NetSDK_JAVA
编程指导手册（智能 AI 分册）

V2.0.1

前言

目的

欢迎使用 NetSDK（以下简称 SDK）编程指导手册。

SDK 是软件开发者在开发网络硬盘录像机、网络视频服务器、网络摄像机、网络球机和智能设备等产品监控联网应用时的开发套件。

本文档描述了 IVSS、NVR、IPC、ITC、客流量统计设备、闸机等设备的智能业务涉及的 SDK 接口以及调用流程，基础核心业务流程（例如初始化、登录、普通报警和智能报警等）详见《NetSDK 编程指导手册》。

本文档提供的示例代码仅为演示接口调用方法，不保证能直接拷贝编译。

读者对象

使用 SDK 的软件开发工程师、产品经理和项目经理。

符号约定

在本文档中可能出现下列标识，代表的含义如下。

标识	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员伤亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 防静电	表示静电敏感的设备。
 当心触电	表示高压危险。
 激光辐射	表示强激光辐射。
 风扇警告	表示危险运动部件，请远离运动风扇叶片。
 当心机械伤人	表示设备部件机械伤人。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

修订记录

版本号	修订内容	发布日期
V2.0.1	新增工装检测事件	2025.07
V2.0.0	前言补充，新增智能事件枚举	2025.02
V1.0.3	新增附录 3 智能事件。	2023.12
V1.0.2	全文优化语言。	2023.02
V1.0.1	更新依赖库信息。	2021.04
V1.0.0	首次发布。	2020.10

名词解释

以下对本文档中使用的专业名词分别说明，帮助您更好的理解各个业务功能。

名词	解释
目标检测	通过对视频进行智能分析，检测出其中的目标及目标的特征信息（年龄、性别，表情等）。
目标识别	包含目标检测，通过对视频进行智能分析，根据布控的人脸库，检测出视频中的人脸是否在布控的人脸库中。
人脸库	预先导入一些人脸图片到 IVSS、NVR、前端 IPC 等设备中，通过检测人脸是否在其中，确认抓拍到的人的身份。
ITC	Intelligent Traffic Camera，智能交通摄像机，具有抓拍车辆图片并自动分析交通事件的功能。
绊线入侵检测	自动检测穿越警戒线的行为。
区域入侵检测	自动检测目标入侵警戒区的行为，包括“穿越区域”和“在区域内”。
客流量	相机标定区域内的人数信息。

目录

第 1 章 产品概述

1.1 概述

本文档主要介绍 SDK 接口参考信息，包括主要功能、接口函数和回调函数。

主要功能包括通用功能、目标检测和目标识别、通用行为事件、人体检测、热成像温度检测、门禁事件、客流量统计、智能交通和人证对比。根据环境不同，开发包包含的文件会不同，具体如下所示。

- Windows 开发包所包含的文件，请参见表 1-1。

表1-1 Windows 开发包包括的文件

库类型	库文件名称	库文件说明
功能库	dhnetsdk.h	头文件
	dhnetsdk.dll	库文件
	avnetsdk.dll	库文件
	dhconfigsdk.h	头文件
	dhconfigsdk.dll	库文件
播放（编码解码）辅助库	dhplay.dll	播放库
	StreamConvertor.dll	转码库

- Linux 开发包所包含的文件，请参见表 1-2。

表1-2 Linux 开发包包括的文件

库类型	库文件名称	库文件说明
功能库	dhnetsdk.h	头文件
	libdhnetsdk.so	库文件
	libavnetsdk.so	库文件
配置库	avglobal.h	头文件
	dhconfigsdk.h	头文件
	libdhconfigsdk.so	库文件

说明

- SDK 的功能库和配置库是必备库。
- 功能库是设备网络 SDK 的主体，主要用于网络客户端与各类产品之间的通讯交互，负责远程控制、查询、配置及码流数据的获取和处理等。
- 配置库针对配置功能的结构体进行打包和解析。
- 推荐使用播放库进行码流解析和播放。

