

NetSDK_JAVA (Intelligent AI)






Programming Manual



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
V1.0.3	Add Appendix 2 Intelligent Events	December 2023
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
Target detection	Detect the faces and their feature information (age, gender, and expression) through the intelligent analysis of videos.
Target recognition	Detect whether the faces are in the armed face library through the intelligent analysis of videos, including target detection.
Face library	Detect whether the faces are in the face library 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

Foreword	II
Glossary	III
1 Overview	1
1.1 General.....	1
1.2 Applicability	2
1.3 Application Scenarios.....	2
1.3.1 Target Detection/ Target Recognition/Human Detection	2
1.3.2 People Counting	2
1.3.3 Intelligent Traffic	3
1.3.4 General Behavior.....	4
1.3.5 Turnstile	5
2 General Functions	7
2.1 NetSDK Initialization	7
2.1.1 Introduction.....	7
2.1.2 Interface Overview	7
2.1.3 Process Description.....	7
2.1.4 Sample Code.....	8
2.2 Device Login	10
2.2.1 Introduction.....	10
2.2.2 Interface Overview	10
2.2.3 Process Description.....	11
2.2.4 Sample Code.....	12
2.3 Real-Time Monitoring.....	14
2.3.1 Introduction.....	14
2.3.2 Interface Overview	14
2.3.3 Process Description.....	14
2.3.4 Sample Code.....	18
2.4 Subscription to Intelligent Event.....	20
2.4.1 Introduction.....	20
2.4.2 Interface Overview	20
2.4.3 Process Description.....	21
2.4.4 Sample Code.....	22
3 Target Detection	25
3.1 Subscription to Event	25
3.1.1 Introduction.....	25
3.1.2 Process Description.....	25
3.1.3 Enumeration and Structure	25
3.2 Sample Code.....	25
4 Target Recognition	27
4.1 Subscription to Event	27
4.1.1 Introduction.....	27
4.1.2 Process Description.....	27
4.1.3 Enumeration and Structure	27
4.2 Sample Code.....	27

5 General Behavior.....	30
5.1 Subscription to Event	30
5.1.1 Introduction.....	30
5.1.2 Process Description.....	30
5.1.3 Enumeration and Structure	30
5.2 Sample Code.....	30
6 Human Detection	33
6.1 Subscription to Event	33
6.1.1 Introduction.....	33
6.1.2 Process Description.....	33
6.1.3 Enumeration and Structure	33
6.2 Sample Code.....	33
7 Thermal Temperature Event.....	36
7.1 Subscription to Event	36
7.1.1 Introduction.....	36
7.1.2 Process Description.....	36
7.1.3 Enumeration and Structure	36
7.2 Sample Code.....	36
8 Access Control Event	38
8.1 Event Subscription.....	38
8.1.1 Introduction.....	38
8.1.2 Process Description.....	38
8.1.3 Enumeration and Structure	38
8.2 Sample Code.....	38
9 People Counting	40
9.1 Introduction	40
9.2 Interface Overview	40
9.3 Process Description.....	41
9.4 Sample Code.....	41
10 Intelligent Traffic Event.....	44
10.1 Subscription to Event.....	44
10.1.1 Introduction	44
10.1.2 Process Description	44
10.1.3 Enumeration and Structure	44
10.2 Sample Code	46
11 Person and ID Card Comparison.....	67
11.1 Subscription to Event.....	67
11.1.1 Introduction	67
11.1.2 Process Description	67
11.1.3 Enumeration and Structure	67
11.2 Sample Code	67
12 Interface.....	71
12.1 SDK Initialization.....	71
12.1.1 CLIENT_Init	71
12.1.2 CLIENT_Cleanup	71
12.1.3 CLIENT_SetAutoReconnect	71
12.1.4 CLIENT_SetNetworkParam	72

12.2 Device Login.....	72
12.2.1 CLIENT_LoginWithHighLevelSecurity.....	72
12.2.2 CLIENT_Logout.....	72
12.3 Real-time Monitoring.....	73
12.3.1 CLIENT_RealPlayEx.....	73
12.3.2 CLIENT_StopRealPlayEx.....	73
12.4 Subscribing Intelligent Event.....	74
12.4.1 CLIENT_RealLoadPictureEx.....	74
12.4.2 CLIENT_StopLoadPic.....	75
12.5 Subscribing People Counting.....	75
12.5.1 CLIENT_AttachVideoStatSummary.....	75
12.5.2 CLIENT_DetachVideoStatSummary.....	76
13 Callback.....	77
13.1 Note.....	77
13.2 fDisconnectCallBack.....	77
13.3 fHaveReConnectCallBack.....	77
13.4 fRealDataCallBackEx.....	77
13.5 fAnalyzerDataCallBack.....	78
13.6 fVideoStatSumCallBack.....	78
Appendix 1 Cybersecurity Recommendations.....	80
Appendix 2 Intelligent Event.....	82

1 Overview

1.1 General

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

The main functions include general functions, target detection, target recognition, general behavior event, human detection, thermal temperature, 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	dhnet sdk.h	Header file
	dhnet sdk.lib	Lib file
	dhnet sdk.dll	Library file
	avnet sdk.dll	Library file
Configuration library	avglobal.h	Header file
	dhconfig sdk.h	Header file
	dhconfig sdk.lib	Lib file
	dhconfig sdk.dll	Library file
Play (encoding/decoding) auxiliary library	dhplay.dll	Play 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	dhnet sdk.h	Header file
	libdhnet sdk.so	Library file
	libavnet sdk.so	Library file
Configuration library	avglobal.h	Header file
	dhconfig sdk.h	Header file
	libdhconfig sdk.so	Library file



- 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 avnet sdk.dll or libavnet sdk.so, the corresponding auxiliary library is required.

1.2 Applicability

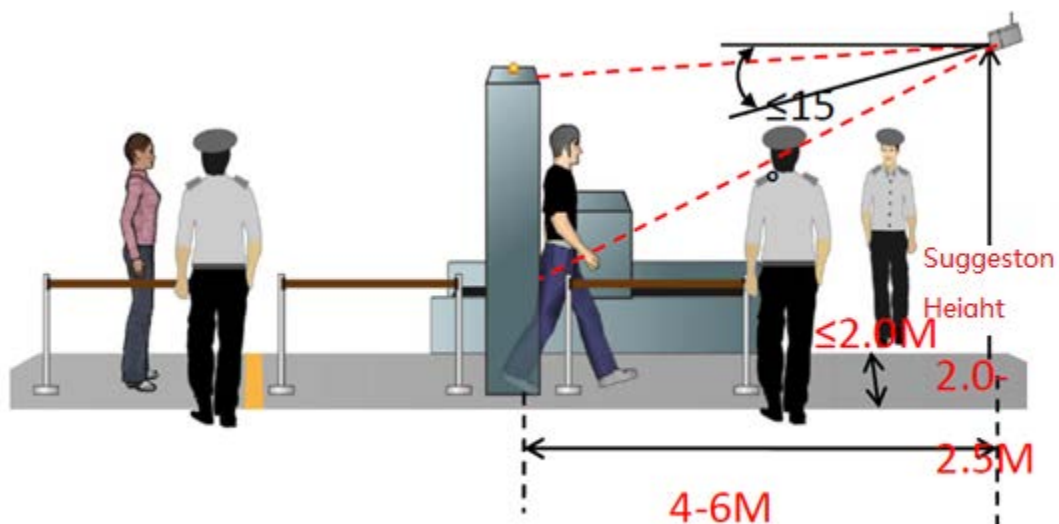
- Recommended memory: No less than 512 M.
- Jdk version: jdk1.6; jdk1.8.
- Systems supported by SDK:
 - ◇ Windows 10/Windows 8.1/Windows 7/ 2000 and Windows Server 2008/2003.
 - ◇ Linux
General Linux system like Red Hat/SUSE

1.3 Application Scenarios

1.3.1 Target Detection/ Target Recognition/Human Detection

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

Figure 1-1 Target recognition



1.3.2 People Counting

For the application of people counting products in the actual scenario, see Figure 1-2.

Figure 1-2 People counting scenario



1.3.3 Intelligent Traffic

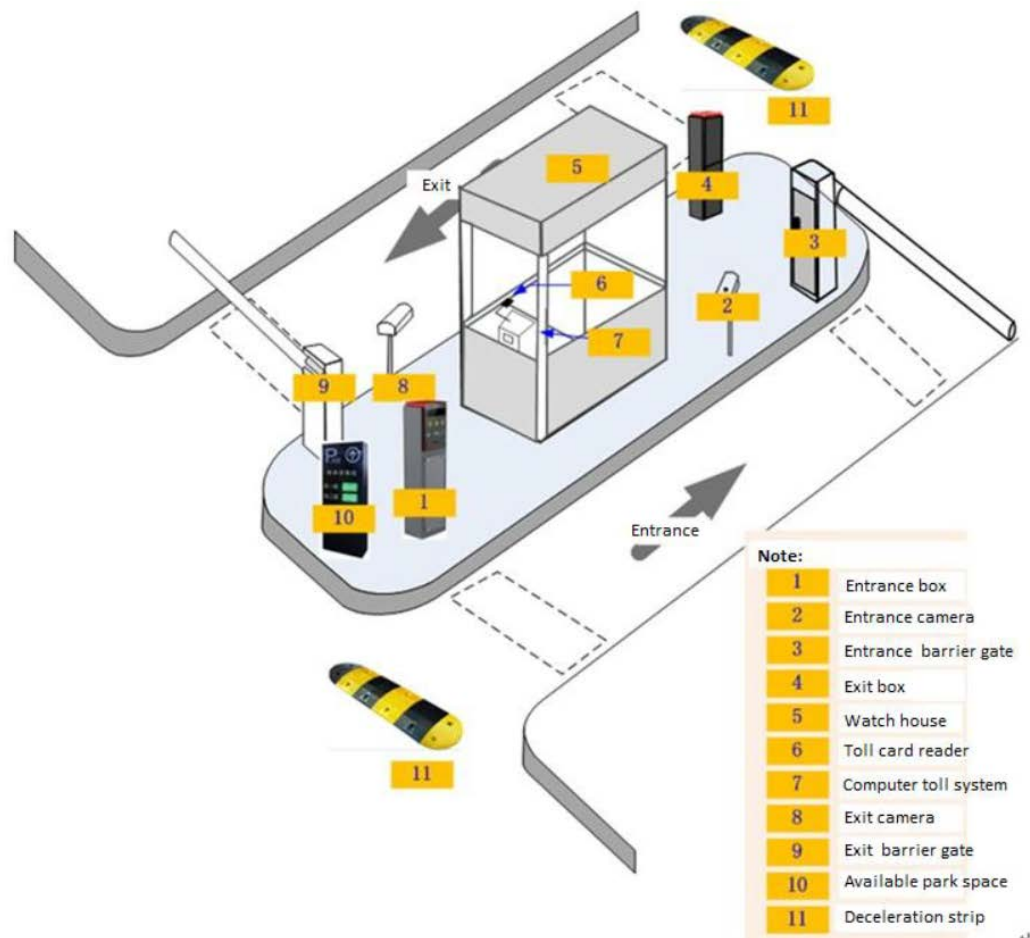
- ITC at the intersection is used for capturing traffic violations and traffic flow statistics, see Figure 1-3.

