFIC_UNDERSPEED_INFO
- The event of traffic violation—illegal parking

  Enumerated value corresponding to the event of traffic violation—illegal parking: EVENT_IVS_TRAFFIC_PARKING

  Structure corresponding to the event of traffic violation—illegal parking: DEV_EVENT_TRAFFIC_PARKING_INFO
- The event of traffic violation—wrong lane

  Enumerated value corresponding to the event of traffic violation—wrong lane: EVENT_IVS_TRAFFIC_WRONGROUTE

  Structure corresponding to the event of traffic violation—wrong lane: DEV_EVENT_TRAFFIC_WRONGROUTE_INFO
- The event of traffic violation—illegal lane change

  Enumerated value corresponding to the event of traffic violation—illegal lane change: EVENT_IVS_TRAFFIC_CROSSLANE

  Structure corresponding to the event of traffic violation—illegal lane change: DEV_EVENT_TRAFFIC_CROSSLANE_INFO
- The event of traffic violation—crossing solid yellow line

  Enumerated value corresponding to the event of traffic violation—crossing solid yellow line: EVENT_IVS_TRAFFIC_OVERYELLOWLINE

  Structure corresponding to the event of traffic violation—crossing solid yellow line: DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO
- The event of traffic violation—vehicle with yellow plate in lane

  Enumerated value corresponding to the event of traffic violation—vehicle with yellow plate in lane: EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE

  Structure corresponding to the event of traffic violation—vehicle with yellow plate in lane: DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO
- The event of traffic violation—pedestrian priority on zebra crossing

  Enumerated value corresponding to the event of traffic violation—pedestrian priority on zebra crossing: EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY

  Structure corresponding to the event of traffic violation—pedestrian priority on zebra crossing: DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO

- The traffic event of manual capture
  Enumerated value corresponding to the traffic event of manual capture: EVENT_IVS_TRAFFIC_MANUALSNAP
  Structure corresponding to the traffic event of manual capture: DEV_EVENT_TRAFFIC_MANUALSNAP_INFO
- The event of vehicle in lane
  Enumerated value corresponding to the event of vehicle in lane: EVENT_IVS_TRAFFIC_VEHICLEINROUTE
  Structure corresponding to the event of vehicle in lane: DEV_EVENT_TRAFFIC_VEHICLEINROUTE_INFO
- The event of traffic violation—vehicle in bus lane
  Enumerated value corresponding to the event of traffic violation—vehicle in bus lane: EVENT_IVS_TRAFFIC_VEHICLEINBUSROUTE
  Structure corresponding to the event of traffic violation—vehicle in bus lane: DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INFO
- The event of traffic violation—illegal backing
  Enumerated value corresponding to the event of traffic violation—illegal backing: EVENT_IVS_TRAFFIC_BACKING
  Structure corresponding to the event of traffic violation—illegal backing: DEV_EVENT_IVS_TRAFFIC_BACKING_INFO
- The event of parking space occupied
  Enumerated value corresponding to the event of parking space occupied: EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING
  Structure corresponding to the event of parking space occupied: DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO
- The event of parking space not occupied
  Enumerated value corresponding to the event of parking space not occupied: EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING
  Structure corresponding to the event of parking space not occupied: DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO
- The event of traffic violation—not fastening seat belt
  Enumerated value corresponding to the event of traffic violation—not fastening seat belt: EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT
  Structure corresponding to the event of traffic violation—not fastening seat belt: DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT

10.2 Sample Code

```
/*
     * intelligent alarm event callback
     */
                         private class AnalyzerDataCB implements
                         NetSDKLib.fAnalyzerDataCallBack {
        public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
                                                   Pointer pAlarmInfo, Pointer pBuffer, int
                         dwBufSize,
                                                    Pointer dwUser, int nSequence, Pointer
                    reserved)
      {
          if (lAnalyzerHandle.longValue() == 0) {
              return -1;
          }

                       if(dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFICJUNCTION
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_RUNREDLIGHT
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_OVERLINE
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_RETROGRADE
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_TURNLEFT
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_TURNRIGHT
                      || dwAlarmType == NetSDKLib.EVENT_IVS_TRAFFIC_UTURN
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_OVERSPEED
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_UNDERSPEED
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_PARKING
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_WRONGROUTE
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_CROSSLANE
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_OVERYELLOWLINE
                        || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE
                        || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_MANUALSNAP
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_VEHICLEINROUTE
                        || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_VEHICLEINBUSROUTE
                      || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_BACKING
                        || dwAlarmType ==
                    NetSDKLib.EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING
```

```
                                          || dwAlarmType ==
                                    NetSDKLib.EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING
                                          || dwAlarmType ==
                                    NetSDKLib.EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT) {

                                                // get recognition object, vehicle object event occurrence time
                                    and lane No., and more
                                                GetStuObject(dwAlarmType, pAlarmInfo);

                                              // save imagesm get image buffer
                                                savePlatePic(pBuffer, dwBufSize, trafficInfo);

                                              // display list, image, and interfaces

                                                EventQueue eventQueue
                                    =Toolkit.getDefaultToolkit().getSystemEventQueue();
                                                if (eventQueue != null)
                                                    {
                                                eventQueue.postEvent(new
                                    TrafficEvent(target,snapImage,plateImage,trafficInfo));
                                                          }
                                                    }

                                                    return 0;
            }

            // get recognition object, vehicle object event occurrence time and lane No., and more
            private void GetStuObject(int dwAlarmType, Pointer pAlarmInfo)    {
                                    if(pAlarmInfo == null) {
                                          return;
                                    }

                                    switch(dwAlarmType) {
                                                      case NetSDKLib.EVENT_IVS_TRAFFICJUNCTION:
                                    ///< traffic checkpoint event
                                                      {
                                    NetSDKLib.DEV_EVENT_TRAFFICJUNCTION_INFO msg = new
                                    NetSDKLib.DEV_EVENT_TRAFFICJUNCTION_INFO();
                                                            ToolKits.GetPointerData(pAlarmInfo, msg);

                                    trafficInfo.m_EventName =
                                    Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFICJUN
                                    CTION);
                              try {
                                    trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                                    "GBK").trim();
                                                            } catch (UnsupportedEncodingException e) {
```

```
                                            e.printStackTrace();
                                        }
                        trafficInfo.m_PlateType = new
                            String(msg.stTrafficCar.szPlateType).trim();
                        trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                        trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                        trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                            trafficInfo.m_IllegalPlace =
                            ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                            ss);
                        trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                        trafficInfo.m_PlateColor = new
                            String(msg.stTrafficCar.szPlateColor).trim();
                        trafficInfo.m_VehicleColor = new
                            String(msg.stTrafficCar.szVehicleColor).trim();
                        trafficInfo.m_VehicleType = new
                            String(msg.stuVehicle.szObjectSubType).trim();
                        trafficInfo.m_VehicleSize =
                            Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                        trafficInfo.m_Utc = msg.UTC;
                        trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                        trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                        trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                        trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;


                                        break;
                                    }
                                    case
                            NetSDKLib.EVENT_IVS_TRAFFIC_RUNREDLIGHT: ///< running
                            red light event
                                    {
                            NetSDKLib.DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO msg =
                            new NetSDKLib.DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                            trafficInfo.m_EventName =
                            Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_R
                            UNREDLIGHT);
                        try {
                            trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                            "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                        trafficInfo.m_PlateType = new
                            String(msg.stTrafficCar.szPlateType).trim();
```

```
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                        trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case NetSDKLib.EVENT_IVS_TRAFFIC_OVERLINE:
                        ///< driving on lane event
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_OVERLINE_INFO msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_OVERLINE_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_O
                        VERLINE);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                            trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case
                        NetSDKLib.EVENT_IVS_TRAFFIC_RETROGRADE: ///<
                        wrong-way driving event
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_RETROGRADE_INFO msg =
                        new NetSDKLib.DEV_EVENT_TRAFFIC_RETROGRADE_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_RE
                        TROGRADE);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                    break;
                                }
                                case NetSDKLib.EVENT_IVS_TRAFFIC_TURNLEFT:
                        ///< illegal left turn
                                {
                        NetSDKLib.DEV_EVENT_TRAFFIC_TURNLEFT_INFO msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_TURNLEFT_INFO();
                                    ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_TU
                        RNLEFT);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                    } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                    }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case
                        NetSDKLib.EVENT_IVS_TRAFFIC_TURNRIGHT: ///< turning right
                        illegally
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_TURNRIGHT_INFO msg =
                        new NetSDKLib.DEV_EVENT_TRAFFIC_TURNRIGHT_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_TU
                        RNRIGHT);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case NetSDKLib.EVENT_IVS_TRAFFIC_UTURN: ///<
                        illegal U turn
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_UTURN_INFO msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_UTURN_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_UT
                        URN);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case
                        NetSDKLib.EVENT_IVS_TRAFFIC_OVERSPEED: ///< overspped