1.2 适用性

- 推荐内存：不低于 512M。
- Jdk 使用版本：jdk1.6；jdk1.8。
- SDK 支持的系统：
  - ◇ Windows

    Windows 10/Windows 8.1/Windows 7/2000 以及 Windows Server 2008/2003。

◇ Linux

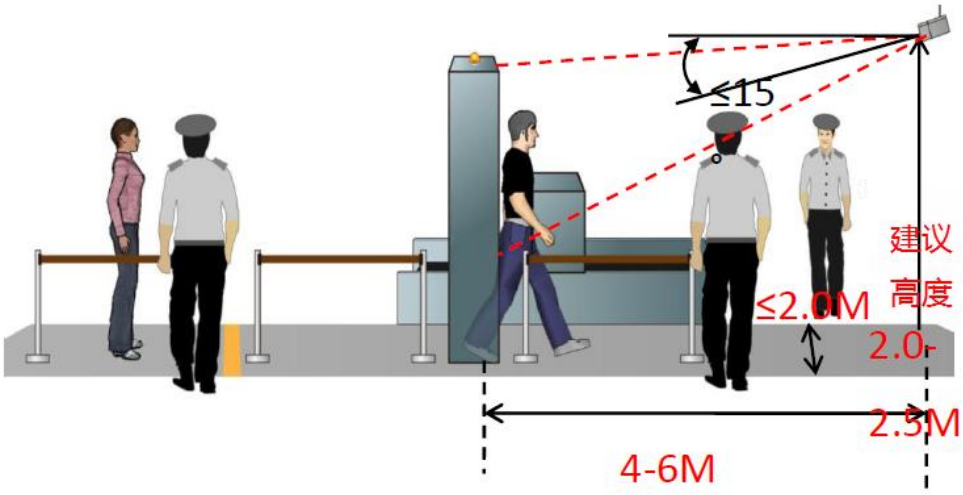
Red Hat/SUSE 等通用 Linux 系统。

1.3 应用场景

1.3.1 目标检测/目标识别/人体检测

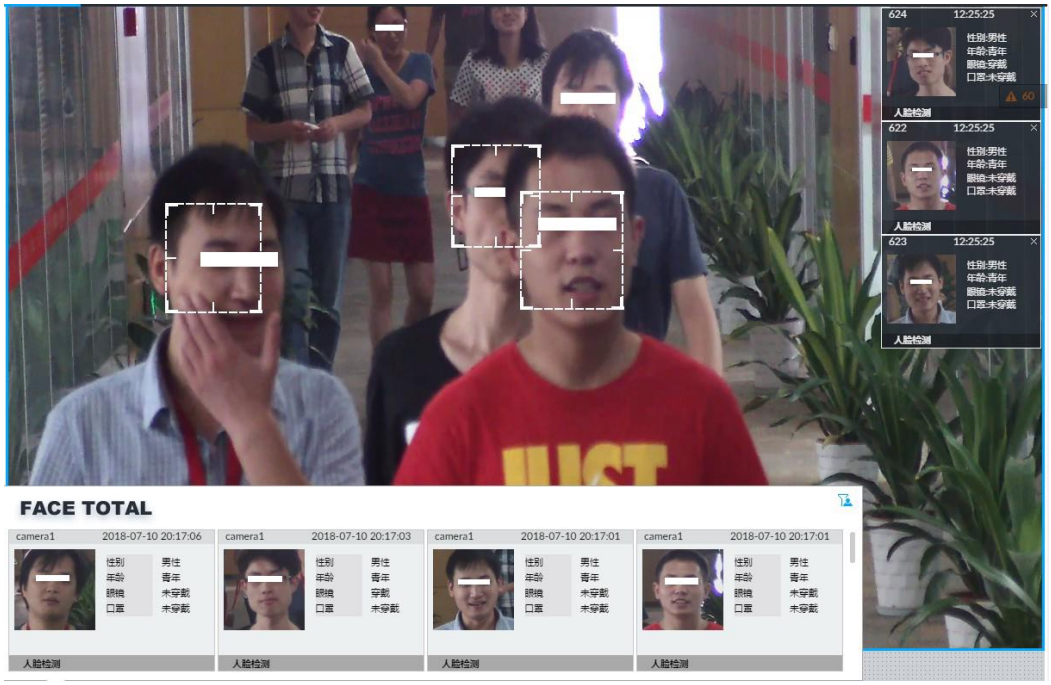
目标检测、目标识别和人体识别设备的使用场景如**错误!未找到引用源。**所示。

图1-1 目标识别



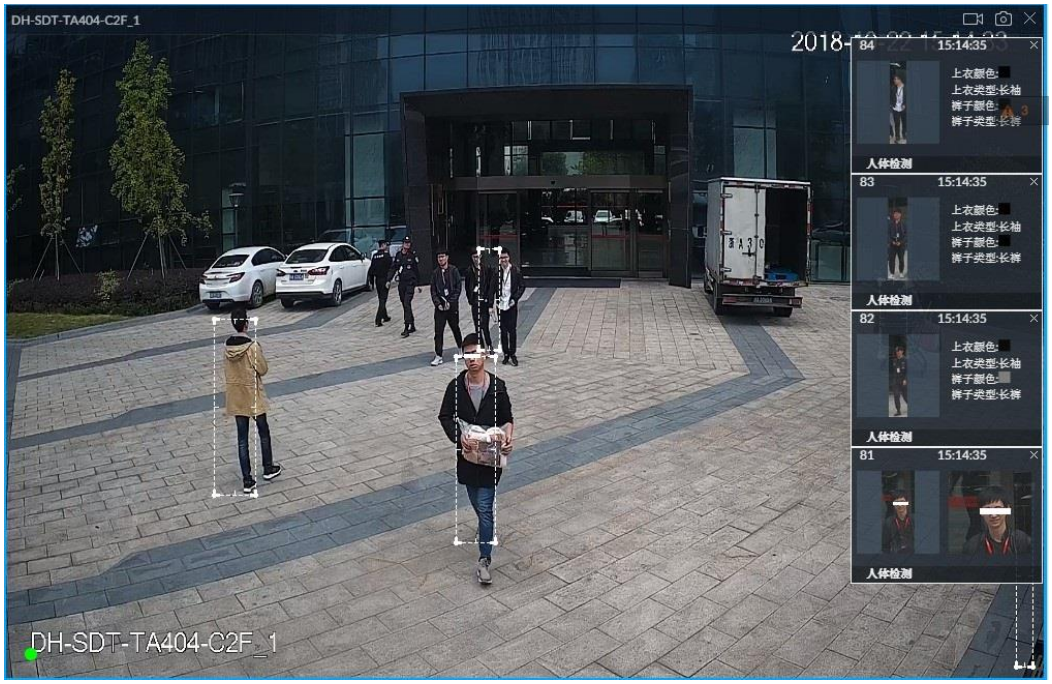
目标检测场景，如图 1-2 所示。

图1-2 目标检测场景



人体检测场景，如图 1-3 所示。

图1-3 人体检测场景



1.3.2 客流量统计

客流产品在实际场景的应用，如图 1-4 所示。

图1-4 客流场景

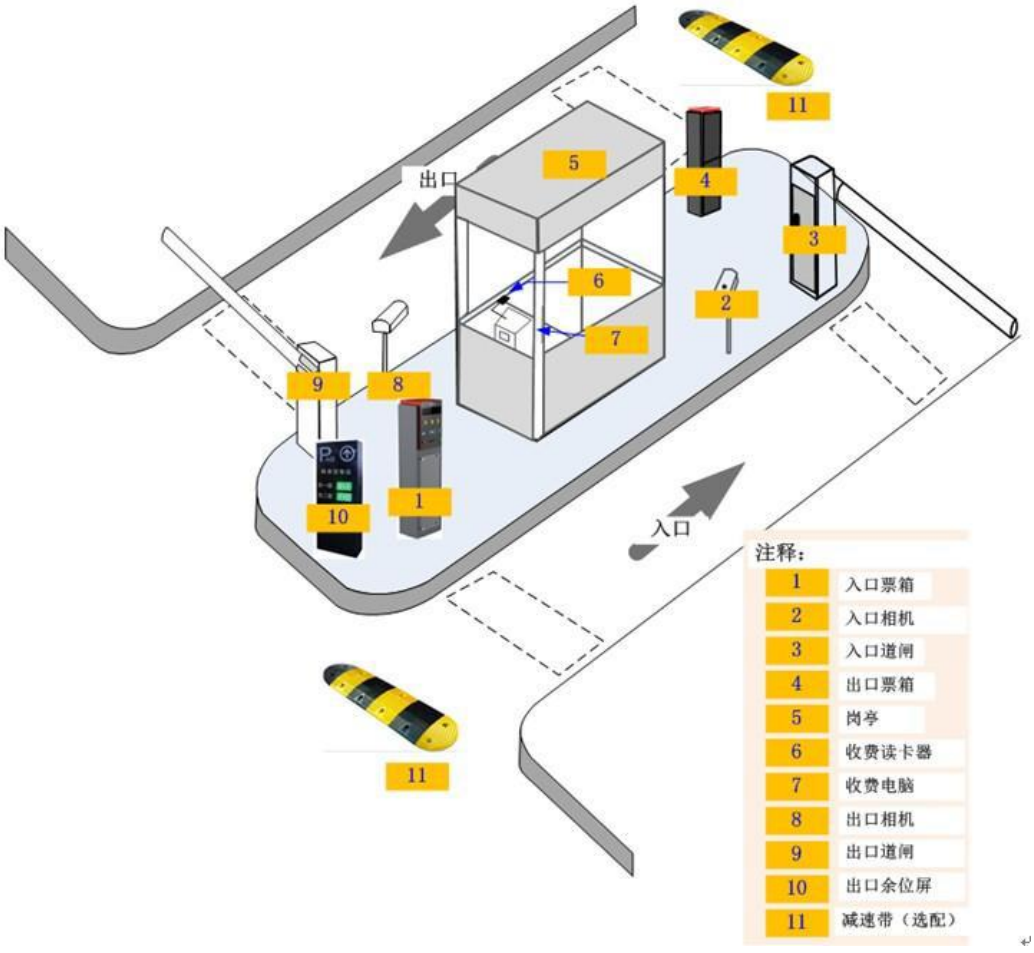
1.3.3 智能交通

- ITC 应用于交通路口，用于抓拍交通违章行为及车辆流量统计，如图 1-5 所示。

图1-5 ITC 在交通路口的应用

- ITC 应用于停车场出入口，用于控制车辆进出停车场及监控车位是否有空余，如图 1-6 所示。

图1-6 ITC 在停车场出入口的应用



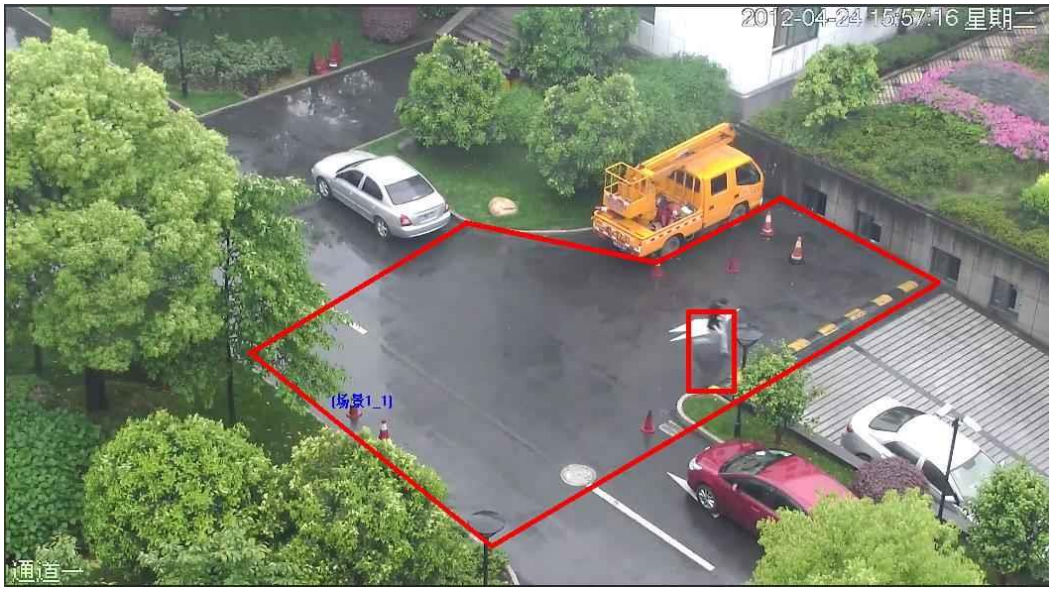
1.3.4 通用行为

指人或者车辆穿越规则线（绊线入侵）或者入侵警戒区（区域入侵）产生相应的报警事件，同时可以区分出目标物体（人或者车辆）。

通用行为应用场景如图 1-7、图 1-8 所示。

图1-7 通用行为场景-绊线入侵

图1-8 通用行为场景-区域入侵



1.3.5 闸机

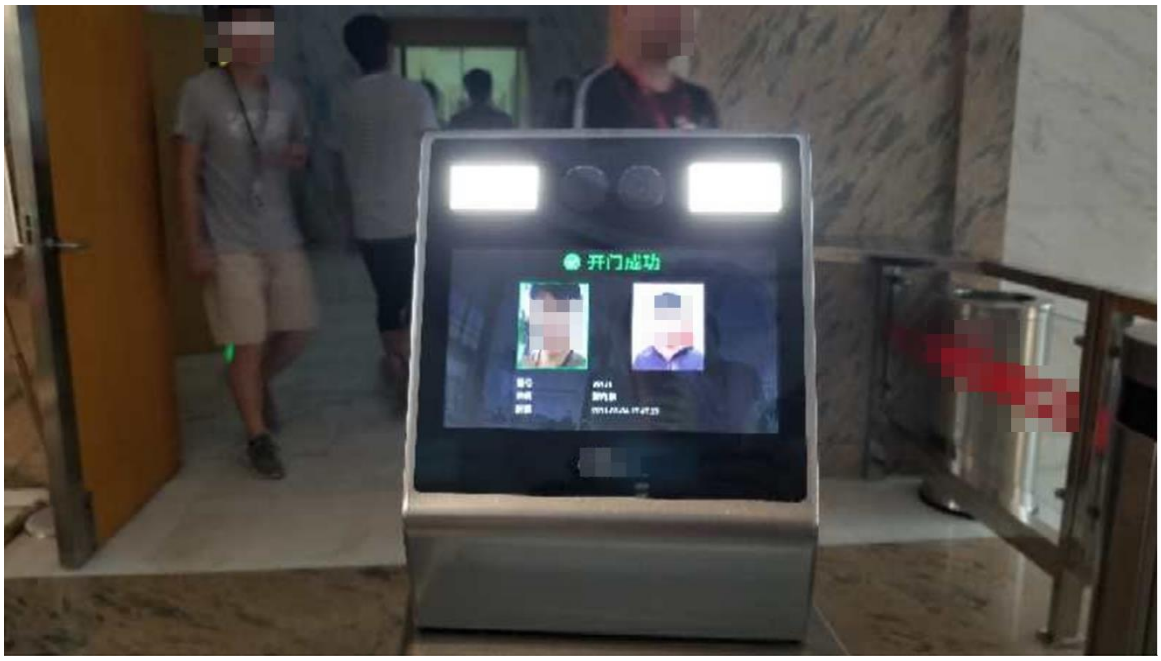
闸机主要应用园区、景区、学校、小区、办公楼等。将采集的人脸照片和人员信息上传平台后，再由平台将数据下发到闸机系统。

闸机外观如图 1-9 所示。