Figure 1-3 Applications of ITC at the intersection

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 1	Normal
5	2020-07-04 ...	junction (卡口)	1353045725	1	1	皖A7 3	Normal
4	2020-07-04 ...	junction (卡口)	1352718015	1	1	浙AL 1	Normal
3	2020-07-04 ...	junction (卡口)	1352521403	1	1	浙AX 7	Normal
2	2020-07-04 ...	over speed (超速)	1351866037	1	1	浙A1 N	Normal
1	2020-07-04 ...	over speed (超速)	1351407281	1	1	浙A5 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 1-4.

Figure 1-4 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 1-5 and Figure 1-6.

Figure 1-5 General behavior scenario—tripwire



Figure 1-6 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.

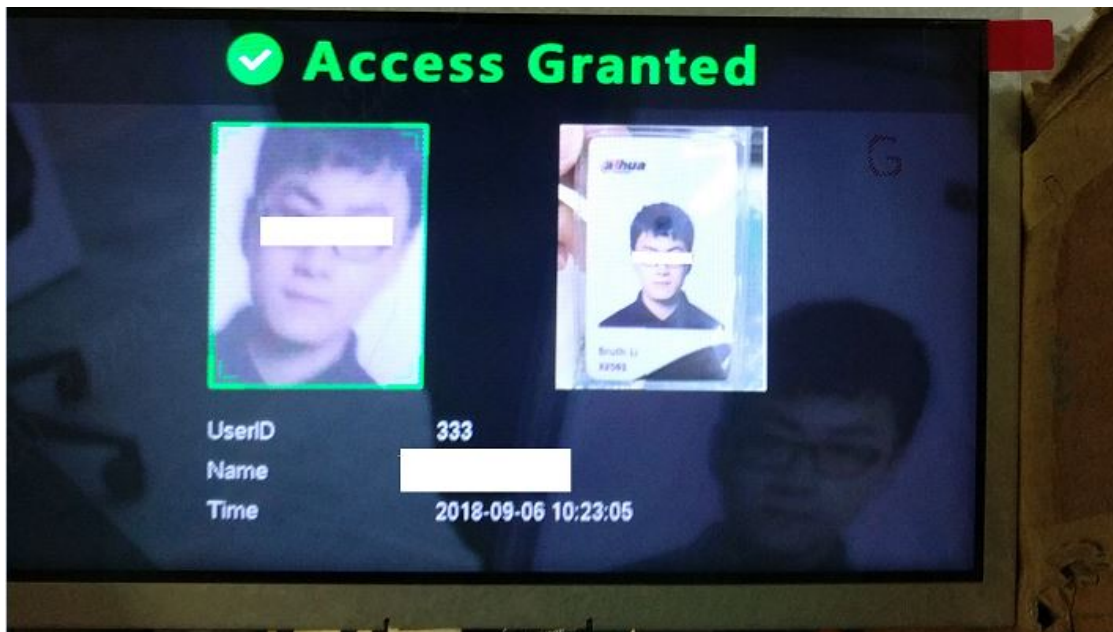
For the appearance of the access control turnstile, see Figure 1-7.

Figure 1-7 Swing turnstile appearance



You can unlock the turnstile by face or swiping card. For unlocking by face, see Figure 1-8.

Figure 1-8 Unlocking by face



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.

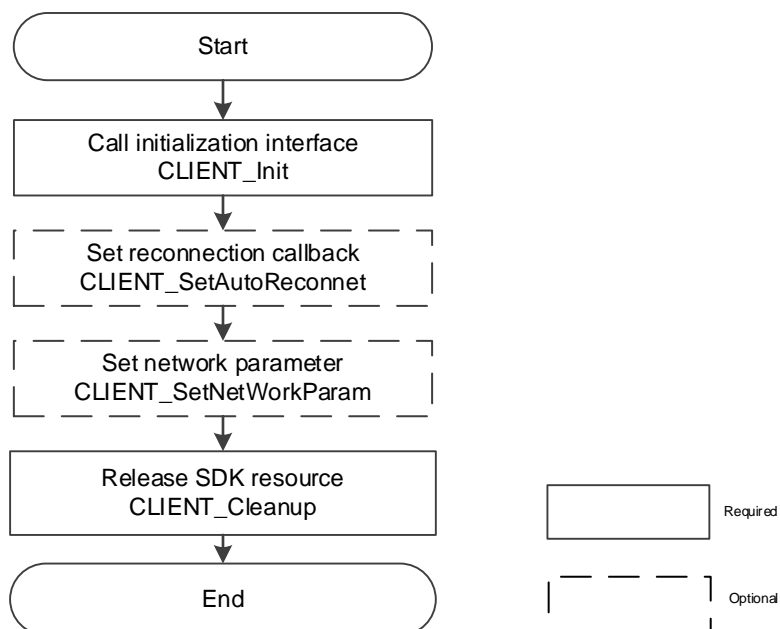
2.1.2 Interface Overview

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

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

- Step 2** (Optional) Call **CLIENT_SetAutoReconnect** to set reconnection callback to allow the auto reconnecting after disconnection 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.getAbsolutePath().getParent() + "\\sdklog\\" +
ToolKits.getDate() + ".log";
        setLog.nPrintStrategy = 0;
        setLog.bSetFilePath = 1;
        System.arraycopy(logPath.getBytes(), 0, setLog.szLogFilePath, 0,
logPath.getBytes().length);
        System.out.println(logPath);
        setLog.bSetPrintStrategy = 1;
        bLogopen = netsdk.CLIENT_LogOpen(setLog);
        if(!bLogopen ) {
            System.err.println("Failed to open NetSDK log");
        }

        // 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(blNit) {
            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.

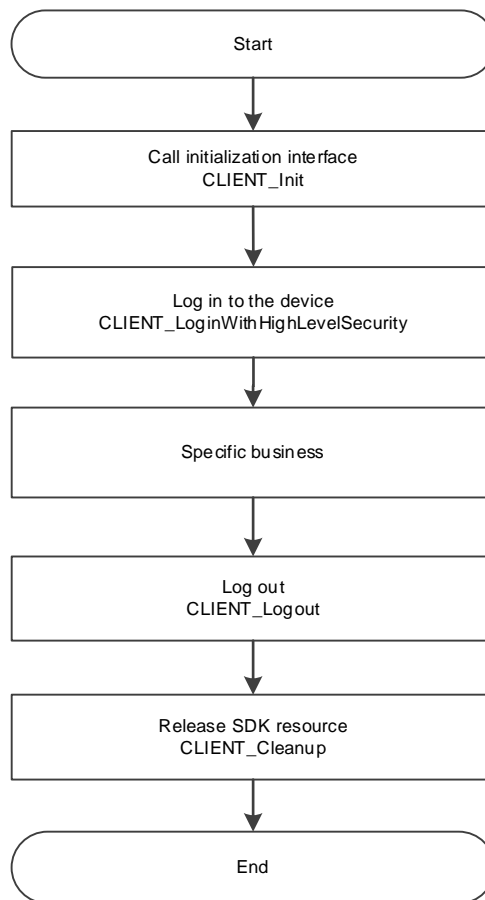
2.2.2 Interface Overview

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: <code>NET_PARAM stuNetParam = new NET_PARAM(); stuNetParam.nWaittime = 8000; // unit ms CLIENT_SetNetworkParam (stuNetParam);</code>
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_HIGHELEVEL_SECURITY pstInParam=
            new NET_IN_LOGIN_WITH_HIGHELEVEL_SECURITY();
            pstInParam.szIP= m_strIp;
            pstInParam.nport= m_nPort;
            pstInParam.szUserName= m_strUser;
            pstInParam.szPassword= m_strPassword;
            //Input parameter
            NET_OUT_LOGIN_WITH_HIGHELEVEL_SECURITY pstOutParam=
            new NET_OUT_LOGIN_WITH_HIGHELEVEL_SECURITY();
            m_hLoginHandle =
            netsdk.CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIG
            HLEVEL_SECURITY pstInParam,
            NET_OUT_LOGIN_WITH_HIGHELEVEL_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