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_OVERSPEED_INFO msg =
                        new NetSDKLib.DEV_EVENT_TRAFFIC_OVERSPEED_INFO();
```

```
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                    trafficInfo.m_EventName =
                    Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_O
                    VERSPEED);
                try {
                    trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                    "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                trafficInfo.m_PlateType = new
                    String(msg.stTrafficCar.szPlateType).trim();
                trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                trafficInfo.m_IllegalPlace =
                    ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                    ss);
                trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                trafficInfo.m_PlateColor = new
                    String(msg.stTrafficCar.szPlateColor).trim();
                trafficInfo.m_VehicleColor = new
                    String(msg.stTrafficCar.szVehicleColor).trim();
                trafficInfo.m_VehicleType = new
                    String(msg.stuVehicle.szObjectSubType).trim();
                trafficInfo.m_VehicleSize =
                    Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                trafficInfo.m_Utc = msg.UTC;
                trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case
                    NetSDKLib.EVENT_IVS_TRAFFIC_UNDERSPEED: ///<
                    underspeed
                                    {
                    NetSDKLib.DEV_EVENT_TRAFFIC_UNDERSPEED_INFO msg =
                    new NetSDKLib.DEV_EVENT_TRAFFIC_UNDERSPEED_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                    trafficInfo.m_EventName =
                    Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_U
                    NDERSPEED);
                try {
                    trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                    "GBK").trim();
                            } catch (UnsupportedEncodingException e) {
                                e.printStackTrace();
                            }
                trafficInfo.m_PlateType = new
                    String(msg.stTrafficCar.szPlateType).trim();
                trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                trafficInfo.m_IllegalPlace =
                    ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                    ss);
                trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                trafficInfo.m_PlateColor = new
                    String(msg.stTrafficCar.szPlateColor).trim();
                trafficInfo.m_VehicleColor = new
                    String(msg.stTrafficCar.szVehicleColor).trim();
                trafficInfo.m_VehicleType = new
                    String(msg.stuVehicle.szObjectSubType).trim();
                trafficInfo.m_VehicleSize =
                    Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                trafficInfo.m_Utc = msg.UTC;
                trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                            break;
                        }
                        case NetSDKLib.EVENT_IVS_TRAFFIC_PARKING:
                    ///< illegally parking
                        {
                    NetSDKLib.DEV_EVENT_TRAFFIC_PARKING_INFO msg = new
                    NetSDKLib.DEV_EVENT_TRAFFIC_PARKING_INFO();
                            ToolKits.GetPointerData(pAlarmInfo, msg);

                    trafficInfo.m_EventName =
                    Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PA
                    RKING);
                try {
                    trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                    "GBK").trim();
                            } catch (UnsupportedEncodingException e) {
                                e.printStackTrace();
                            }
```

```
trafficInfo.m_PlateType = new
        String(msg.stTrafficCar.szPlateType).trim();
trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
trafficInfo.m_GroupID =    String.valueOf(msg.stuFileInfo.nGroupId);
trafficInfo.m_IllegalPlace =
        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
        ss);
trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
trafficInfo.m_PlateColor = new
        String(msg.stTrafficCar.szPlateColor).trim();
trafficInfo.m_VehicleColor = new
        String(msg.stTrafficCar.szVehicleColor).trim();
trafficInfo.m_VehicleType = new
        String(msg.stuVehicle.szObjectSubType).trim();
trafficInfo.m_VehicleSize =
        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
trafficInfo.m_Utc = msg.UTC;
trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                break;
                        }
                        case
        NetSDKLib.EVENT_IVS_TRAFFIC_WRONGROUTE: ///< driving
        on the wrong lane
                        {
        NetSDKLib.DEV_EVENT_TRAFFIC_WRONGROUTE_INFO msg =
        new
        NetSDKLib.DEV_EVENT_TRAFFIC_WRONGROUTE_INFO();
                                ToolKits.GetPointerData(pAlarmInfo, msg);

        trafficInfo.m_EventName =
        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_W
        RONGROUTE);
try {
        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
        "GBK").trim();
                                } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                }
trafficInfo.m_PlateType = new
        String(msg.stTrafficCar.szPlateType).trim();
trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
trafficInfo.m_IllegalPlace =
      ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
      ss);
trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
trafficInfo.m_PlateColor = new
      String(msg.stTrafficCar.szPlateColor).trim();
trafficInfo.m_VehicleColor = new
      String(msg.stTrafficCar.szVehicleColor).trim();
trafficInfo.m_VehicleType = new
      String(msg.stuVehicle.szObjectSubType).trim();
trafficInfo.m_VehicleSize =
      Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
trafficInfo.m_Utc = msg.UTC;
trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                        break;
                  }
                  case
      NetSDKLib.EVENT_IVS_TRAFFIC_CROSSLANE: ///< changing
      lanes illegally
                  {
      NetSDKLib.DEV_EVENT_TRAFFIC_CROSSLANE_INFO msg =
      new NetSDKLib.DEV_EVENT_TRAFFIC_CROSSLANE_INFO();
                        ToolKits.GetPointerData(pAlarmInfo, msg);

      trafficInfo.m_EventName =
      Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_C
      ROSSLANE);
try {
      trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
      "GBK").trim();
                        } catch (UnsupportedEncodingException e) {
                              e.printStackTrace();
                        }
trafficInfo.m_PlateType = new
      String(msg.stuTrafficCar.szPlateType).trim();
trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
trafficInfo.m_IllegalPlace =
      ToolKits.GetPointerDataToByteArr(msg.stuTrafficCar.szDeviceAddr
      ess);
trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
```

```
                    trafficInfo.m_PlateColor = new
                        String(msg.stuTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stuTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stuTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                    break;
                                }
                                case
                        NetSDKLib.EVENT_IVS_TRAFFIC_OVERYELLOWLINE: ///<
                        crossing yellow line
                                {
                        NetSDKLib.DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO
                        msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO();
                                    ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_O
                        VERYELLOWLINE);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                    } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                    }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                    break;
                                }
                                case
                        NetSDKLib.EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE:
///<yellow plate vehicle occupying lane
                                {
                        NetSDKLib.DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INF
                        O msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INF
                        O();
                                    ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_YE
                        LLOWPLATEINLANE);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                    } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                    }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                    break;
                                }
                                case
                        NetSDKLib.EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY:
///< pedestrian first event at the zebra areas
                                {
                        NetSDKLib.DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INF
                        O msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INF
                        O();
                                    ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PE
                        DESTRAINPRIORITY);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                    } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                    }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case
                        NetSDKLib.EVENT_IVS_TRAFFIC_MANUALSNAP:
///< traffic manually capturing event