图1-9 闸机外观

闸机可以通过人脸或者刷卡的方式开门，人脸开门如图 1-10 所示。

图1-10 人脸开门



第 2 章 通用功能

2.1 NetSDK 初始化

2.1.1 简介

初始化是 SDK 进行各种业务的第一步。初始化本身不包含监控业务，但会设置一些影响全局业务的参数。

- SDK 的初始化将会占用一定的内存。
- 同一个进程内，只有第一次初始化有效。
- 使用完毕后需要调用 SDK 清理接口以释放资源。

2.1.2 接口总览

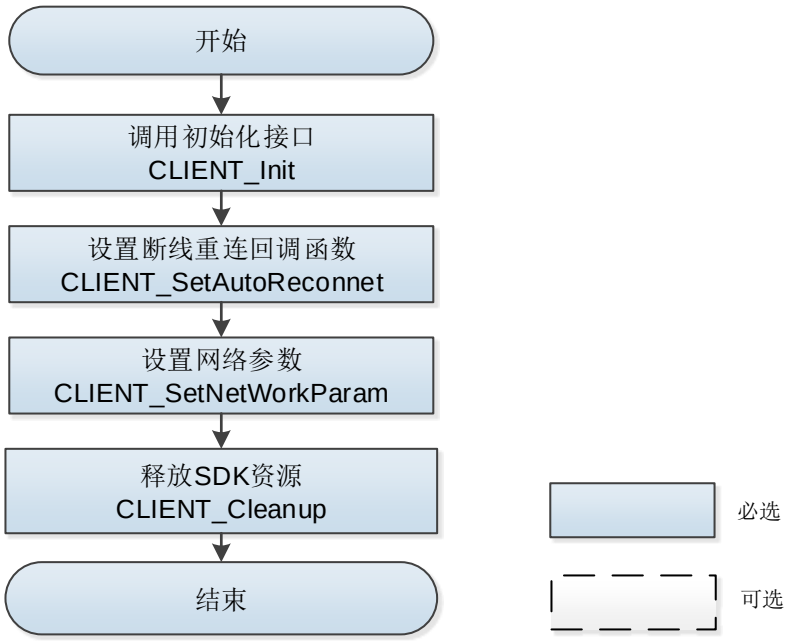
表2-1 SDK 初始化接口信息

接口	说明
CLIENT_Init	SDK 初始化接口
CLIENT_Cleanup	SDK 清理接口
CLIENT_SetAutoReconnect	设置断线重连回调接口
CLIENT_SetNetworkParam	设置登录网络环境接口

2.1.3 流程说明

SDK 初始化业务流程如图 2-1 所示。

图2-1 SDK 初始化业务流程



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。

步骤2　（可选）调用 CLIENT_SetAutoReconnect 设置断线重连回调函数，设置后 SDK 内部断线自动重连。

步骤3　（可选）调用 CLIENT_SetNetworkParam 设置网络登录参数，参数中包含登录设备超时时间和尝试次数。

步骤4　SDK 所有功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- SDK 的 CLIENT_Init 和 CLIENT_Cleanup 接口需成对调用，支持单线程多次成对调用，但建议全局调用一次。
- 初始化：CLIENT_Init 接口内部多次调用时，仅在内部用做计数，不会重复申请资源。
- 清理：CLIENT_Cleanup 接口内会清理所有已开启的业务，如登录、实时预览和报警订阅等。
- 断线重连：SDK 可以设置断线重连功能，当遇到一些特殊情况（例如断网、断电等）设备断线时，在 SDK 内部会定时持续不断地进行登录操作，直至成功登录设备。断线重连后可以恢复实时预览和录像回放业务，其他业务无法恢复。