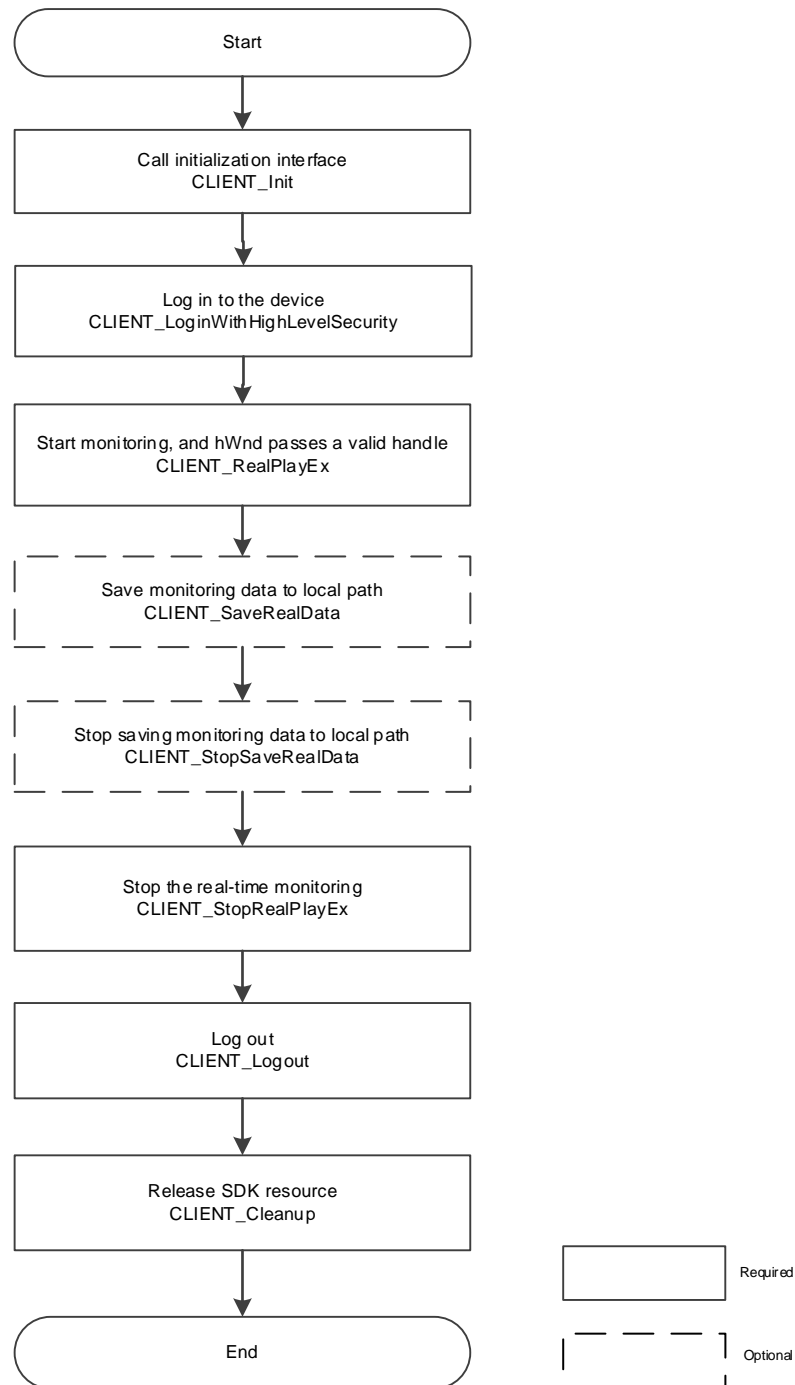
2.3.3 Process Description

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



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.

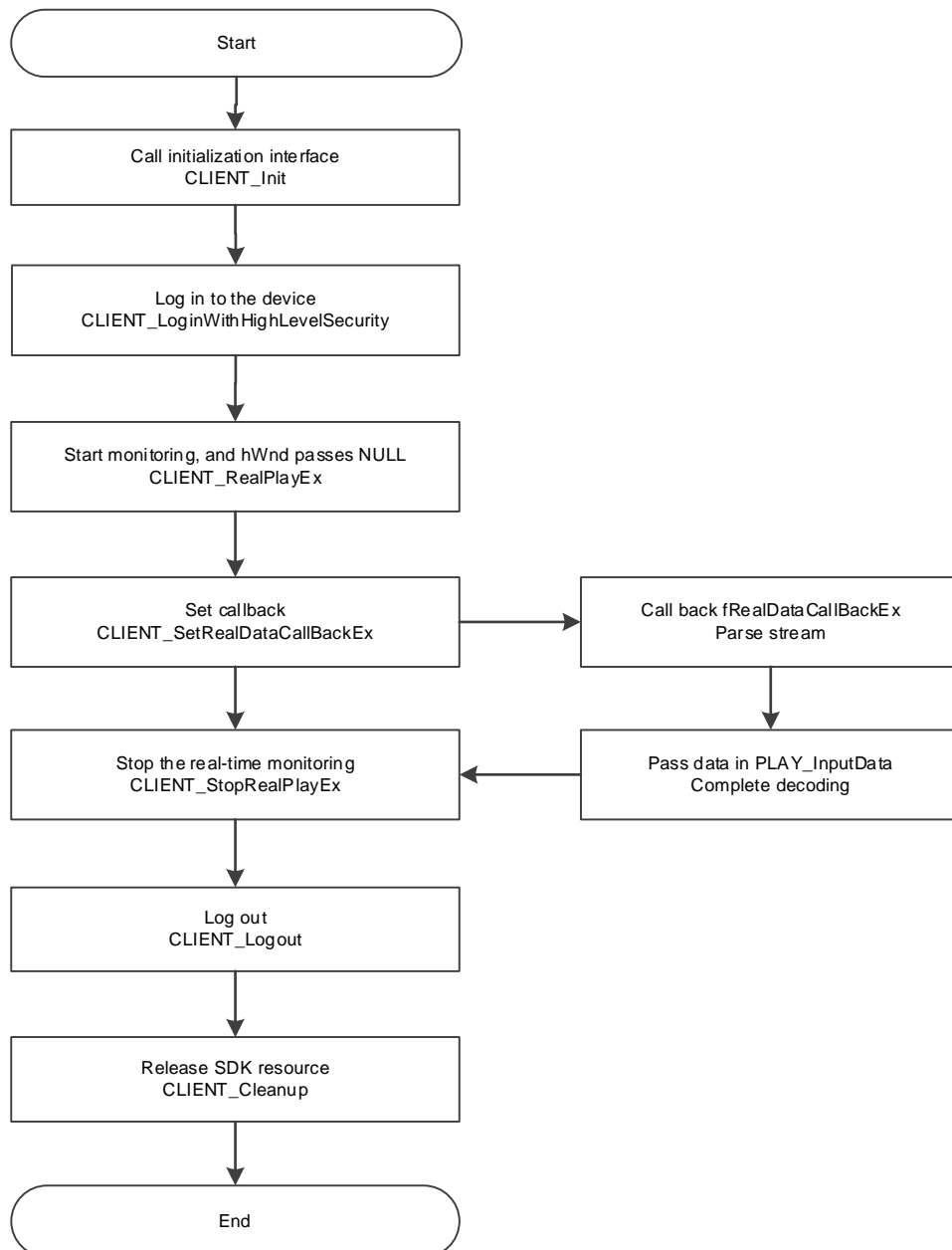
```
NET_PARAM stuNetParam = new NET_PARAM();  
stuNetParam.nGetConnInfoTime = 5000; if the value is 0, it is 1000 ms by default  
CLIENT_SetNetworkParam (stuNetParam);
```

- 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_FAILED_EVENT in the alarm callback that is set in CLIENT_SetDVRMessCallBack. This event includes the detailed error codes. See "DEV_PLAY_RESULT Structure" in Network SDK Development Manual.

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



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_hLoginHandle,
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 IRealHandle, 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_hLoginHandl
        e, channel, Native.getComponentPointer(realPlayWindow), stream);

        LoginModule.netsdk.CLIENT_SetRealDataCallBackEx(m_hPlayHand
        le,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.

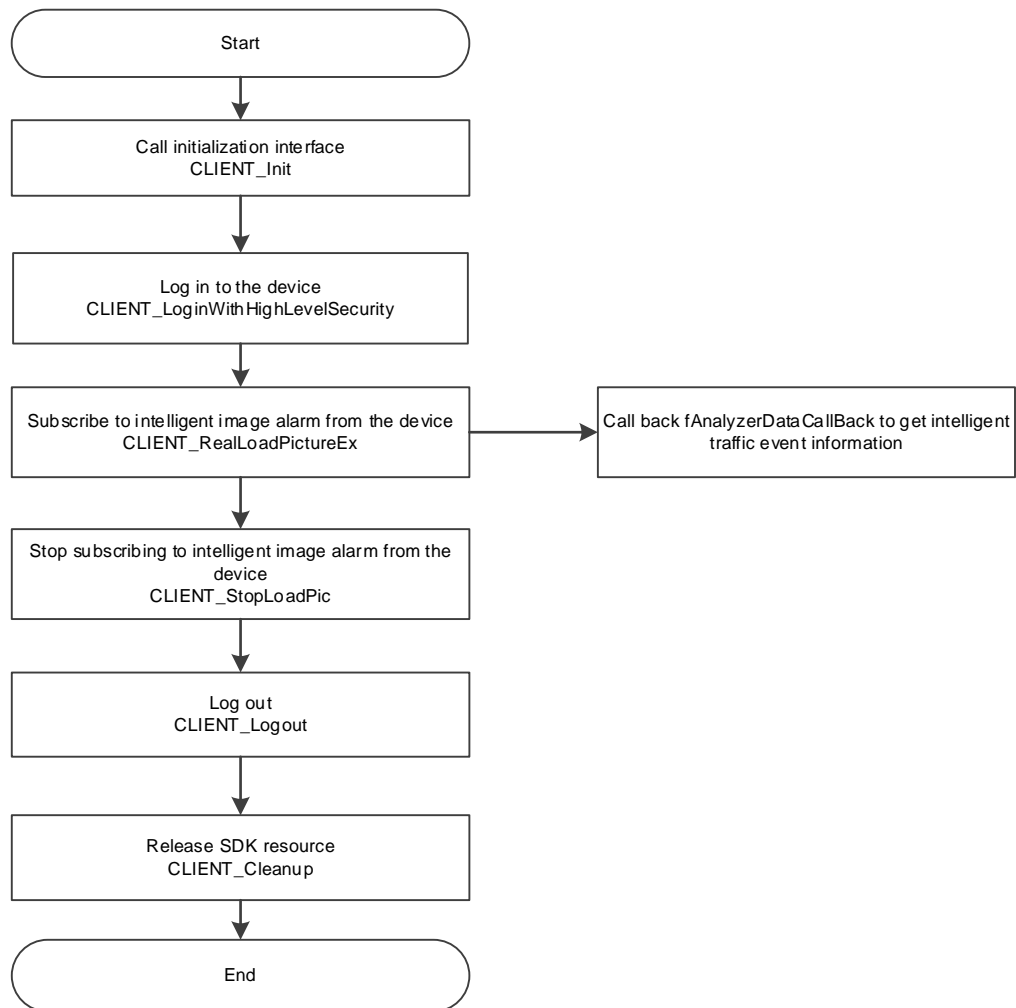
2.4.2 Interface Overview

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



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 traffic products 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 Target 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 subscripion 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 IAnalyzerHandle, int dwAlarmType,
                    Pointer pAlarmInfo, Pointer pBuffer, int
                    dwBufSize,
                    Pointer dwUser, int nSequence, Pointer
                    reserved)
    {
        if (IAnalyzerHandle.longValue() == 0 || pAlarmInfo == null) {
            return -1;
        }
    }
}
```

```

        switch(dwAlarmType)
        {
            case NetSDKLib.EVENT_IVS_FACEDETECT:    ///< target
                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 Target Recognition