                                    {
                        JOptionPane.showMessageDialog(null,
                        Res.string().getManualCaptureSucceed(),
                        Res.string().getPromptMessage(),
                        JOptionPane.INFORMATION_MESSAGE);
                        NetSDKLib.DEV_EVENT_TRAFFIC_MANUALSNAP_INFO msg =
                        new NetSDKLib.DEV_EVENT_TRAFFIC_MANUALSNAP_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_M
                        ANUALSNAP);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
                                    case
                        NetSDKLib.EVENT_IVS_TRAFFIC_VEHICLEINROUTE:
///< vehicle occupying lane
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_VEHICLEINROUTE_INFO
                        msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_VEHICLEINROUTE_INFO();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_VE
                        HICLEINROUTE);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;
```

```
                                        break;
                                    }
                                    case
                        NetSDKLib.EVENT_IVS_TRAFFIC_VEHICLEINBUSROUTE:
///<occupying public lanes
                                    {
                        NetSDKLib.DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INF
                        O msg = new
                        NetSDKLib.DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INF
                        O();
                                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_VE
                        HICLEINBUSROUTE);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                        } catch (UnsupportedEncodingException e) {
                                            e.printStackTrace();
                                        }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                        break;
                                    }
```

```
                              case NetSDKLib.EVENT_IVS_TRAFFIC_BACKING:
                        ///< reverse illegally event
                              {
                        NetSDKLib.DEV_EVENT_IVS_TRAFFIC_BACKING_INFO msg =
                        new NetSDKLib.DEV_EVENT_IVS_TRAFFIC_BACKING_INFO();
                                ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_BA
                        CKING);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                } catch (UnsupportedEncodingException e) {
                                    e.printStackTrace();
                                }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                break;
                              }
                              case
                        NetSDKLib.EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING:
///< parking space occupied
                              {
                        NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_I
                        NFO msg = new
```

```
                  NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_I
                  NFO();
                              ToolKits.GetPointerData(pAlarmInfo, msg);

                  trafficInfo.m_EventName =
                  Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PA
                  RKINGSPACEPARKING);
               try {
                  trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                  "GBK").trim();
                              } catch (UnsupportedEncodingException e) {
                                 e.printStackTrace();
                              }
               trafficInfo.m_PlateType = new
                  String(msg.stTrafficCar.szPlateType).trim();
               trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
               trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
               trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
               trafficInfo.m_IllegalPlace =
                  ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                  ss);
               trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
               trafficInfo.m_PlateColor = new
                  String(msg.stTrafficCar.szPlateColor).trim();
               trafficInfo.m_VehicleColor = new
                  String(msg.stTrafficCar.szVehicleColor).trim();
               trafficInfo.m_VehicleType = new
                  String(msg.stuVehicle.szObjectSubType).trim();
               trafficInfo.m_VehicleSize =
                  Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
               trafficInfo.m_Utc = msg.UTC;
               trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
               trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
               trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
               trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                              break;
                           }
                           case
                  NetSDKLib.EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING
                  :
///< parking space empty
                           {

                  NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKIN
                  G_INFO msg = new
```

```
                        NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKIN
                        G_INFO();
                                    ToolKits.GetPointerData(pAlarmInfo, msg);

                        trafficInfo.m_EventName =
                        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PA
                        RKINGSPACENOPARKING);
                    try {
                        trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                        "GBK").trim();
                                    } catch (UnsupportedEncodingException e) {
                                        e.printStackTrace();
                                    }
                    trafficInfo.m_PlateType = new
                        String(msg.stTrafficCar.szPlateType).trim();
                    trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                    trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                    trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                    trafficInfo.m_IllegalPlace =
                        ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddre
                        ss);
                    trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                    trafficInfo.m_PlateColor = new
                        String(msg.stTrafficCar.szPlateColor).trim();
                    trafficInfo.m_VehicleColor = new
                        String(msg.stTrafficCar.szVehicleColor).trim();
                    trafficInfo.m_VehicleType = new
                        String(msg.stuVehicle.szObjectSubType).trim();
                    trafficInfo.m_VehicleSize =
                        Res.string().getTrafficSize(msg.stTrafficCar.nVehicleSize);
                    trafficInfo.m_Utc = msg.UTC;
                    trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                    trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                    trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                    trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                    break;
                                }
                                case
                        NetSDKLib.EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT:
///< not wearing seat belt
                                {
                        NetSDKLib.DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT msg =
                        new NetSDKLib.DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT();
                                    ToolKits.GetPointerData(pAlarmInfo, msg);
                    trafficInfo.m_EventName =
                    Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_WI
                    THOUT_SAFEBELT);
                try {
                    trafficInfo.m_PlateNumber = new String(msg.stuObject.szText,
                    "GBK").trim();
                                } catch (UnsupportedEncodingException e) {
                                    e.printStackTrace();
                                }
                trafficInfo.m_PlateType = new
                    String(msg.stuTrafficCar.szPlateType).trim();
                trafficInfo.m_FileCount = String.valueOf(msg.stuFileInfo.bCount);
                trafficInfo.m_FileIndex = String.valueOf(msg.stuFileInfo.bIndex);
                trafficInfo.m_GroupID =   String.valueOf(msg.stuFileInfo.nGroupId);
                trafficInfo.m_IllegalPlace =
                    ToolKits.GetPointerDataToByteArr(msg.stuTrafficCar.szDeviceAddr
                    ess);
                trafficInfo.m_LaneNumber = String.valueOf(msg.nLane);
                trafficInfo.m_PlateColor = new
                    String(msg.stuTrafficCar.szPlateColor).trim();
                trafficInfo.m_VehicleColor = new
                    String(msg.stuTrafficCar.szVehicleColor).trim();
                trafficInfo.m_VehicleType = new
                    String(msg.stuVehicle.szObjectSubType).trim();
                trafficInfo.m_VehicleSize =
                    Res.string().getTrafficSize(msg.stuTrafficCar.nVehicleSize);
                trafficInfo.m_Utc = msg.UTC;
                trafficInfo.m_bPicEnble = msg.stuObject.bPicEnble;
                trafficInfo.m_OffSet = msg.stuObject.stPicInfo.dwOffSet;
                trafficInfo.m_FileLength = msg.stuObject.stPicInfo.dwFileLenth;
                trafficInfo.m_BoundingBox = msg.stuObject.BoundingBox;

                                break;
                            }
                            default:
                                break;
        }
```