2.1.4 示例代码

```
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;

/**
 * 登录接口实现
 * 主要有 ：初始化、登录、登出功能
 */
public class LoginModule {

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

    // 登录句柄
    public static LLong m_hLoginHandle = new LLong(0);

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

    //初始化
```

```
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;
}

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

    // 设置断线重连回调接口，设置过断线重连成功回调函数后，当设备出现断线情
况，SDK 内部会自动进行重连操作
// 此操作为可选操作，但建议用户进行设置
netsdk.CLIENT_SetAutoReconnect(haveReConnect, null);

//设置登录超时时间和尝试次数，可选
int waitTime = 5000; //登录请求响应超时时间设置为 5S
int tryTimes = 1;      //登录时尝试建立链接 1 次
netsdk.CLIENT_SetConnectTime(waitTime, tryTimes);

    // 设置更多网络参数，NET_PARAM 的 nWaittime，nConnectTryNum 成员与
CLIENT_SetConnectTime
// 接口设置的登录设备超时时间和尝试次数意义相同，可选
NetSDKLib.NET_PARAM netParam = new NetSDKLib.NET_PARAM();
netParam.nConnectTime = 10000;          // 登录时尝试建立链接的超时时间
netParam.nGetConnInfoTime = 3000;     // 设置子连接的超时时间
netsdk.CLIENT_SetNetworkParam(netParam);

return true;
```

```
    }

    //清除环境
    public static void cleanup() {
        if(bLogopen) {
            netsdk.CLIENT_LogClose();
        }

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

2.2 设备登录

2.2.1 简介

设备登录，即用户鉴权，是进行其他业务的前提。

用户登录设备产生唯一的登录 ID，其他功能的 SDK 接口需要传入登录 ID 才可执行。登出设备后，登录 ID 失效。

2.2.2 接口总览

表2-2 设备登录接口信息

接口	说明
CLIENT_LoginWithHighLevelSecurity	高安全级别登录接口
CLIENT_Logout	登出接口

2.2.3 流程说明

登录业务流程如图 2-2 所示。

图2-2 登录业务流程

流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　登录成功后，用户可以实现需要的业务功能。
步骤4　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤5　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- 登录句柄：登录成功时接口返回值非 0（即句柄可能小于 0，也属于登录成功）；同一设备登录多次，每次的登录句柄不一样。如果无特殊业务，建议只登录一次，登录的句柄可以重复用于其他各种业务。
- 句柄重复：登录句柄有可能与存在过的句柄相同，属于正常现象。例如登录设备 A 获得 loginIDA，将 loginIDA 注销，再次进行登录操作，可能又获取到 LoginIDA。但是在句柄的整个生命周期内，不会出现重复的句柄。
- 登出：接口内部会释放登录会话中已打开的业务，但建议用户不要依赖登出接口的清理功能。例如打开预览后，在不需要使用预览时，用户应该调用结束预览的接口。
- 登录与登出配对使用，登录会消耗一定的内存和 sokcet 信息，在登出后释放资源。
- 登录失败：建议通过登录接口的 error 参数（登录错误码）初步排查。常见错误码请参见表 2-3。
- 多设备登录：SDK 初始化后，可以登录多台设备，但是相应的登录句柄、登录信息需要调整。

表2-3 常见错误码及含义

error 的错误码	含义
1	密码不正确

error 的错误码	含义
2	用户名不存在

3	登录超时。规避示例代码如下：
	NET_PARAM stuNetParam = new NET_PARAM();
	stuNetParam.nWaittime = 8000; // unit ms
	CLIENT_SetNetworkParam (stuNetParam);

4	账号已登录
5	账号已被锁定
6	账号被列为禁止名单
7	资源不足，设备系统忙
8	子连接失败
9	主连接失败
10	超过最大用户连接数
11	缺少 avnetsdk 或 avnetsdk 的依赖库
12	设备未插入 U 盘或 U 盘信息错误
13	客户端 IP 地址没有登录权限

2.2.4 示例代码

```
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 初始化，SDK 清理省略

    // 设备信息
    public static NetSDKLib.NET_DEVICEINFO_Ex m_stDeviceInfo = new
    NetSDKLib.NET_DEVICEINFO_Ex();

    //登录句柄
    public static LLong m_hLoginHandle = new LLong(0);

    //登录设备
    public static boolean login(String m_strIp, int m_nPort, String m_strUser, String
    m_strPassword) {
    //入参
```

```
    NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY pstInParam=
    new NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY();
    pstInParam. szIP= m_strIp;
    pstInParam.nport= m_nPort;
    pstInParam.szUserName= m_strUser;
    pstInParam.szPassword= m_strPassword;
    //出参
    NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY pstOutParam=
    new NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY();
    m_hLoginHandle =
    netsdk.CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIGHLEVEL_SEC
    URITY pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY pstOutParam);

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

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

    //登出设备
    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 实时预览

2.3.1 简介

实时预览，即向存储设备或前端设备获取实时码流的功能，是监控系统的重要组成部分。

SDK 登录设备后，可向设备获取主码流和辅码流。

- 支持用户传入窗口句柄，SDK 直接进行码流解析及播放（此功能仅限 Windows 版本）。

- 支持回调实时码流数据给用户，让用户自己处理。
- 支持保存实时录像到指定文件，用户可通过自行保存回调码流实现，也可以通过调用 SDK 接口实现。

2.3.2 接口总览

表2-4 实时预览接口信息

接口	说明
CLIENT_RealPlayEx	开始实时预览扩展接口
CLIENT_StopRealPlayEx	停止实时预览扩展接口
CLIENT_SaveRealData	开始本地保存实时预览数据
CLIENT_StopSaveRealData	停止本地保存实时预览数据
CLIENT_SetRealDataCallBackEx	设置实时预览数据回调函数扩展接口

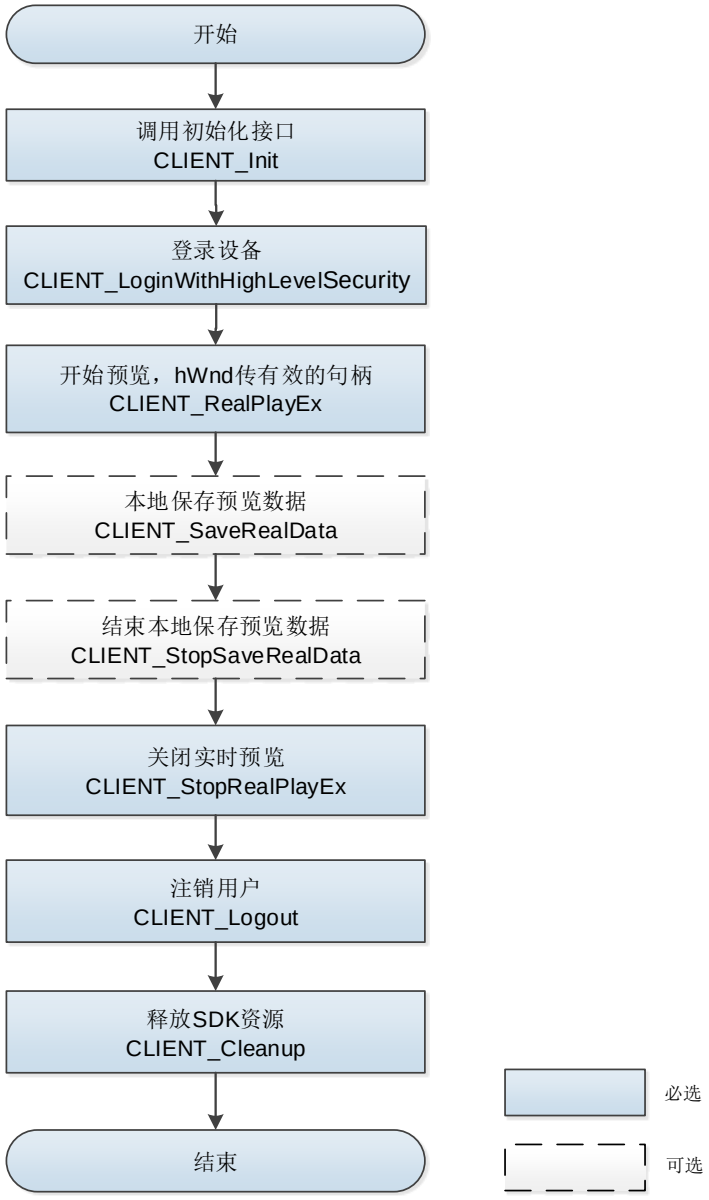
2.3.3 流程说明

实时监控的实现方式有两种，分别为 SDK 集成播放库进行播放及用户自己调用播放库播放码流方式进行播放。

2.3.3.1 SDK 解码播放

SDK 内部调用辅助库里的 PlaySDK 库实现实时播放。SDK 解码播放流程如图 2-3 所示。

图2-3 SDK 解码播放流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　调用 CLIENT_RealPlayEx 启动实时预览，参数 **hWnd** 为有效窗口句柄。
步骤4　（可选）调用 CLIENT_SaveRealData 开始保存预览数据。
步骤5　（可选）调用 CLIENT_StopSaveRealData 结束保存，生成本地视频文件。
步骤6　实时预览使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时预览。
步骤7　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤8　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- SDK 解码播放只支持 Windows 系统，非 windows 系统需要用户获取码流后自己调用解码显示。
- 多线程调用：同一个登录会话内的业务，不支持多线程调用；但可以多个线程处理不同的