4.1 Subscription to Event

4.1.1 Introduction

Target 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 target 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: ///  

target 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 subscription 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 (IAnalyzerHandle == null || IAnalyzerHandle.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.stuScenelImage.nLength>0)
                {
                    String strFileName = path + "\\\" + System.currentTimeMillis() +
                    "HumanTrait_ panoramaic image.jpg";
                    ToolKits.savePicture(pBuffer, msg.stuScenelImage.nOffSet,
                    msg.stuScenelImage.nLength, strFileName);
                }
                else
                {
                    System.out.println("no panoramaic image");
                }

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

                //save face panoramaic image
            }
        }
    }
}

```



```

        if(msg.stuFaceScenelImage.nLength>0)
        {
String strFileName = path + "\\\" + System.currentTimeMillis() +
"HumanTrait_face panoramaic image.jpg";
ToolKits.savePicture(pBuffer, msg.stuFaceScenelImage.nOffset,
msg.stuFaceScenelImage.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 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

```
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 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 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.stMan
            TempInfo.dbHighTemp+"
            nTempUnit"+msg.stManTempInfo.nTempUnit+"blsOverTemp"+msg.stM
            anTempInfo.blsOverTemp+"blsUnderTemp"+msg.stManTempInfo.blsUn
            derTemp+"\n");

            //Visible image panorama
            if(msg.stVisScenelImage!=null &&
            msg.stVisScenelImage.nLength> 0){
            String bigPicture = picturePath + "\\" + System.currentTimeMillis() +
            ".jpg";
            ToolKits.savePicture(pBuffer, msg.stVisScenelImage.nOffset,
            msg.stVisScenelImage.nLength, bigPicture);
            }
            //Thermal imaging panorama
            if(msg.stThermalScenelImage!=null &&
            msg.stThermalScenelImage.nLength> 0){
            String smallPicture = picturePath + "\\" + System.currentTimeMillis() + "small.jpg";
            ToolKits.savePicture(pBuffer, msg.stThermalScenelImage.nOffset,
            msg.stThermalScenelImage.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.

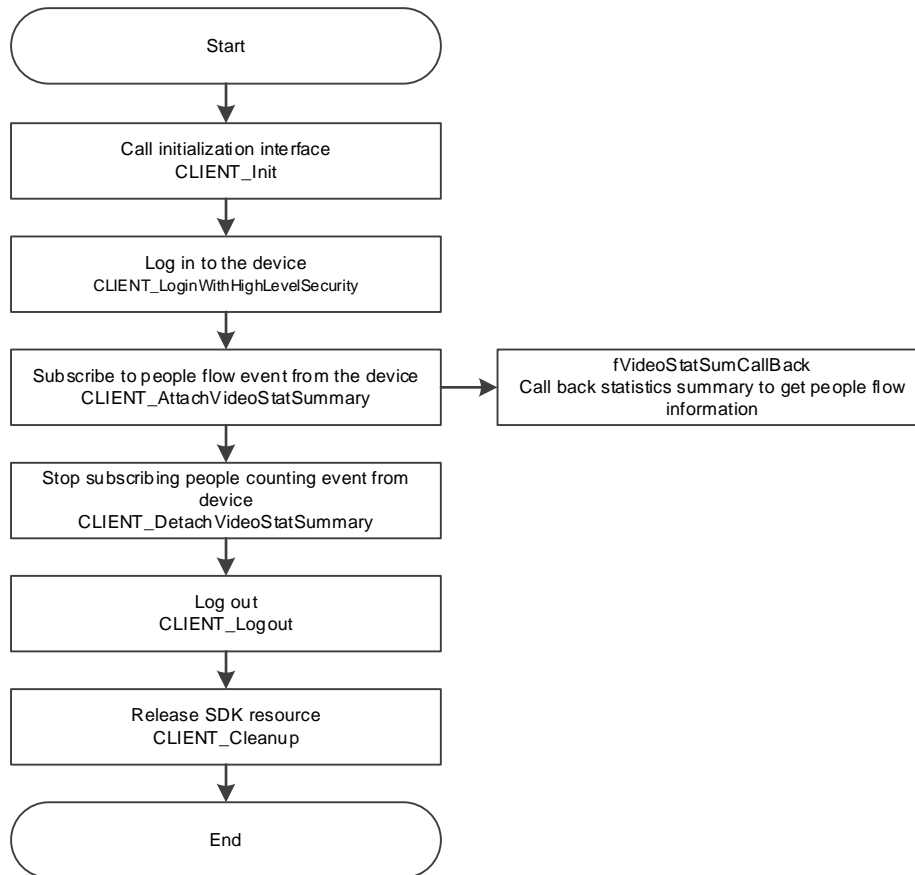
9.2 Interface Overview

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



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(videoStatHandle);
            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 IAttachHandle,
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 subscription 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_UNDEERSPEED
Structure corresponding to the event of traffic violation—underspeed:
DEV_EVENT_TRAFFIC_UNDEERSPEED_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_TRAFFICJUNCTION);
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.szDeviceAddress);
            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_RUNREDLIG
                HT);
            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.szDeviceAddress);
            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_OVERLINE);
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.szDeviceAddress);
            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_RETROGRADE);
    }
    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.GetPointerDataToByteArray(msg.stTrafficCar.szDeviceAddress);
    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_TURNLEFT);
    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.szDeviceAddress);
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_TURNRIGHT
                );
            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.szDeviceAddress);
            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_UTURN);
                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.szDeviceAddress);
                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_OVERSPEED
                );
            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.szDeviceAddress);
            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_UNDEERSPEED_INFO();
ToolKits.GetPointerData(pAlarmInfo, msg);

trafficInfo.m_EventName =
Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_UNDEERSPE
ED);
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.szDeviceAddress);
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_PARKING);
        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.szDeviceAddress);
        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_WRONGROUTE);
        }
    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.szDeviceAddress);
        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_CROSSLAN  

            E);
        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.szDeviceAddress);
        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_OVERYELLOWLINE);
    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.szDeviceAddress);
    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_INFO msg =
                new NetSDKLib.DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO();
                ToolKits.GetPointerData(pAlarmInfo, msg);

                trafficInfo.m_EventName =
                Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_YELLOWPL
                ATEINLANE);
            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.szDeviceAddress);
                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_INFO msg =
                new NetSDKLib.DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO();

```

```

ToolKits.GetPointerData(pAlarmInfo, msg);

trafficInfo.m_EventName =
Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PEDESTRIAN
NPRIORITY);
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.szDeviceAddress);
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_MANUALSN