11 Person and ID Card Comparison

11.1 Subscription to Event

11.1.1 Introduction

Check whether the person detected matches the ID card information.

11.1.2 Process Description

This chapter is only about callback of specific events. For event subscriprion and receiving, see "2.4 Subscribing Intelligent Event".

11.1.3 Enumeration and Structure

- Enumerated value corresponding to the event: EVENT_IVS_CITIZEN_PICTURE_COMPARE
- Structure corresponding to the event: DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO

11.2 Sample Code

```
        /* intelligent alarm event callback */
public static class fAnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack {
                    private BufferedImage snapBufferedImage = null;
                    private BufferedImage idBufferedImage = null;

                    private fAnalyzerDataCB() {}

                    private static class fAnalyzerDataCBHolder {
                        private static final fAnalyzerDataCB instance = new
                    fAnalyzerDataCB();
                    }
                    public static fAnalyzerDataCB getInstance() {
                        return fAnalyzerDataCBHolder.instance;
                    }

                        @Override
                        public int invoke(LLong lAnalyzerHandle, int dwAlarmType,
                                Pointer pAlarmInfo, Pointer pBuffer, int dwBufSize,
                                Pointer dwUser, int nSequence, Pointer reserved) {
                    if(pAlarmInfo == null) {
                        return 0;
                    }

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

                                                  switch(dwAlarmType)
                    {
                                                        case
                                        NetSDKLib.EVENT_IVS_CITIZEN_PICTURE_COMPARE:
// Person and ID card comparison
                                                        {
                                        DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO msg = new
                                        DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO();
                                                              ToolKits.GetPointerData(pAlarmInfo, msg);

                                  try {
                                        System.out.println("event occurrence time：" +
                                        msg.stuUTC.toString());
                                        System.out.println("event name :" + new String(msg.szName,
                                        "GBK").trim());

                                        // face and ID comparision result, similarity ≥threashold,
                                        comparision successful, 1-successful, 0-failure
                                                              System.out.println("comparison result:" +
                                        msg.bCompareResult);

                                                              System.out.println("image similiarity:" +
                                        msg.nSimilarity);
                                                              System.out.println("detection threashold:" +
                                        msg.nThreshold);

                                                              if (msg.emSex == 1) {
                                                                    System.out.println("gender: male");
                                                                    }else if (msg.emSex == 2){
                                                                          System.out.println("gender: female");
                                                                    }else {
                                                                          System.out.println("gender: known");
                                                                    }

                                                              // nationality
                                                              // 0- invalid data; 1-Han; 2-Mongol; 3-Hui; 4-Zang;
                                        5-Uyghur
                                                                    // 6-Miao; 7-Yi; 8-Zhuang; 9-Buyei;
                                        10-Korean; 11-Manchu; 12-Done
                                                                    // 13-Yao; 14-Bai; 15-Tujia; 16-Hani;
                                        17-Kazak; 18-Dai
                                                                    // 19-Li; 20-Lisu; 21-Va; 22-She; 23-
                                        Gaoshan; 24-Lahu
                                        // 25-Sui; 26-Dongxiang; 27-Naxi; 28-Jingpo;
                    29-Kirgiz
                                        // 30-Tu; 31-Daur; 32-Mulao; 33-Qiang;
                    34-Blang; 35-Salar
                    // 36-Maonan; 37-Gelao; 38-Xibe; 39-Achang; 40-Pumi; 41-Tajik
                                        // 42-Nu; 43-Uzbek; 44-Tussians; 45-Ewenki;
                    46-De'ang
                    // 47-Bonan; 48-Yugur; 49-Gin; 50-Tatar; 51-Derung; 52-Oroqen
                                        // 53-Hezhen; 54-Monba; 55-Lhoba; 56-Jino
                                    System.out.println("nation:" + msg.nEthnicity);

                    System.out.println("resident name:" + new String(msg.szCitizen,
                    "GBK").trim());
                    System.out.println("address:" + new String(msg.szAddress,
                    "GBK").trim());
                                    System.out.println("ID No.:" + new
                    String(msg.szNumber).trim());
                    System.out.println("issuing authority:" + new String(msg.szAuthority,
                    "GBK").trim());

                                    System.out.println("DOB:" +
                    msg.stuBirth.toStringTimeEx());
                    System.out.println("valid starting date:" +
                    msg.stuValidityStart.toStringTimeEx());
                    if (msg.bLongTimeValidFlag == 1) {
                           System.out.println("valid end date：forever");
                    }else{
                           System.out.println("valid end date:"+
                    msg.stuValidityEnd.toStringTimeEx());
                    }
                    System.out.println("IC card number：" + new String(msg.szCardNo,
                    "GBK").trim());
                                    } catch (Exception e) {
                                        e.printStackTrace();
                                    }

                // take images
                            String strFileName = path + "\\" +
                    System.currentTimeMillis() + "citizen_snap.jpg";
                            byte[] snapBuffer =
                    pBuffer.getByteArray(msg.stuImageInfo[0].dwOffSet,
                    msg.stuImageInfo[0].dwFileLenth);
                            ByteArrayInputStream snapArrayInputStream = new
                    ByteArrayInputStream(snapBuffer);
                            try {
                                    snapBufferedImage =
                    ImageIO.read(snapArrayInputStream);
```

```
                                        if(snapBufferedImage == null) {
                                            return 0;
                                        }
                                        ImageIO.write(snapBufferedImage, "jpg", new
                        File(strFileName));
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }

                                // ID card image
                                strFileName = path + "\\" + System.currentTimeMillis() +
                        "citizen_id.jpg";
                                byte[] idBuffer =
                        pBuffer.getByteArray(msg.stuImageInfo[1].dwOffSet,
                        msg.stuImageInfo[1].dwFileLenth);
                                ByteArrayInputStream idArrayInputStream = new
                        ByteArrayInputStream(idBuffer);
                                try {
                                        idBufferedImage =
                        ImageIO.read(idArrayInputStream);
                                        if(idBufferedImage == null) {
                                            return 0;
                                        }
                                        ImageIO.write(idBufferedImage, "jpg", new
                        File(strFileName));
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }

                                        break;
                                    }
                default:
                        break;
            }

                                return 0;
                            }
    }
```