登录会话中的业务，但不建议这样调用。

- 超时：接口内申请预览资源需和设备做一些约定，然后才请求预览数据，过程中有一些超时时间的设定（请参见 NET_PARAM 结构体），其中与预览相关的字段为 **nGetConnInfoTime**。如果实际使用中（如网络状况不良）有超时现象，可将 **nGetConnInfoTime** 的值修改大一些。示例代码如下，在 CLIENT_Init 函数后调用，调用一次即可：

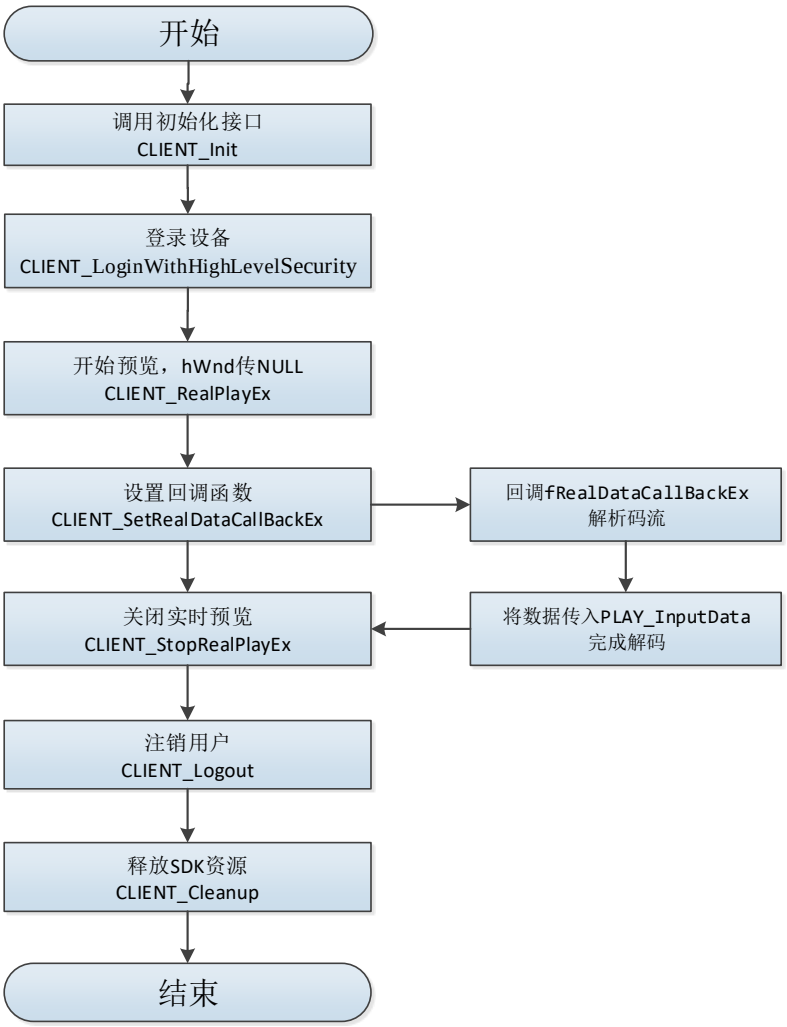
```
NET_PARAM stuNetParam = new NET_PARAM();
stuNetParam. nGetConnInfoTime = 5000; 为 0 默认为 1000ms
CLIENT_SetNetworkParam (stuNetParam);
```

- 重复打开失败：部分设备不支持同一个通道多次打开，当重复打开同一通道的预览，可能会出现第一次打开成功，后续打开失败的现象。建议：
  - ◇ 将已打开的通道先关闭。例如已经开启通道一的主码流视频，希望再打开通道一的辅码流视频时，可先关闭通道一的主码流视频，再开启通道一的辅码流视频。
  - ◇ 登录两次设备获取两个登录句柄，分别处理主码流和辅码流业务。
- 接口成功无画面：SDK 内部解码需要使用到 dhplay.dll，建议查看运行目录下是否缺少 dhplay.dll 及其依赖的辅助库，具体请参见表 1-1。
- 系统资源不足的情况下，设备可能返回错误而不恢复码流，可以在报警回调函数（即 CLIENT_SetDVRMessCallBack 中设置的回调函数）收到事件 DH_REALPLAY_FAILD_EVENT，该事件包含了详细的错误码，请参见《网络 SDK 开发手册》中的“DEV_PLAY_RESULT 结构体”。

2.3.3.2 调用私有播放库

SDK 回调实时预览码流给用户，用户调用 PlaySDK 进行解码播放。用户调用私有播放库解码播放流程如图 2-4 所示。

图2-4 第三方解码播放流程图



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　登录成功后，调用 CLIENT_RealPlayEx 启动实时预览，参数 **hWnd** 为 **NULL**。
步骤4　调用 CLIENT_SetRealDataCallBackEx 设置实时数据回调函数。
步骤5　在回调函数中将数据传给 PlaySDK 完成解码。
步骤6　实时预览使用完毕后，调用 CLIENT_StopRealPlayEx 停止实时预览。
步骤7　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤8　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

注意事项

- 码流格式：推荐使用 PlaySDK 解码。
- 画面卡顿：
  - ◇ 使用 PlaySDK 解码时，解码通道缓存大小有默认（PlaySDK 中的 PLAY_OpenStream 接口）。如果码流的分辨率很大，建议修改参数值，例如改为 3M。
  - ◇ SDK 回调函数需用户返回后才能回调出下一段视频数据，建议用户在回调中不要做耗时操作，否则会严重影响性能。

2.3.4 示例代码

2.3.4.1 SDK 解码播放

```
import java.awt.Panel;

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

import com.sun.jna.Native;

/**
 * 实时预览接口实现
 * 主要有 ：开始拉流、停止拉流功能
 */
public class RealPlayModule {
    // 开始预览
    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("开始实时预览失败，错误码" + ToolKits.getErrorCodePrint());
        } else {
            System.out.println("Success to start realplay");

   //自行定义码流保存文件，可选操作，如果需要保存视频则使用
   String outFile="example/outputfile";
   LoginModule.netsdk.CLIENT_SaveRealData(m_hPlayHandle,outFile);
        }
        return m_hPlayHandle;
    }

    //停止预览
    public static void stopRealPlay(LLong m_hPlayHandle) {
        if(m_hPlayHandle.longValue() == 0) {
            return;
        }
   //关闭文件保存。
   LoginModule.netsdk.CLIENT_StopSaveRealData(m_hPlayHandle);
        boolean bRet = LoginModule.netsdk.CLIENT_StopRealPlayEx(m_hPlayHandle);
        if(bRet) {
            m_hPlayHandle.setValue(0);
        }
    }
```

```
}
```

2.3.4.2 调用播放库

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



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



    LoginModule.netsdk.CLIENT_SetRealDataCallBackEx(m_hPlayHandle,m_DataCallBack
Ex, null, 0x00000001);

        if(m_hPlayHandle.longValue() == 0) {
            System.err.println("开始实时预览失败，错误码" + 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 智能事件订阅