```

```

        AP);
    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.GetPointerDataToByteArray(msg.stTrafficCar.szDeviceAddress);
    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_VEHICLEINR
            OUTE);
    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.szDeviceAddress);
        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_INFO msg =
                new NetSDKLib.DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INFO();
            ToolKits.GetPointerData(pAlarmInfo, msg);

            trafficInfo.m_EventName =
                Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_VEHICLEINB
                USROUTE);
            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.szDeviceAddress);
            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_BACKING);
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.szDeviceAddress);
                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_INFO msg =
            new NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO();
            ToolKits.GetPointerData(pAlarmInfo, msg);

            trafficInfo.m_EventName =
            Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PARKINGSP
            ACEPARKING);
            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.nGroupld);
            trafficInfo.m_IllegalPlace =
                ToolKits.GetPointerDataToByteArr(msg.stTrafficCar.szDeviceAddress);
            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_PARKINGSPACENOPARKING_INFO  
        msg = new  
        NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO();  
        ToolKits.GetPointerData(pAlarmInfo, msg);  
  
        trafficInfo.m_EventName =  
        Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PARKINGSP  
        ACENOPARKING);  
    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.szDeviceAddress);  
    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_WITHOUT_
        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.szDeviceAddress);
    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 subscription 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 comparison result, similarity ≥threshold, comparison
                successful, 1-successful, 0-failure
                System.out.println("comparison result:" +
                msg.bCompareResult);

                System.out.println("image similarity:" +
                msg.nSimilarity);
                System.out.println("detection threshold:" +
                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");
                }

                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.stulmageInfo[0].dwOffSet,
msg.stulmageInfo[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.stulmageInfo[1].dwOffSet,
msg.stulmageInfo[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	<ul style="list-style-type: none">● 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: <ul style="list-style-type: none">• When hWnd is valid, the corresponding window displays picture.• When hWnd is NULL, get the video data through setting a callback and send to user for 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 IRealHandle);	
Parameter	[in]IRealHandle	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 ILoginID, int nChannelID, int dwAlarmType, int bNeedPicFile, StdCallCallback cbAnalyzerData, Pointer dwUser, Pointer Reserved);	
Parameter	[in]ILoginID	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 Description of Intelligent event

dwAlarmType definition	value	Definition	Callback structural body	pAlarmInfo
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_PSRISEDETECTION	0x0000011E	Getting up Detection	DEV_EVENT_PSRISEDETECTION_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 IAnalyzerHandle);	
Parameter	[in]IAnalyzerHandle	Intelligent event subscribing handle
Return Value	BOOL type <ul style="list-style-type: none"> ● 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 ILoginID, 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 ILoginID, String pchDVRIP, int nDVRPort, Pointer dwUser); }	
Parameter	[out]ILoginID	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 ILoginID, String pchDVRIP, int nDVRPort, Pointer dwUser); }	
Parameter	[out]ILoginID	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 {

Options	Description	
	<pre> public void invoke(LLong IRealHandle, int dwDataType, Pointer pBuffer, int dwBufSize, int param, Pointer dwUser); </pre>	
Parameter	[out]IRealHandle	Return value of CLIENT_RealPlayEx
	[out]dwDataType	Data type: 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. <ul style="list-style-type: none"> When dwDataType is 0, param is null pointer. When dwDataType is 2, param is the structure pointer tagCBYUVDDataParam.
	[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	<pre> public interface fAnalyzerDataCallBack extends Callback { public int invoke(LLong IAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo, Pointer pBuffer, int dwBufSize, Pointer dwUser, int nSequence, Pointer reserved); } </pre>	
Parameter	[out]IAnalyzerHandle	Return value of CLIENT_RealLoadPictureEx
	[out]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

Options	Description	
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.

Appendix 2 Intelligent Events

Type	Code	Remarks
EVENT_IVS_ALL	0x00000001	subscription all event
EVENT_IVS_CROSSLINEDETECTION	0x00000002	cross line event(Corresponding to DEV_EVENT_CROSSLINE_INFO)
EVENT_IVS_CROSSREGIONDETECTION	0x00000003	cross region event(Corresponding to DEV_EVENT_CROSSREGION_INFO)
EVENT_IVS_LEFTDETECTION	0x00000005	left event(Corresponding to DEV_EVENT_LEFT_INFO)
EVENT_IVS_STAYDETECTION	0x00000006	stay event(Corresponding to DEV_EVENT_STAY_INFO)
EVENT_IVS_WANDERDETECTION	0x00000007	wander event(Corresponding to DEV_EVENT_WANDER_INFO)
EVENT_IVS_PRESERVATION	0x00000008	reservation event(Corresponding to DEV_EVENT_PRESERVATION_INFO)
EVENT_IVS_MOVEDETECTION	0x00000009	move event(Corresponding to DEV_EVENT_MOVE_INFO)
EVENT_IVS_RIOTERDETECTION	0x0000000B	rioter event(Corresponding to DEV_EVENT_RIOTERL_INFO)
EVENT_IVS_FIREDETECTION	0x0000000C	fire event(Corresponding to DEV_EVENT_FIRE_INFO)
EVENT_IVS_SMOKEDETECTION	0x0000000D	smoke event(Corresponding to DEV_EVENT_SMOKE_INFO)
EVENT_IVS_FIGHTDETECTION	0x0000000E	fight event(Corresponding to DEV_EVENT_FIGHT_INFO)
EVENT_IVS_NUMBERSTAT	0x00000010	number stat event(Corresponding to DEV_EVENT_NUMBERSTAT_INFO)
EVENT_IVS_VIDEOABNORMALDETECTION	0x00000013	video abnormal event(Corresponding to DEV_EVENT_VIDEOABNORMALDETECTION_INFO)
EVENT_IVS_TRAFFICACCIDENT	0x00000016	traffic accident event(Corresponding to DEV_EVENT_TRAFFICACCIDENT_INFO)
EVENT_IVS_TRAFFICJUNCTION	0x00000017	traffic junction event(Corresponding to DEV_EVENT_TRAFFICJUNCTION_INFO)
EVENT_IVS_TRAFFICGATE	0x00000018	traffic gate event(Corresponding to DEV_EVENT_TRAFFICGATE_INFO)
EVENT_IVS_FACEDETECT	0x0000001A	target detection(Corresponding to DEV_EVENT_FACEDETECT_INFO)
EVENT_IVS_TRAFFICJAM	0x0000001B	traffic-Jam(Corresponding to DEV_EVENT_TRAFFICJAM_INFO)
EVENT_IVS_TRAFFIC_NONMOTORINMOTORROUTE	0x0000001C	Non-motor vehicles occupy the lanes(Corresponding to DEV_EVENT_TRAFFIC_NONMOTORINMOTORROUTE)

		RROUTE_INFO)
EVENT_IVS_TRAFFIC_RUNREDLIGHT	0x00000100	traffic-RunRedLight(Corresponding to DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO)
EVENT_IVS_TRAFFIC_OVERLINE	0x00000101	traffic-Overline(Corresponding to DEV_EVENT_TRAFFIC_OVERLINE_INFO)
EVENT_IVS_TRAFFIC_RETROGRADE	0x00000102	traffic-Retrograde(Corresponding to DEV_EVENT_TRAFFIC_RETROGRADE_INFO)
EVENT_IVS_TRAFFIC_TURNLEFT	0x00000103	traffic-TurnLeft(Corresponding to DEV_EVENT_TRAFFIC_TURNLEFT_INFO)
EVENT_IVS_TRAFFIC_TURNRIGHT	0x00000104	traffic-TurnRight(Corresponding to DEV_EVENT_TRAFFIC_TURNRIGHT_INFO)
EVENT_IVS_TRAFFIC_UTURN	0x00000105	traffic-Uturn(Corresponding to DEV_EVENT_TRAFFIC_UTURN_INFO)
EVENT_IVS_TRAFFIC_OVERSPEED	0x00000106	traffic-Overspeed(Corresponding to DEV_EVENT_TRAFFIC_OVERSPEED_INFO)
EVENT_IVS_TRAFFIC_UNDERSPEED	0x00000107	traffic-Underspeed(Corresponding to DEV_EVENT_TRAFFIC_UNDERSPEED_INFO)
EVENT_IVS_TRAFFIC_PARKING	0x00000108	traffic-Parking(Corresponding to DEV_EVENT_TRAFFIC_PARKING_INFO)
EVENT_IVS_TRAFFIC_WRONGROUTE	0x00000109	traffic-WrongRoute(Corresponding to DEV_EVENT_TRAFFIC_WRONGROUTE_INFO)