12 Interface

12.1 SDK Initialization

12.1.1 CLIENT_Init

Table 12-1 SDK initialization CLIENT_Init

Options	**Description**	
Description	Initialize the whole SDK	
Function	public boolean CLIENT_Init(Callback cbDisConnect, Pointer dwUser);	
Parameter	[in]cbDisConnect	Disconnection callback function
	[in]dwUser	Userparameter of disconnection callback function
Return Value	Success: True; Failure: False	
Note	● Precondition of calling network SDK functions ● When callback function is NULL, if the device iss offline, will not be called back to users.	

12.1.2 CLIENT_Cleanup

Table 12-2 SDK clearing CLIENT_Cleanup

Options	**Description**
Description	Clear SDK
Function	public void CLIENT_Cleanup();
Parameter	None
Return Value	None
Note	SDK clearing interface, called before end

12.1.3 CLIENT_SetAutoReconnect

Table 12-3 Configuring disconnection callback function CLIENT_SetAutoReconnect

Options	**Description**	
Description	Configuring disconnection callback function	
Function	public void CLIENT_SetAutoReconnect(Callback cbAutoConnect, Pointer dwUser);	
Parameter	[in]cbAutoConnect	Disconnection callback function
	[in]dwUser	User parameter of disconnection callback function
Return Value	None	
Note	If callbackfunction interface is NULL, the device will not auto reconnected.	

12.1.4 CLIENT_SetNetworkParam

Table 12-4 Configuring network parameter CLIENT_SetNetworkParam

Options	Description	
Description	Configuring network parameter	
Function	public void CLIENT_SetNetworkParam(NET_PARAM pNetParam);	
Parameter	[in]pNetParam	Parameters like network delay, reconnected times, and buffer size.
Return Value	None	
Note	Adjust as needed.	