2.4.1 简介

智能订阅，即智能设备对实时码流进行分析，当检测到预先设定好的事件时，将事件发送给用户。智能事件有交通违章、停车场有无车位等事件。

智能订阅实现方式为 SDK 主动连接设备，并向设备订阅智能事件功能，设备检测到智能事件立即发送给 SDK。

当前支持的智能订阅事件参见 NetSDKLib.java 中以 EVENT_IVS_开头的常量，包含了常规的交通占道、车辆违规等事件。

2.4.2 接口总览

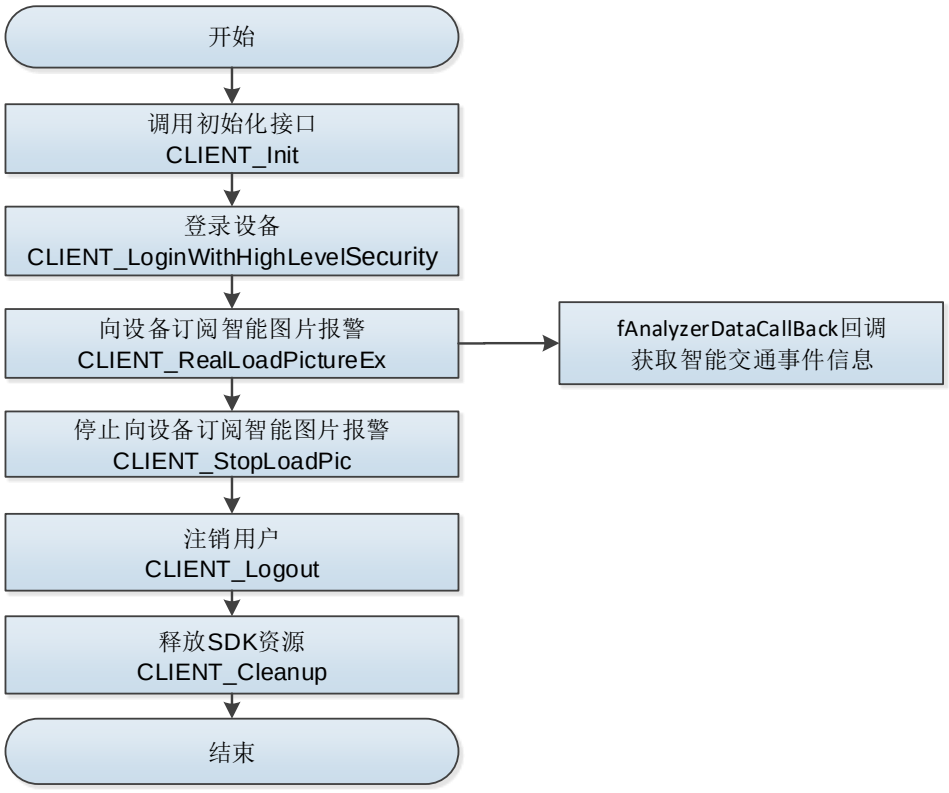
表2-5 智能交通时间上报的接口信息

接口	说明
CLIENT_RealLoadPictureEx	订阅智能事件
CLIENT_StopLoadPic	取消订阅智能事件
fAnalyzerDataCallBack	用于回调获取智能事件的信息

2.4.3 流程说明

智能订阅事件上报流程如图 2-5 所示。

图2-5 智能订阅事件上报业务流程



流程说明

步骤1　调用 CLIENT_Init 函数完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 函数登录设备。
步骤3　调用 CLIENT_RealLoadPictureEx 函数向设备订阅智能事件。
步骤4　订阅成功后设备上报的智能交通事件通过 fAnalyzerDataCallBack 回调函数获取智能事件并通知用户。
步骤5　智能事件上报功能使用完毕后，调用 CLIENT_StopLoadPic 函数停止订阅智能事件。
步骤6　业务使用完后，调用 CLIENT_Logout 函数登出设备。
步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 函数释放 SDK 资源。

注意事项

- 订阅事件类型：如果需要同时上报不同智能事件时，支持订阅所有智能事件（EVENT_IVS_ALL）；也支持订阅单个智能事件。
- 设置是否接收图片：由于某些设备所在网络环境是 3G 或 4G 网络，当 SDK 连接设备时，如不需要接收图片可以把 CLIENT_ RealLoadPictureEx 接口中 bNeedPicFile 参数设置为 False，只接收智能交通事件信息，不带图片。
- 通过通道号传-1 进行全通道订阅。部分智能交通类产品不支持全通道订阅。若传-1 订阅失败，请尝试单通道订阅。

2.4.4 示例代码

```
//本实例将用门禁事件举例
//省略 SDK 初始化以及门禁设备登录
// 订阅句柄
public static LLong m_hAttachHandle = new LLong(0);
private AnalyzerDataCB analyzerCallback = new AnalyzerDataCB();
private boolean isAttach = false;
// 监听
private void setOnClickListener() {
    // 订阅。即门禁设备订阅智能事件上报
    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);
            }
        }
```

```
        });

        // 取消订阅
        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();
            }
        });
    }
    //门禁系统智能事件回调，继承自 fAnalyzerDataCallBack 并自己实现逻辑
    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();
            }

            ///< 门禁事件
            if(dwAlarmType == NetSDKLib.EVENT_IVS_ACCESS_CTL) {
                    DEV_EVENT_ACCESS_CTL_INFO msg = new
            DEV_EVENT_ACCESS_CTL_INFO();
                ToolKits.GetPointerData(pAlarmInfo, msg);

                // 保存图片，获取图片缓存
               String snapPicPath = path + "\\" + System.currentTimeMillis() +
               "GateSnapPicture.jpg";  // 保存图片地址
                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();
                }

                // 图片以及门禁信息界面显示
                EventQueue eventQueue = Toolkit.getDefaultToolkit().getSystemEventQueue();
                if (eventQueue != null) {
                eventQueue.postEvent( new AccessEvent(target, gateBufferedImage, msg));


                                                                    }
            }

            return 0;
        }
}
```

第 3 章 目标检测

3.1 事件订阅

3.1.1 简介

当摄像头在指定区域检测到人脸出现时，产生智能事件消息并上报 NetSDK。

3.1.2 流程说明

本章节只介绍针对具体事件的回调处理。关于事件的订阅和接收流程，请参见“2.4 智能事件订阅。

3.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_FACEDETECT
- 事件对应的结构体：DEV_EVENT_FACEDETECT_INFO

3.2 示例代码

```
 * 写成静态主要是防止被回收
 */
private static class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack {

    private AnalyzerDataCB() {}

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

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

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

```
                switch(dwAlarmType)
                {
                    case NetSDKLib.EVENT_IVS_FACEDETECT:    ///< 目标检测
                    {
                                            DEV_EVENT_FACEDETECT_INFO msg = new
                        DEV_EVENT_FACEDETECT_INFO();
                        ToolKits.GetPointerData(pAlarmInfo, msg);

                        // 保存图片，获取图片缓存
                        try {
                            saveFaceDetectPic(pBuffer, dwBufSize, msg);
                        } catch (FileNotFoundException e) {
                            e.printStackTrace();
                        }

                        // 列表、图片界面显示
                       EventQueue.invokeLater(new FaceDetectRunnable(globalBufferedImage,
                       personBufferedImage, msg));

                        // 释放内存
                        msg = null;
                        System.gc();

                        break;
                    }
}
//停止订阅
if(m_hAttachHandle.longValue() != 0) {
                LoginModule.netsdk.CLIENT_StopLoadPic(m_hAttachHandle);
                m_hAttachHandle.setValue(0);
            }
```

第 4 章 目标识别

4.1 事件订阅

4.1.1 简介

目标识别事件是指检测到的人脸与服务器内部数据库中的人脸图片比对成功时，向平台上报事件。

目标识别事件中携带的信息包括识别出的人员信息、每个人员的图片文件、与当前人脸的相似度。

4.1.2 流程说明

本章节只介绍针对具体事件的回调处理。关于事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

4.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_FACERECOGNITION
- 事件对应的结构体：DEV_EVENT_FACERECOGNITION_INFO

4.2 示例代码