EVENT_IVS_TRAFFIC_CROSSLANE	0x0000010A	traffic-CrossLane(Corresponding to DEV_EVENT_TRAFFIC_CROSSLANE_INFO)
EVENT_IVS_TRAFFIC_OVERYELLOWLINE	0x0000010B	traffic-OverYellowLine(Corresponding to DEV_EVENT_TRAFFIC_OVERYELLOWLINE_I NFO)
EVENT_IVS_TRAFFIC_YELLOWPLATEINLAN E	0x0000010E	traffic-YellowPlateInLane(Corresponding to DEV_EVENT_TRAFFIC_YELLOWPLATEINLAN E_INFO)
EVENT_IVS_TRAFFIC_PEDESTRAINPRIORIT Y	0x0000010F	Traffic offense-Pedestral has higher priority at the crosswalk(Corresponding to DEV_EVENT_TRAFFIC_PEDESTRAINPRIORIT Y_INFO)
EVENT_IVS_TRAFFIC_NOPASSING	0x00000111	no passing(Corresponding to DEV_EVENT_TRAFFIC_NOPASSING_INFO)
EVENT_IVS_ABNORMALRUNDETECTION	0x00000112	abnormal run(Corresponding to DEV_EVENT_ABNORMALRUNDETECTION_I NFO)
EVENT_IVS_RETROGRADEDETECTION	0x00000113	retrograde(Corresponding to DEV_EVENT_RETROGRADEDETECTION_INF O)
EVENT_IVS_TAKENAWAYDETECTION	0x00000115	taking away things(Corresponding to DEV_EVENT_TAKENAWAYDETECTION_INFO)
EVENT_IVS_PARKINGDETECTION	0x00000116	parking(Corresponding to

		DEV_EVENT_PARKINGDETECTION_INFO)
EVENT_IVS_FACERECOGNITION	0x00000117	Target recognition(Corresponding to DEV_EVENT_FACERECOGNITION_INFO)
EVENT_IVS_TRAFFIC_MANUALSNAP	0x00000118	manual snap(Corresponding to DEV_EVENT_TRAFFIC_MANUALSNAP_INFO)
EVENT_IVS_TRAFFIC_FLOWSTATE	0x00000119	traffic flow state(Corresponding to DEV_EVENT_TRAFFIC_FLOW_STATE)
EVENT_IVS_TRAFFIC_VEHICLEINROUTE	0x0000011B	traffic vehicle route(Corresponding to DEV_EVENT_TRAFFIC_VEHICLEINROUTE_INFO)
EVENT_ALARM_MOTIONDETECT	0x0000011C	motion detect(Corresponding to DEV_EVENT_ALARM_INFO)
EVENT_ALARM_LOCALALARM	0x0000011D	local alarm(Corresponding to DEV_EVENT_ALARM_INFO)
EVENT_IVS_PSRISEDETECTION	0x0000011E	rise detect(Corresponding to DEV_EVENT_PSRISEDETECTION_INFO)
EVENT_IVS_TRAFFIC_TOLLGATE	0x00000120	traffic tollgate(Corresponding to DEV_EVENT_TRAFFICJUNCTION_INFO)
EVENT_IVS_DENSITYDETECTION	0x00000121	density detection of persons(Corresponding to DEV_EVENT_DENSITYDETECTION_INFO)
EVENT_IVS_VIDEODIAGNOSIS	0x00000122	video diagnosis result(Corresponding to NET_VIDEODIAGNOSIS_COMMON_INFO and NET_REAL_DIAGNOSIS_RESULT)
EVENT_IVS_TRAFFIC_VEHICLEINBUSROUTE	0x00000124	take up in bus route(Corresponding to DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INFO)
EVENT_IVS_TRAFFIC_BACKING	0x00000125	illegal astern(Corresponding to DEV_EVENT_IVS_TRAFFIC_BACKING_INFO)
EVENT_IVS_AUDIO_ABNORMALDETECTION	0x00000126	audio abnormity(Corresponding to DEV_EVENT_IVS_AUDIO_ABNORMALDETECTION_INFO)
EVENT_IVS_TRAFFIC_RUNYELLOWLIGHT	0x00000127	run yellow light(Corresponding to DEV_EVENT_TRAFFIC_RUNYELLOWLIGHT_INFO)
EVENT_IVS_CLIMBDETECTION	0x00000128	climb detection(Corresponding to DEV_EVENT_IVS_CLIMB_INFO)
EVENT_IVS_LEAVEDETECTION	0x00000129	leave detection(Corresponding to DEV_EVENT_IVS_LEAVE_INFO)
EVENT_IVS_TRAFFIC_PARKINGONYELLOW BOX	0x0000012A	parking on yellow box(Corresponding to DEV_EVENT_TRAFFIC_PARKINGONYELLOW_BOX_INFO)
EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING	0x0000012B	parking space parking(Corresponding to DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO)

EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING	0x0000012C	parking space no parking(Corresponding to DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO)
EVENT_IVS_TRAFFIC_PEDESTRAIN	0x0000012D	passerby(Corresponding to DEV_EVENT_TRAFFIC_PEDESTRAIN_INFO)
EVENT_IVS_TRAFFIC_THROW	0x0000012E	throw(Corresponding to DEV_EVENT_TRAFFIC_THROW_INFO)
EVENT_IVS_TRAFFIC_PARKINGSPACEOVERLINE	0x00000134	Parking line pressing events(Corresponding to DEV_EVENT_TRAFFIC_PARKINGSPACEOVERLINE_INFO)
EVENT_IVS_TRAFFIC_OVERSTOPLINE	0x00000137	Cross stop line event(Corresponding to DEV_EVENT_TRAFFIC_OVERSTOPLINE)
EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT	0x00000138	Traffic unfasten seat belt event(Corresponding to DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT)
EVENT_IVS_TRAFFIC_DRIVER_SMOKING	0x00000139	Driver smoking event(Corresponding to DEV_EVENT_TRAFFIC_DRIVER_SMOKING)
EVENT_IVS_TRAFFIC_DRIVER_CALLING	0x0000013A	Driver call event(Corresponding to DEV_EVENT_TRAFFIC_DRIVER_CALLING)
EVENT_IVS_TRAFFIC_PEDESTRAINRUNREDLIGHT	0x0000013B	Pedestrian red light(Corresponding to DEV_EVENT_TRAFFIC_PEDESTRAINRUNREDLIGHT_INFO)
EVENT_IVS_TRAFFIC_PASSNOTINORDER	0x0000013C	Pass not in order(corresponding DEV_EVENT_TRAFFIC_PASSNOTINORDER_INFO)
EVENT_ALARM_ANALOGALARM	0x00000150	Analog alarm channel alarm event(corresponding DEV_EVENT_ALARM_ANALOGALRM_INFO)
EVENT_IVS_CROSSLINEDETECTION_EX	0x00000151	Warning lineexpansion event(Corresponding to DEV_EVENT_CROSSLINE_INFO_EX)
EVENT_ALARM_VIDEOBLIND	0x00000153	Video tampering event(Corresponding to DEV_EVENT_ALARM_VIDEOBLIND)
EVENT_IVS_TRAFFIC_JAM_FORBID_INT0	0x00000163	Jam Forbid into event(corresponding DEV_EVENT_ALARM_JAMFORBIDINTO_INFO)
EVENT_IVS_SNAPBYTIME	0x00000164	Snap by time event(corresponding DEV_EVENT_SNAPBYTIME)
EVENT_IVS_QSYTRAFFICCARWEIGHT	0x00000168	Event of QSYTrafficCarWeight (corresponding to DEV_EVENT_QSYTRAFFICCARWEIGHT_INFO)
EVENT_IVS_SCENE_CHANGE	0x0000016D	Event of scene change (corresponding to DEV_ALRAM_SCENCHANGE_INFO,CFG_VIDEOABNORMALDETECTION_INFO)

EVENT_IVS_TUMBLE_DETECTION	0x00000177	Event of tumble detection(corresponding to DEV_EVENT_TUMBLE_DETECTION_INFO)
EVENT_IVS_ACCESS_CTL	0x00000204	Access control events (Corresponding to DEV_EVENT_ACCESS_CTL_INFO)
EVENT_IVS_SNAPMANUAL	0x00000205	SnapManual events(Corresponding to DEV_EVENT_SNAPMANUAL)
EVENT_IVS_TRAFFIC_ELETAGINFO	0x00000206	RFID electronic tag events(Corresponding to DEV_EVENT_TRAFFIC_ELETAGINFO_INFO)
EVENT_IVS_TRAFFIC_TIREDPHYSIOLOGICAL	0x00000207	physiological fatigue driving events(Corresponding to DEV_EVENT_TIREDPHYSIOLOGICAL_INFO)
EVENT_IVS_CITIZEN_PICTURE_COMPARE	0x00000209	Event of comparison with ID and card picture(corresponding to DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO)
EVENT_IVS_TRAFFIC_TIREDLOWERHEAD	0x0000020A	Event of driver lower head(Corresponding to DEV_EVENT_TIREDLOWERHEAD_INFO)
EVENT_IVS_TRAFFIC_DRIVERLOOKAROUND	0x0000020B	Event of driver look around(Corresponding to DEV_EVENT_DRIVERLOOKAROUND_INFO)
EVENT_IVS_TRAFFIC_DRIVERLEAVEPOST	0x0000020C	Event of driver leave post(Corresponding to DEV_EVENT_DRIVERLEAVEPOST_INFO)
EVENT_IVS_MAN_STAND_DETECTION	0x0000020D	stereo standing event(Corresponding to DEV_EVENT_MANSTAND_DETECTION_INFO)
EVENT_IVS_MAN_NUM_DETECTION	0x0000020E	Event of regional population statistics (Corresponding to DEV_EVENT_MANNUM_DETECTION_INFO)
EVENT_IVS_TRAFFIC_DRIVERYAWN	0x00000210	Event of driver yawn(Corresponding to DEV_EVENT_DRIVERYAWN_INFO)
EVENT_IVS_HUMANTRAIT	0x00000215	Event of human trait(Corresponding to DEV_EVENT_HUMANTRAIT_INFO)
EVENT_IVS_TRAFFIC_QUEUEJUMP	0x0000021C	Event of car jump a queue(Corresponding to DEV_EVENT_TRAFFIC_QUEUEJUMP_INFO)
EVENT_IVS_XRAY_DETECTION	0x00000223	X ray detection(Corresponding to DEV_EVENT_XRAY_DETECTION_INFO)
EVENT_IVS_TRAFFIC_TRUCKFORBID	0x00000229	truck forbid Event(Corresponding to DEV_EVENT_TRAFFICTRUCKFORBID_INFO)
EVENT_IVS_HIGHSPEED	0x0000022B	Event of high speed(Corresponding to DEV_EVENT_HIGHSPEED_INFO)
EVENT_IVS_CROWDDETECTION	0x0000022C	Event of crowd detection(Corresponding to DEV_EVENT_CROWD_DETECTION_INFO)
EVENT_IVS_TRAFFIC_CARDISTANCESHORT	0x0000022D	Event of car distance short(Corresponding to

		DEV_EVENT_TRAFFIC_CARDISTANCESHORT_INFO)
EVENT_IVS_PEDESTRIAN_JUNCTION	0x00000230	Pedestrian Junction Event(Corresponding to DEV_EVENT_PEDESTRIAN_JUNCTION_INFO)
EVENT_IVS_VEHICLE_RECOGNITION	0x00000231	Vehicle recognition alarm(Corresponding to DEV_EVENT_VEHICLE_RECOGNITION_INFO)
EVENT_IVS_BANNER_DETECTION	0x0000023B	Event of Banner detection(Corresponding to DEV_EVENT_BANNER_DETECTION_INFO)
EVENT_IVS_NORMAL_FIGHTDETECTION	0x0000023C	Event of normal fight(only be used to rule of normal fight, the alarm event is same as EVENT_IVS_FIGHTDETECTION)
EVENT_IVS_ELEVATOR_ABNORMAL	0x0000023D	Event of elevator abnormal(Corresponding to DEV_EVENT_ELEVATOR_ABNORMAL_INFO)
EVENT_IVS_TRAFFIC_PARKING_B	0x00000240	Event of Class B traffic-Parking(Corresponding to DEV_EVENT_TRAFFIC_PARKING_B_INFO)
EVENT_IVS_TRAFFIC_PARKING_C	0x00000241	Event of Class C traffic-Parking(Corresponding to DEV_EVENT_TRAFFIC_PARKING_C_INFO)