12.2 Device Login

12.2.1 CLIENT_LoginWithHighLevelSecurity

Table 12-5 Log in to device CLIENT_LoginWithHighLevelSecurity

Options	Description	
Description	Log in to device	
Function	public LLong CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY pstOutParam);	
Parameter	[in]pstInParam	Input parameter
	[out]pstOutParam	Output parameter
Return Value	Success: handle; failure: 0	
Note	Packed in NetSDKLib nterfaces; called by the following method: CLIENT_LoginWithHighLevelSecurity(pstInParam, pstOutParam);	

12.2.2 CLIENT_Logout

Table 12-6 Log out CLIENT_Logout

Options	Description	
Description	Lug out of the device	
Function	public boolean CLIENT_Logout(LLong lLoginID);	
Parameter	[in]lLoginID	Value returned by CLIENT_LoginWithHighLevelSecurity
Return Value	Success: true; failure: false	
Description	Packed in NetSDKLib nterfaces; called by the following method: CLIENT_Logout(m_hLoginHandle);	

12.3 Real-time Monitoring

12.3.1 CLIENT_RealPlayEx

Table 12-7 Open real-time monitoring CLIENT_RealPlayEx

Options	Description	
Description	Open real-time monitoring	
Function	public LLong CLIENT_RealPlayEx(LLong lLoginID, int nChannelID, Pointer hWnd, int rType);	
Parameter	[in]lLoginID	Value returned by CLIENT_LoginWithHighLevelSecurity
	[in]nChannelID	Video channel number, integer start from 0
	[in]hWnd	Window handle, valid only in Windows
	[in]rType	Live view types
Return Value	Success: non 0; failure: 0	
Note	Windows: ● 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 12-8 Description of preview type

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.
DH_RType_Multiplay_36	Multi-picture preview—36 pictures.

12.3.2 CLIENT_StopRealPlayEx

Table 12-9 CLIENT_StopRealPlayEx

Options	Description
Description	Stop the real-time monitoring.

Options	Description	
Function	public boolean CLIENT_StopRealPlayEx(LLong lRealHandle);	
Parameter	[in]lRealHandle	The return value of CLIENT_RealPlayEx
Return Value	Success: TRUE. Failure: FALSE.	
Description	None.	

12.4 Subscribing Intelligent Event

12.4.1 CLIENT_RealLoadPictureEx

Table 12-10 Subscribing Intelligent Event　CLIENT_RealLoadPictureEx

Options	Description	
Description	Subscribing Intelligent Event	

Function	public LLong CLIENT_RealLoadPictureEx(
	　　LLong lLoginID, int nChannelID,
	　　int dwAlarmType, int bNeedPicFile,
	　　StdCallCallback cbAnalyzerData,
	　　Pointer dwUser, Pointer Reserved);	

Parameter	[in]lLoginID	The value returned by CLIENT_LoginWithHighLevelSecurity
	[in]nChannelID	Device channel number. (from 0)
	[in]dwAlarmType	Intelligent traffic event type.
	[in]bNeedPicFile	Whether to subscribe picture file
	[in]cbAnalyzerData	Intelligent data analysis callback function.
	[in]dwUser	The user parameters.
	[in]Reserved	Reserve parameter.
Return Value	Success: LLONG subscribing handle; failure: 0	
Note	If interface failed to return, use CLIENT_GetLastError to get error code	

Table 12-11 Descriptiom of Intelligent event

dwAlarmType definition	value	Definition	Callback pAlarmInfo structural body
EVENT_IVS_ALL	0x00000001	All events	None
EVENT_IVS_CROSSFENCEDETECTION	0x0000011F	Crossing fence	DEV_EVENT_CROSSFENCEDETECTION_INFO
EVENT_IVS_CROSSLINEDETECTION	0x00000002	Tripwire	DEV_EVENT_CROSSLINE_INFO
EVENT_IVS_CROSSREGIONDETECTION	0x00000003	intrusion	DEV_EVENT_CROSSREGION_INFO
EVENT_IVS_LEFTDETECTION	0x00000005	Abandoned object	DEV_EVENT_LEFT_INFO
EVENT_IVS_PRESERVATION	0x00000008	Preserved object	DEV_EVENT_PRESERVATION_INFO
EVENT_IVS_TAKENAWAYDETECTION	0x00000115	Missing object	DEV_EVENT_TAKENAWAYDETECTION_INFO

dwAlarmType definition	value	Definition	Callback pAlarmInfo structural body
EVENT_IVS_WANDERDETECTION	0x00000007	Loitering detection	DEV_EVENT_WANDER_INFO
EVENT_IVS_VIDEOABNORMALDETECTION	0x00000013	Video abnormal detection	DEV_EVENT_VIDEOABNORMALDETECTION_INFO
EVENT_IVS_AUDIO_ABNORMALDETECTION	0x00000126	Audio abnormal detection	DEV_EVENT_IVS_AUDIO_ABNORMALDETECTION_INFO
EVENT_IVS_CLIMBDETECTION	0x00000128	Climbing Detection	DEV_EVENT_IVS_CLIMB_INFO
EVENT_IVS_FIGHTDETECTION	0x0000000E	Fighting Detection	DEV_EVENT_FLOWSTAT_INFO
EVENT_IVS_LEAVEDETECTION	0x00000129	Leave Post Detection	DEV_EVENT_IVS_LEAVE_INFO
EVENT_IVS_PRISONERRISEDETECTION	0x0000011E	Getting up Detection	DEV_EVENT_PRISONERRISEDETECTION_INFO
EVENT_IVS_PASTEDETECTION	0x00000004	Illegal sticker detection	DEV_EVENT_PASTE_INFO