```
 * 写成静态主要是防止被回收
 */
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:  ///< 目标识别事件
                {
                    // DEV_EVENT_FACERECOGNITION_INFO 结构体比较大，new 对象会比较耗时， ToolKits.GetPointerData 内容拷贝是不耗时的。
                    // 如果多台设备或者事件处理比较频繁，可以考虑将 static DEV_EVENT_FACERECOGNITION_INFO msg = new DEV_EVENT_FACERECOGNITION_INFO(); 改为全局。
                    // 写成全局，是因为每次 new 花费时间较多，如果改为全局，此 case 下的处理需要加锁
                    // 加锁，是因为共用一个对象，防止数据出错

                    // 耗时 800ms 左右
                    DEV_EVENT_FACERECOGNITION_INFO msg = new DEV_EVENT_FACERECOGNITION_INFO();

                    // 耗时 20ms 左右
                    ToolKits.GetPointerData(pAlarmInfo, msg);

                    // 保存图片，获取图片缓存
                    // 耗时 20ms 左右
                    try {
                        saveTargetRecognitionPic(pBuffer, dwBufSize, msg);
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    }

                    // 列表、图片界面显示
                    // 回调属于子线程，以下是个 UI 线程，来刷新 UI
        EventQueue.invokeLater(new TargetRecognitionRunnable(globalBufferedImage, personBufferedImage, candidateBufferedImage, msg, index));




                    // 释放内存
                    msg = null;
                    System.gc();

                    break;
                }
}
```

```
//停止订阅
if(m_hAttachHandle.longValue() != 0) {
                LoginModule.netsdk.CLIENT_StopLoadPic(m_hAttachHandle);
                m_hAttachHandle.setValue(0);
            }
```

第 5 章 通用行为

5.1 事件订阅

5.1.1 简介

通用行为事件主要是指区域入侵和绊线入侵等事件。区域入侵是指指定区域内检测到有人闯入而报警。绊线入侵指检测到有人穿过相机设定的线时报警。

5.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

5.1.3 枚举和结构体

- 绊线入侵
  - ◇ 绊线入侵事件对应的枚举值：EVENT_IVS_CROSSLINEDETECTION
  - ◇ 绊线入侵事件对应的结构体：DEV_EVENT_CROSSLINE_INFO
- 区域入侵事件
  - ◇ 区域入侵事件对应的枚举值：EVENT_IVS_CROSSREGIONDETECTION
  - ◇ 区域入侵事件对应的结构体：DEV_EVENT_CROSSREGION_INFO

5.2 示例代码

```
/**
 * 通用行为回调
 */
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;
    }

    // 回调
    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 : // 警戒线事件
            {
                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("【警戒线事件】 时间(UTC):%s 通道号:%d 开始时间:%s
                结束时间:%s 事件触发累计次数:%d 事件源设备唯一标识:%s \n",
                msg.UTC, msg.nChannelID, msg.stuObject.stuStartTime,
                msg.stuObject.stuEndTime,
                msg.nOccurrenceCount, new String(msg.szSourceDevice));
                break;
            }
            case NetSDKLib.EVENT_IVS_CROSSREGIONDETECTION: ///<  警戒区事件
            {
                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(" 警戒区事件 时间(UTC)：" + msg.UTC + " 通道号:" +
                msg.nChannelID + " 开始时间:" + msg.stuObject.stuStartTime + " 结束时
                间:" + msg.stuObject.stuEndTime);
//                  PrintStruct.print(msg);
                break;
            }
            }
```

```
        }
```

第 6 章 人体检测

6.1 事件订阅

6.1.1 简介

当摄像头在指定区域检测到人体特征时，产生智能事件消息并上报 NetSDK。

6.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

6.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_HUMANTRAIT
- 事件对应的结构体：DEV_EVENT_HUMANTRAIT_INFO

6.2 示例代码

```
/**
 * 人体检测回调
 */
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;
    }
```

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

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

        switch(dwAlarmType)
        {
                case NetSDKLib.EVENT_IVS_HUMANTRAIT:    // 人体特征事件
                {
                    DEV_EVENT_HUMANTRAIT_INFO msg = new
                    DEV_EVENT_HUMANTRAIT_INFO();
                    ToolKits.GetPointerData(pAlarmInfo, msg);
                    PrintStruct.print(msg);

                    //保存全景图片
                    if(msg.stuSceneImage.nLength>0)
                    {
                        String strFileName = path + "\\" + System.currentTimeMillis() +
                        "HumanTrait_全景图.jpg";
                        ToolKits.savePicture(pBuffer, msg.stuSceneImage.nOffSet,
                        msg.stuSceneImage.nLength, strFileName);
                    }
                    else
                    {
                        System.out.println("无全景图");
                    }

                    //保存人脸图
                    if(msg.stuFaceImage.nLength>0)
                    {
                        String strFileName = path + "\\" + System.currentTimeMillis() +
                        "HumanTrait_人脸图.jpg";
                         ToolKits.savePicture(pBuffer, msg.stuFaceImage.nOffSet,
                         msg.stuFaceImage.nLength, strFileName);
                    }
                    else
                    {
                        System.out.println("无人脸图");
                    }
```

```
                    //保存人脸全景图
                    if(msg.stuFaceSceneImage.nLength>0)
                    {
                        String strFileName = path + "\\" + System.currentTimeMillis() +
                        "HumanTrait_人脸全景图.jpg";
                         ToolKits.savePicture(pBuffer, msg.stuFaceSceneImage.nOffSet,
                         msg.stuFaceSceneImage.nLength, strFileName);
                    }
                    else
                    {
                        System.out.println("无人脸全景图");
                    }

                    //保存人体图
                    if(msg.stuHumanImage.nLength>0)
                    {
                        String strFileName = path + "\\" + System.currentTimeMillis() +
                        "HumanTrait_人体图.jpg";
                         ToolKits.savePicture(pBuffer, msg.stuHumanImage.nOffSet,
                         msg.stuHumanImage.nLength, strFileName);
                    }
                    else
                    {
                        System.out.println("无人体图");
                    }

                    break;
                }
```

第 7 章 热成像测温事件

7.1 事件订阅

7.1.1 简介

当 TPC 热成像设备在指定区域检测到人体时，会通过热成像技术上报人体温度。

7.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

7.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_ANATOMY_TEMP_DETECT
- 事件对应的结构体：DEV_EVENT_ANATOMY_TEMP_DETECT_INFO

7.2 示例代码

```
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;
    }

    // 回调
    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 :
           // 人体温度智能检测事件
                    {
                         NetSDKLib.DEV_EVENT_ANATOMY_TEMP_DETECT_INFO msg =
                         new NetSDKLib.DEV_EVENT_ANATOMY_TEMP_DETECT_INFO();
                        ToolKits.GetPointerData(pAlarmInfo, msg);

                         System.out.printf("【人体温智能检测事件 】 时间(UTC):%s 通道
                         号:%d nAction:%d szName:%s nPresetID:%d \n",
                         msg.UTC, msg.nChannelID, msg.nAction, new
                         String(msg.szName).trim(), msg.nPresetID);

                         System.out.printf("【区域内人员体温信息】
                         nObjectID"+msg.stManTempInfo.nObjectID+"dbHighTemp"+msg.stMa
                         nTempInfo.dbHighTemp+"
                         nTempUnit"+msg.stManTempInfo.nTempUnit+"bIsOverTemp"+msg.stM
                         anTempInfo.bIsOverTemp+"bIsUnderTemp"+msg.stManTempInfo.bIsU
                         nderTemp+"\n");
                        //可见光全景图
                        if(msg.stVisSceneImage!=null && msg.stVisSceneImage.nLength> 0){
                             String bigPicture = picturePath + "\\" + System.currentTimeMillis() +
                             ".jpg";
                              ToolKits.savePicture(pBuffer, msg.stVisSceneImage.nOffset,
                              msg.stVisSceneImage.nLength, bigPicture);
                        }
                        //热成像全景图
               if(msg.stThermalSceneImage!=null && msg.stThermalSceneImage.nLength> 0){
                 String smallPicture = picturePath + "\\" + System.currentTimeMillis() +
"small.jpg";
                    ToolKits.savePicture(pBuffer, msg.stThermalSceneImage.nOffset,
                    msg.stThermalSceneImage.nLength, smallPicture);
                         }
                         break;
                    }
}
```