EVENT_IVS_TRAFFIC_PARKING_D	0x00000242	Event of Class D traffic-Parking(Corresponding to DEV_EVENT_TRAFFIC_PARKING_D_INFO)
EVENT_IVS_FIREWARNING	0x00000245	Event of FireWarning(Corresponding to DEV_EVENT_FIREWARNING_INFO)
EVENT_IVS_SHOPPRESENCE	0x00000246	Event of ShopPresence(Corresponding to DEV_EVENT_SHOPPRESENCE_INFO)
EVENT_IVS_DISTANCE_DETECTION	0x0000024A	Event of distance detection(Corresponding to DEV_EVENT_DISTANCE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_OVERLOAD	0x0000024B	Event of non-motor overload (Corresponding to DEV_EVENT_TRAFFIC_NONMOTOR_OVERLOAD_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_WITHOUTSAFEHAT	0x0000024C	Event of non-motor without safehat (Corresponding to DEV_EVENT_TRAFFIC_NONMOTOR_WITHOUTSAFEHAT_INFO)
EVENT_IVS_FLOWBUSINESS	0x0000024E	Event of flowBusiness (Corresponding to DEV_EVENT_FLOWBUSINESS_INFO)
EVENT_IVS_CITY_MOTORPARKING	0x0000024F	Event of CityMotorParking (Corresponding to DEV_EVENT_CITY_MOTORPARKING_INFO)

EVENT_IVS_CITY_NONMOTORPARKING	0x00000250	Event of CityNonMotorParking (Corresponding to EV_EVENT_CITY_NONMOTORPARKING_INFO)
EVENT_IVS_LANEDEPARTURE_WARNNING	0x00000251	Lane Departure warnning(Corresponding to DEV_EVENT_LANEDEPARTURE_WARNNING_INFO)
EVENT_IVS_FORWARDCOLLISION_WARNNING	0x00000252	Forward Collision Warnning(Corresponding to DEV_EVENT_FORWARDCOLLISION_WARNNING_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_HOLDUMBRELLA	0x00000254	Event of NonMotor hold umbrella (Corresponding to DEV_EVENT_TRAFFIC_NONMOTOR_HOLDUMBRELLA_INFO)
EVENT_IVS_FLOATINGOBJECT_DETECTION	0x00000257	Event of FloatingObject detection (Corresponding to DEV_EVENT_FLOATINGOBJECT_DETECTION_INFO)
EVENT_IVS_PHONECALL_DETECT	0x0000025A	Event of phone call detect(Corresponding to DEV_EVENT_PHONECALL_DETECT_INFO)
EVENT_IVS_SMOKING_DETECT	0x0000025B	Event of Smoking Detection(Corresponding to DEV_EVENT_SMOKING_DETECT_INFO)
EVENT_IVS_RADAR_SPEED_LIMIT_ALARM	0x0000025C	Event of Radar speed limit alarm(Corresponding to DEV_EVENT_RADAR_SPEED_LIMIT_ALARM_INFO)
EVENT_IVS_WATER_LEVEL_DETECTION	0x0000025D	Event of Water level detection (Corresponding to DEV_EVENT_WATER_LEVEL_DETECTION_INFO)
EVENT_IVS_HOLD_UMBRELLA	0x0000025E	Event of Hold umbrella detection(Corresponding to DEV_EVENT_HOLD_UMBRELLA_INFO)
EVENT_IVS_GARBAGE_EXPOSURE	0x0000025F	Event of Garbage Exposure detection (Corresponding to DEV_EVENT_GARBAGE_EXPOSURE_INFO)
EVENT_IVS_DUSTBIN_OVER_FLOW	0x00000260	Event of Dustbin Overflow detection (Corresponding to DEV_EVENT_DUSTBIN_OVER_FLOW_INFO)
EVENT_IVS_DOOR_FRONT_DIRTY	0x00000261	Event of Door Front Dirty detection(Corresponding to DEV_EVENT_DOOR_FRONT_DIRTY_INFO)
EVENT_IVS_TRAFFIC_PARKING_MANUAL	0x00000265	Event of manual traffic-Parking(Corresponding to DEV_EVENT_TRAFFIC_PARKING_MANUAL_I

		NFO)
EVENT_IVS_HELMET_DETECTION	0x00000266	Event of helmet detection (Corresponding to DEV_EVENT_HELMET_DETECTION_INFO)
EVENT_IVS_HOTSPOT_WARNING	0x00000268	Event of Hot spot warning(Corresponding to DEV_EVENT_HOTSPOT_WARNING_INFO)
EVENT_IVS_WEIGHING_PLATFORM_DETECTION	0x00000269	Event of weighing platform detection(Corresponding to DEV_EVENT_WEIGHING_PLATFORM_DETECTION_INFO)
EVENT_IVS_CLASSROOM_BEHAVIOR	0x0000026A	Event of classroom behavior detection(Corresponding to DEV_EVENT_CLASSROOM_BEHAVIOR_INFO)
EVENT_IVS_VEHICLE_DISTANCE_NEAR	0x0000026B	Event of safe driving vehicle distance near alarm(Corresponding to DEV_EVENT_VEHICLE_DISTANCE_NEAR_INFO)
EVENT_IVS_TRAFFIC_DRIVER_ABNORMAL	0x0000026C	Event of traffic driver abnormal alarm(Corresponding to DEV_EVENT_TRAFFIC_DRIVER_ABNORMAL_INFO)
EVENT_IVS_WORKCLOTHES_DETECT	0x0000026E	Event of work clothes(helmet/clothes)detection(Corresponding to DEV_EVENT_WORKCLOTHES_DETECT_INFO)
EVENT_IVS_SECURITYGATE_PERSONALARM	0x0000026F	Event of security gate person alarm(Corresponding to DEV_EVENT_SECURITYGATE_PERSONALARM_INFO)
EVENT_IVS_STAY_ALONE_DETECTION	0x00000270	Event of stay alone detection (Corresponding to DEV_EVENT_STAY_ALONE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_ROAD_BLOCK	0x00000271	Event of traffic road block(Corresponding to DEV_EVENT_TRAFFIC_ROAD_BLOCK_INFO)
EVENT_IVS_TRAFFIC_ROAD_CONSTRUCTION	0x00000272	Event of traffic road construction(Corresponding to DEV_EVENT_TRAFFIC_ROAD_CONSTRUCTION_INFO)
EVENT_IVS_WORKSTATDETECTION	0x00000274	Event of work stat detection(Corresponding to DEV_EVENT_WORKSTATDETECTION_INFO)
EVENT_IVS_FEATURE_ABSTRACT	0x00000276	Event of feature abstract(Corresponding to DEV_EVENT_FEATURE_ABSTRACT_INFO)

EVENT_IVS_INTELLI_SHELF	0x00000277	Event of intelligent replenishment(Corresponding to DEV_EVENT_INTELLI_SHELF_INFO)
EVENT_ALARM_SMARTMOTION_HUMAN	0x00000279	Event of smart motion detection(human),(Corresponding to DEV_EVENT_SMARTMOTION_HUMAN_INFO)
EVENT_ALARM_SMARTMOTION_VEHICLE	0x0000027A	Event of smart motion detection(vehicle),(Corresponding to DEV_EVENT_SMARTMOTION_VEHICLE_INFO)
EVENT_IVS_CAR_DRIVING_IN_OUT	0x0000027B	Event of car driving in or out(Corresponding to DEV_EVENT_CAR_DRIVING_IN_OUT_INFO)
EVENT_IVS_VIOLENT_THROW_DETECTION	0x0000027D	Event violent throw detection (Corresponding to DEV_EVENT_VIOLENT_THROW_DETECTION_INFO)
EVENT_IVS_FACEBODY_DETECT	0x00000281	Event of target body detect (Corresponding to DEV_EVENT_FACEBODY_DETECT_INFO)
EVENT_IVS_FACEBODY_ANALYSE	0x00000282	Event of target body analyse (Corresponding to DEV_EVENT_FACEBODY_ANALYSE_INFO)
EVENT_IVS_GASSTATION_VEHICLE_DETECT	0x00000283	Event of gas station vehicle detection(Corresponding to DEV_EVENT_GASSTATION_VEHICLE_DETECT_INFO)
EVENT_IVS_ANIMAL_DETECTION	0x00000286	Event of animal detection(Corresponding to DEV_EVENT_ANIMAL_DETECTION_INFO)
EVENT_IVS_SHOP_WINDOW_POST	0x00000287	Event of shop window post(Corresponding to DEV_EVENT_SHOP_WINDOW_POST_INFO)
EVENT_IVS_SHOP_SIGN_ABNORMAL	0x00000288	Event of shop sign abnormal (Corresponding to DEV_EVENT_SHOP_SIGN_ABNORMAL_INFO)
EVENT_IVS_BREED_DETECTION	0x00000289	Event of breed detection (Corresponding to DEV_EVENT_BREED_DETECTION_INFO)
EVENT_IVS_HIGH_TOSS_DETECT	0x0000028D	Event of high toss detection(Corresponding to DEV_EVENT_HIGH_TOSS_DETECT_INFO)
EVENT_IVS_RADAR_REGION_DETECTION	0x00000295	Event of Radar cross region detection(Corresponding to DEV_EVENT_RADAR_REGION_DETECTION_INFO)

EVENT_IVS_PARKING_LOT_STATUS_DETECTION	0x00000297	Event of parking lot status detection (Corresponding to DEV_EVENT_PARKING_LOT_STATUS_DETECTION_INFO)
EVENT_IVS_DREGS_UNCOVERED	0x00000299	Event of loading test not covered by muck truck(Corresponding to DEV_EVENT_DREGS_UNCOVERED_INFO)
EVENT_IVS_SMART_KITCHEN_CLOTHES_DETECTION	0x0000029D	Event of Smart kitchen wearing detection(alarm for not wearing mask, chef's clothes whose color does not meet the requirements)(Corresponding to DEV_EVENT_SMART_KITCHEN_CLOTHES_DETECTION_INFO)
EVENT_IVS_SLEEP_DETECT	0x0000029E	Event of sleep detect(Corresponding to DEV_EVENT_SLEEP_DETECT_INFO)
EVENT_IVS_PLAY_MOBILEPHONE	0x00000300	Event of play mobile phone(Corresponding to DEV_EVENT_PLAY_MOBILEPHONE_INFO)
EVENT_IVS_ANATOMY_TEMP_DETECT	0x00000303	Event of Intelligent detection of human body temperature(Corresponding to DEV_EVENT_ANATOMY_TEMP_DETECT_INFO)
EVENT_IVS_FOG_DETECTION	0x00000308	Event of fog detect(Corresponding to DEV_EVENT_FOG_DETECTION)
EVENT_IVS_TRAFFIC_VEHICLE_BC	0x00000309	Event of bc (Corresponding to DEV_EVENT_TRAFFIC_VEHICLE_BC)
EVENT_IVS_NONMOTOR_ENTRYING	0x0000030C	Event of non-motor vehicle access elevator(Corresponding to DEV_EVENT_NONMOTOR_ENTRYING_INFO)
EVENT_IVS_WATER_STAGE_MONITOR	0x0000030D	Event of water stage monitor, only used for task intelligent analysis (Corresponding to DEV_EVENT_WATER_STAGE_MONITOR_INFO)
EVENT_IVS_TRAFFIC_ROAD_ALERT	0x0000030E	Event of traffic traffic road alert(Corresponding to DEV_EVENT_TRAFFIC_ROAD_ALERT_INFO)
EVENT_IVS_BREAK_RULE_BUILDING_DETECTION	0x0000030F	Event of break rule building detection(Corresponding to DEV_EVENT_BREAK_RULE_BUILDING_DETECTION_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_RUN_RED_LIGHT	0x00000310	Event of traffic non-motor run redlight (Corresponding to DEV_EVENT_TRAFFIC_NONMOTOR_RUN_RED_LIGHT_INFO)
EVENT_IVS_TRAFFIC_VEHICLE_IN_EMERGENCY_LANE	0x00000311	Event of vehicle in emergency lane (Corresponding to DEV_EVENT_TRAFFIC_VEHICLE_IN_EMERGENCY_LANE)

		NCY_LANE_INFO)