12.4.2 CLIENT_StopLoadPic

Table 12-12 Stop subscribing intelligent event CLIENT_StopLoadPic

Options	Description	
Description	Stop subscribing intelligent event	
Function	public boolean CLIENT_StopLoadPic(LLong lAnalyzerHandle);	
Parameter	[in]lAnalyzerHandle	Intelligent event subscribing handle
Return Value	BOOL type ● Success: TRUE ● Failure: FALSE	
Note	If interface failed to return, use CLIENT_GetLastError to get error code	

12.5 Subscribing People Counting

12.5.1 CLIENT_AttachVideoStatSummary

Table 12-13 Subscribing people counting event CLIENT_AttachVideoStatSummary

Options	Description
Description	Subscribing people counting event
Function	public LLong CLIENT_AttachVideoStatSummary(LLong lLoginID, NET_IN_ATTACH_VIDEOSTAT_SUM pInParam, NET_OUT_ATTACH_VIDEOSTAT_SUM pOutParam, int nWaitTime);

Options	Description	
Parameter	[in] lLoginID	Login handle
	[in] pInParam	The input parameter of subscribing people counting
	[out] pOutParam	The output parameter of subscribing people counting
	[in] nWaitTime	timeout
Return Value	People counting subscribing handle	
Note	None	

12.5.2 CLIENT_DetachVideoStatSummary

Table 12-14 Cancel subscribing people counting event CLIENT_DetachVideoStatSummary

Options	Description	
Description	Cancel subscribing people counting event	
Function	public boolean CLIENT_DetachVideoStatSummary(LLong lAttachHandle);	
Parameter	[in] lAttachHandle	People counting subscribing handle
Return Value	Success: TRUE; failure: FALSE	
Note	None	

13 Callback

13.1 Note

It is recommended that the callback function is written as static single instance mode; otherwise the memory will make the program crash.

13.2 fDisConnectCallBack

Table 13-1 Disconnection callback fDisConnectCallBack

Options	Description	
Description	Disconnection callback	

Function	public interface fDisConnect extends Callback {
	public void invoke(LLong lLoginID, String pchDVRIP, int nDVRPort, Pointer dwUser);
	}	

Parameter	[out]lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity
	[out]pchDVRIP	Disconnected device IP
	[out]nDVRPort	Disconnected device port
	[out]dwUser	User parameters for callback
Return Value	None	
Note	None	

13.3 fHaveReConnectCallBack

Table 13-2 Reconnection callback fHaveReConnectCallBack

Options	Description	
Description	Reconnection callback	

Function	public interface fHaveReConnect extends Callback {
	public void invoke(LLong lLoginID, String pchDVRIP, int nDVRPort, Pointer dwUser);
	}	

Parameter	[out]lLoginID	Return value of CLIENT_LoginWithHighLevelSecurity
	[out]pchDVRIP	Reconnected device IP
	[out]nDVRPort	Reconnected device port
	[out]dwUser	User parameters for callback
Return Value	None	
Note	None	

13.4 fRealDataCallBackEx

Table 13-3 Real-time monitoring data callback fRealDataCallBackEx

Options	Description	
Description	Real-time monitoring data callback	

Function	public interface fRealDataCallBackEx extends Callback {
	public void invoke(LLong lRealHandle, int dwDataType, Pointer pBuffer, int dwBufSize, int param, Pointer dwUser);
	}	

Parameter	[out]lRealHandle	Return value of CLIENT_RealPlayEx
	[out]dwDataType	Data type: 0 indicates original data, and 2 indicates YUV data
	[out]pBuffer	Monitoring data block address
	[out]dwBufSize	Length of monitoring data block, in bytes
	[out]param	Parameter structure for callback data. The type is different if the dwDataType value is different. ● When dwDataType is 0, param is null pointer. ● When dwDataType is 2, param is the structure pointer tagCBYUVDataParam.
	[out]dwUser	User parameters for callback
Return Value	None	
Note	None	

13.5 fAnalyzerDataCallBack

Table 13-4 Intelligent Event Callback fAnalyzerDataCallBack

Options	Description	
Description	Remote device status callback	

Function	public interface fAnalyzerDataCallBack extends Callback {
	public int invoke(LLong lAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo, Pointer pBuffer, int dwBufSize, Pointer dwUser, int nSequence, Pointer reserved);
	}	

Parameter	[out]lAnalyzerHandle	Return value of CLIENT_RealLoadPictureEx
	[out]dwEventType	Intelligent event type
	pAlarmInfo	Event information cache
	[out]pBuffer	Image cache
	[out]dwBufSize	Image cache size
	[out]dwUser	User data
	[out]nSequence	ESN
	[out]reserved	Reserve
Return Value	None	
Note	After subscribing to the intelligent event of remote device, if an intelligent event is triggered, the camera will report relevant information of the event.	

13.6 fVideoStatSumCallBack

Options	Description	
Description	People counting event subscription callback	

Function	public interface fVideoStatSumCallBack extends Callback {
	public void invoke(LLong lAttachHandle, NET_VIDEOSTAT_SUMMARY pBuf, int dwBufLen, Pointer dwUser);
	}	

Parameter	[out] lAttachHandle	People counting subscription handle
	[out] pBuf	People counting return data
	[out]dwBufLen	Length of return data
	[out]dwUser	User data
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.