EVENT_IVS_TRAFFIC_CAR_MEASUREMENT	0x00000320	Event of Traffic checkpoint measurement (vehicle length, width, height, weight, etc.) (Corresponding to DEV_EVENT_TRAFFIC_CAR_MEASUREMENT_INFO)
EVENT_IVS_TRAFFIC_REAREND_ACCIDENT	0x00000322	Event of Traffic rearend accident (Corresponding to DEV_EVENT_TRAFFIC_REAREND_ACCIDENT_INFO)
EVENT_IVS_FIRE_LANE_DETECTION	0x00000324	Event of Fire lane detection (Corresponding to DEV_EVENT_FIRE_LANE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_NON_MOTOR_RETROGRADE	0x00000328	Event of traffic non motor retrograde (Corresponding to DEV_EVENT_TRAFFIC_NON_MOTOR_RETROGRADE_INFO)
EVENT_IVS_CAR_DRIVING_IN	0x00000330	Event of Cardriving in (Corresponding to DEV_EVENT_CAR_DRIVING_IN_INFO)
EVENT_IVS_CAR_DRIVING_OUT	0x00000331	Event of Car driving out (Corresponding to DEV_EVENT_CAR_DRIVING_OUT_INFO)
EVENT_IVS_TRAFFIC_SPECIAL_VEHICLE_DETECT	0x00000333	Event of special vehicle detect (For rule configuration only)
EVENT_IVS_TRAFFIC_NONMOTOR	0x00000335	Event of traffic nonmotor detector (Corresponding to DEV_EVENT_TRAFFIC_NONMOTOR_INFO)
EVENT_IVS_TRAFFIC_VISIBILITY	0x00000337	Event of traffic visibility detection (Corresponding to DEV_EVENT_TRAFFIC_VISIBILITY_INFO)
EVENT_IVS_TRAFFIC_VEHICLE_CLEANLINESS	0x00000338	Event of traffic vehicle cleanliness detection (Corresponding to DEV_EVENT_TRAFFIC_VEHICLE_CLEANLINESS_INFO)
EVENT_IVS_TRUCKNOTCLEAN_FOR_PRMA	0x0000033A	Event of Trunk no clean Corresponding to DEV_EVENT_TRUCKNOTCLEAN_FOR_PRMA_INFO
EVENT_IVS_ANYTHING_DETECT	0x0000033F	Event of Anything Detection (Corresponding to DEV_EVENT_ANYTHING_DETECT_INFO)
EVENT_IVS_OBJECT_ABNORMAL	0x00000340	Event of Object abnormal (Corresponding to DEV_EVENT_OBJECT_ABNORMAL_INFO)
EVENT_IVS_ARTICLE_DETECTION	0x00000345	Article Detection for rule configuration, corresponding to EVENT_IVS_LEFTDETECTION or EVENT_IVS_TAKENAWAYDETECTION

EVENT_IVS_TRAFFIC_PARKINGSPACE_MANUALSNAP	0x00000346	Manual capture of roadside parking spaces (Corresponding to DEV_EVENT_PARKINGSPACE_MANUALSNAP_INFO)
EVENT_IVS_STREET_SUNCURE	0x00000347	Sun drying incident along the street (Corresponding to DEV_EVENT_STREET_SUNCURE_INFO)
EVENT_IVS_OUTDOOR_ADVERTISEMENT	0x00000348	Outdoor advertising event (Corresponding to DEV_EVENT_OUTDOOR_ADVERTISEMENT_INFO)
EVENT_IVS_HUDDLE_MATERIAL	0x00000349	Random material detection event (Corresponding to DEV_EVENT_HUDDLE_MATERIAL_INFO)
EVENT_IVS_FOLLOW_CAR_ALARM	0x0000034F	Car following alarm (Corresponding to DEV_EVENT_FOLLOW_CAR_ALARM_INFO)
EVENT_IVS_TRAFFIC_PARKING_STATISTICS	0x0000035B	event to Parking statistics (Corresponding to DEV_EVENT_TRAFFIC_PARKING_STATISTICS_INFO)
EVENT_IVS_HEAT_IMAGING_TEMPER	0x0000035C	Thermal temperature abnormal event alarm (Corresponding to DEV_EVENT_HEAT_IMAGING_TEMPER_INFO)
EVENT_IVS_SEWAGE_DETECTION	0x00000362	Pollution detection event (Corresponding to DEV_EVENT_SEWAGE_DETECTION_INFO)
EVENT_IVS_WATERCOLOR_DETECTION	0x00000363	Water color event (Corresponding to DEV_EVENT_WATERCOLOR_DETECTION_INFO)
EVENT_IVS_VIDEO_NORMAL_DETECTION	0x00000365	Video normal events, At the end of the video diagnostic detection cycle, the diagnostic items that have not reported errors are reported to the normal events DEV_EVENT_VIDEO_NORMAL_DETECTION_INFO
EVENT_IVS_OBJECT_PLACEMENT_DETECTION	0x00000369	Item placement detection event (Corresponding to DEV_EVENT_OBJECT_PLACEMENT_DETECTION_INFO)
EVENT_IVS_OBJECT_REMOVAL_DETECTION	0x0000036A	Item removal detection event (Corresponding to DEV_EVENT_OBJECT_REMOVAL_DETECTION_INFO)
EVENT_IVS_FIRE_EXTINGUISHER_DETECTION	0x0000036C	Fire extinguisher detection event (Corresponding to DEV_EVENT_FIRE_EXTINGUISHER_DETECTION_INFO)

EVENT_IVS_DIALRECOGNITION	0x00000371	Dial recognition alarm(Corresponding to DEV_EVENT_DIALRECOGNITION_INFO)
EVENT_IVS_ELECTRICFAULT_DETECT	0x00000372	Electric defect detection alarm(Corresponding to DEV_EVENT_ELECTRICFAULTDETECT_INFO)
EVENT_IVS_TRASH_WITHOUT_COVER_DETECTION	0x00000373	Detection event of uncovered trash can(Corresponding to DEV_EVENT_TRASH_WITHOUT_COVER_DETECTION_INFO)
EVENT_IVS_TRAFFIC_PARKING_BACKING	0x0000037C	PARKING BACKING Event (Corresponding to DEV_EVENT_TRAFFIC_PARKING_BACKING_INFO)#define EVENT_IVS_VALVE_ABNORMAL0x0000037D
EVENT_IVS_BARELAND_DETECTION	0x00000380	Bare soil detection event(DEV_EVENT_BARELAND_DETECTION_INFO)
EVENT_IVS_CONSUMPTION_EVENT	0x00000381	Consumption event (Corresponding to DEV_EVENT_CONSUMPTION_EVENT_INFO)
EVENT_IVS_XRAY_UNPACKING_CHECK	0x00000384	XRAY Unpacking Check(corresponding to DEV_EVENT_XRAY_UNPACKING_CHECK_INFO)
EVENT_IVS_TRAFFIC_CHANGE_LANE_CONTINUES	0x00000387	Traffic Change Lane Continues Event(corresponding to DEV_EVENT_TRAFFIC_CHANGE_LANE_CONTINUES_INFO)
EVENT_IVS_GENEAL_ATTITUDE	0x0000038C	Geneal Attitude Event(Corresponding to DEV_EVENT_GENEAL_ATTITUDE_INFO)
EVENT_IVS_FISHING_DETECTION	0x00000390	Fishing Detection Event(corresponding to DEV_EVENT_FISHING_DETECTION_INFO)
EVENT_IVS_CROWD_LEVEL_DETECTION	0x00000395	Congestion detection event(Corresponding to DEV_EVENT_CROWD_LEVEL_DETECTION_INFO)
EVENT_IVS_ROAD_CONDITIONS_DETECTION	0x0000039A	Road detection event(Corresponding to DEV_EVENT_ROAD_CONDITIONS_DETECTION_INFO)
EVENT_IVS_OPEN_INTELLI	0x0000039D	Open Intelligent Event(Corresponding to DEV_EVENT_OPEN_INTELLI_INFO)
EVENT_IVS_RIDING_MOTOR_CYCLE	0x00000401	Motorcycle straddle detection event(Corresponding to DEV_EVENT_RIDING_MOTOR_CYCLE_INFO)
EVENT_IVS_TRAFFIC_SPEED_DROP_SHARPLY	0x00000404	Traffic Speed Drop Sharply event(Corresponding to DEV_EVENT_TRAFFIC_SPEED_DROP_SHARPLY_INFO)

EVENT_IVS_TRAFFIC_OVERTAKE_ONRIGHT	0x0000040A	Right overtaking event(Corresponding to DEV_EVENT_TRAFFIC_OVERTAKE_ONRIGHT_INFO)
EVENT_IVS_TRAFFIC_TRUCK_OCCUPIED	0x0000040B	Truck occupation event(Corresponding to DEV_EVENT_TRAFFIC_TRUCK_OCCUPIED_INFO)
EVENT_IVS_DROP_DETECTION	0x00000429	Drip detection event(Corresponding to NET_DEV_EVENT_DROP_DETECTION_INFO)
EVENT_IVS_TEMPERATURE_ALARM	0x0000042A	Temperature Alarm(Corresponding to NET_DEV_EVENT_TEMPERATURE_ALARM_INFO)
EVENT_IVS_HUMIDITY_ALARM	0x0000042B	Humidity Alarm(Corresponding to NET_DEV_EVENT_HUMIDITY_ALARM_INFO)
EVENT_IVS_ILLEGAL_CARRIAGE	0x0000042F	Illegal Carriage event(Corresponding to NET_DEV_EVENT_ILLEGAL_CARRIAGE_INFO)
EVENT_IVS_REMOTE_APPROVAL_ALARM	0x00000438	Financial remote approval event(Corresponding to NET_DEV_EVENT_REMOTE_APPROVAL_ALARM_INFO)
EVENT_IVS_ANTI_COUNTERFEIT	0x00000439	Anti-counterfeiting detection event(Corresponding to NET_DEV_EVENT_ANTI_COUNTERFEIT_INFO)
EVENT_IVS_USERMANAGER_FOR_TWSDK	0x00000441	User information reporting event(corresponding to NET_DEV_EVENT_USERMANAGER_FOR_TWSDK_INFO)
EVENT_IVS_POSITION_SNAP	0x00000447	Snap with position(corresponding to NET_DEV_EVENT_POSITION_SNAP_INFO)
EVENT_IVS_CIGARETTE_CASE_DETECTION	0x00000450	Cigarette case detection event(corresponding to NET_DEV_EVENT_CIGARETTE_CASE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_ACCELERATION_RAPID	0x00000457	Traffic Acceleration Rapid Event(corresponding to NET_DEV_EVENT_TRAFFIC_ACCELERATION_RAPID_INFO)
EVENT_IVS_TRAFFIC_TURN_SHARP	0x00000458	Traffic Turn Sharp Event(corresponding to NET_DEV_EVENT_TRAFFIC_TURN_SHARP_INFO)
EVENT_IVS_GARBAGE_PLASTICBAG	0x00000459	Garbage Plasticbag Event(corresponding to NET_DEV_EVENT_GARBAGE_PLASTICBAG_INFO)
EVENT_IVS_COLLISION_CONFLICT	0x0000045B	Collision Conflict Event(corresponding to NET_DEV_EVENT_COLLISION_CONFLICT_IN

		FO)
EVENT_IVS_OBJECT_APPEAR_DETECTION	0x0000045F	Object Appear Detection event(corresponding to NET_DEV_EVENT_OBJECT_APPEAR_DETECTION_INFO)
EVENT_IVS_OBJECT_DISAPPEAR_DETECTION	0x00000460	Object Disappear Detection event(corresponding to NET_DEV_EVENT_OBJECT_DISAPPEAR_DETECTION_INFO)
EVENT_IVS_OBJECT_STATE_DETECTION	0x00000461	Object state event(corresponding to NET_DEV_EVENT_OBJECT_STATE_DETECTION_INFO)
EVENT_IVS_ACTION_COUNT	0x0000046E	Action Count event(corresponding to NET_DEV_EVENT_ACTION_COUNT_INFO)
EVENT_IVS_WADING_DETECTION	0x0000046F	Wading safety detection and water area monitoring alarm (corresponding to NET_DEV_EVENT_WADING_DETECTION_INFO)
EVENT_IVS_SAME_OBJECT_SEARCH_DETECT	0x00000472	Trace object detection events (corresponding to NET_DEV_EVENT_SAME_OBJECT_SEARCH_DETECT_INFO)
EVENT_IVS_SAME_OBJECT_SEARCH_COUNT	0x00000480	Event for counting items based on the chart (corresponding to NET-DEV-EVENT_SAME_OBJECT_SEARCH_COUNT_INFO)