第 8 章 门禁事件

8.1 事件订阅

8.1.1 简介

门禁设备开门时，上报开门相关的事件信息，包括事件、开门方式、开门人员对应信息等。

8.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

8.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_ACCESS_CTL
- 事件对应的结构体：DEV_EVENT_ACCESS_CTL_INFO

8.2 示例代码

```
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();
            }

            ///< 门禁事件
            if(dwAlarmType == NetSDKLib.EVENT_IVS_ACCESS_CTL) {
                              DEV_EVENT_ACCESS_CTL_INFO msg = new
                DEV_EVENT_ACCESS_CTL_INFO();
                ToolKits.GetPointerData(pAlarmInfo, msg);
```

```
                // 保存图片，获取图片缓存
                String snapPicPath = path + "\\" + System.currentTimeMillis() +
                "GateSnapPicture.jpg";   // 保存图片地址
                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();
                }

                // 图片以及门禁信息界面显示
                EventQueue eventQueue =
                Toolkit.getDefaultToolkit().getSystemEventQueue();
                if (eventQueue != null) {
                eventQueue.postEvent( new AccessEvent(target,gateBufferedImage,msg));
                }
            }

            return 0;
        }
```

第 9 章 客流量统计

9.1 简介

客流量统计功能指在经营区域安装前端设备，智能分析服务器根据前端采集的视频数据精确统计出每个入口实时客流进出人数。该类产品被广泛应用于大型商业、旅游行业、公共安全、文博和连锁等行业。

订阅客流量实时数据后，可获取到当天总出入人数和实时出入人数情况。

9.2 接口总览

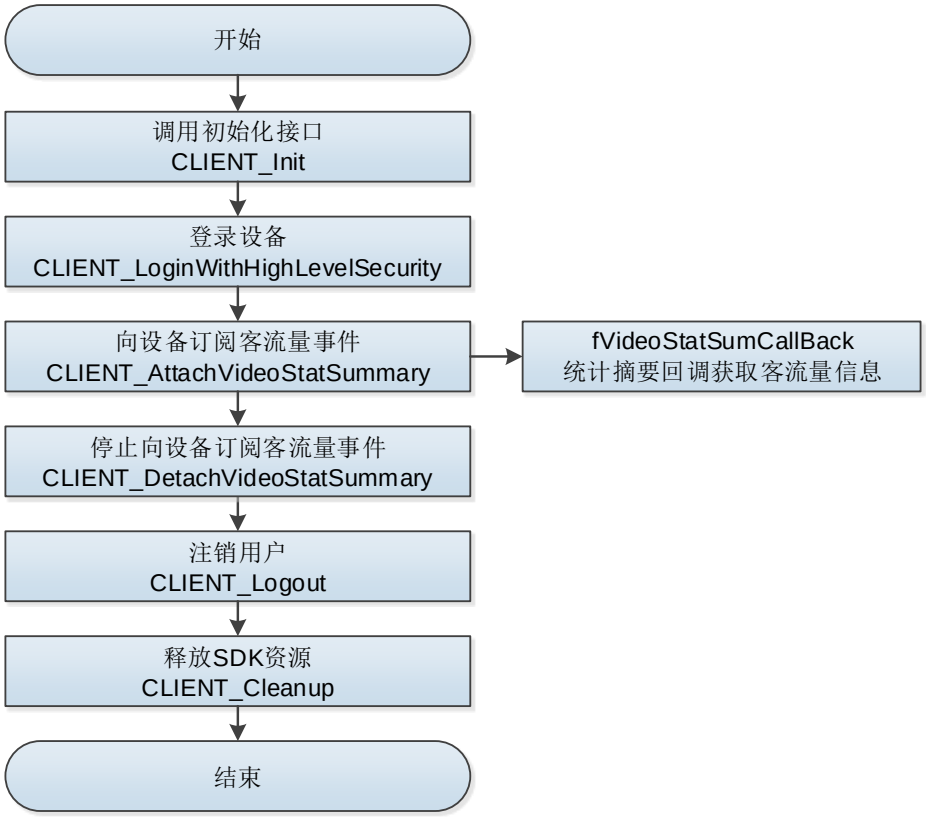
表9-1 客流量统计接口说明

接口	说明
CLIENT_AttachVideoStatSummary	订阅客流量事件
CLIENT_DetachVideoStatSummary	退订客流量事件

9.3 流程说明

客流量订阅流程，如图 9-1 所示。

图9-1 客流量订阅流程



流程说明

步骤1　调用 CLIENT_Init 完成 SDK 初始化流程。
步骤2　初始化成功后，调用 CLIENT_LoginWithHighLevelSecurity 登录设备。
步骤3　调用 CLIENT_AttachVideoStatSummary 向设备订阅客流量事件。
步骤4　订阅成功后，设备上报的客流量事件通过 fVideoStatSumCallBack 回调函数获取客流量事件并通知给用户。
步骤5　客流量事件上报功能使用完毕后，调用 CLIENT_DetachVideoStatSummary 停止订阅客流量事件。
步骤6　业务使用完后，调用 CLIENT_Logout 登出设备。
步骤7　SDK 功能使用完后，调用 CLIENT_Cleanup 释放 SDK 资源。

9.4 示例代码

```
/**
 * 订阅
 */
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");
}

/**
 * 退订
 */
public void detachVideoStatSummary() {
    if (videoStatHandle.longValue() != 0) {
        netsdkApi.CLIENT_DetachVideoStatSummary(videoStatHandle);
```

```
                videoStatHandle.setValue(0);
            }
        }
            }
/*
  * 客流量统计回调
*/
    private static class VideoStatSumCallback implements
NetSDKLib.fVideoStatSumCallBack {
        private static VideoStatSumCallback instance = new VideoStatSumCallback();
        private VideoStatSumCallback() {}
        public static VideoStatSumCallback getInstance() {
            return instance;
        }

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

第 10 章 智能交通事件

10.1 事件订阅

10.1.1 简介

智能交通事件上报，即智能交通设备对实时码流进行分析，当检测到预先设定好的交通事件时，将交通事件发送给用户。智能交通事件有交通违章、停车场有无车位等事件。

智能交通事件上报实现方式为 SDK 主动连接设备，并向设备订阅智能事件，设备检测到智能事件立即发送给 SDK。

10.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

10.1.3 枚举和结构体

- 交通路口事件
  - ◇ 交通路口事件对应的枚举值：EVENT_IVS_TRAFFICJUNCTION
  - ◇ 交通路口事件对应的结构体：DEV_EVENT_TRAFFICJUNCTION_INFO
- 交通违章-压车道线事件
  - ◇ 交通违章-压车道线事件对应的枚举值：EVENT_IVS_TRAFFIC_OVERLINE
  - ◇ 交通违章-压车道线事件对应的结构体：DEV_EVENT_TRAFFIC_OVERLINE_INFO
- 交通违章-逆行事件
  - ◇ 交通违章-逆行事件对应的枚举值：EVENT_IVS_TRAFFIC_RETROGRADE
  - ◇ 交通违章-逆行事件对应的结构体：DEV_EVENT_TRAFFIC_RETROGRADE_INFO
- 交通-闯红灯事件
  - ◇ 交通-闯红灯事件对应的枚举值：EVENT_IVS_TRAFFIC_RUNREDLIGHT
  - ◇ 交通-闯红灯事件对应的结构体：DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO
- 交通-违章左转事件
  - ◇ 交通-违章左转事件对应的枚举值：EVENT_IVS_TRAFFIC_TURNLEFT
  - ◇ 交通-违章左转事件对应的结构体：DEV_EVENT_TRAFFIC_TURNLEFT_INFO
- 交通违章-违章右转事件
  - ◇ 交通违章-违章右转事件对应的枚举值：EVENT_IVS_TRAFFIC_TURNRIGHT
  - ◇ 交通违章-违章右转事件对应的结构体：DEV_EVENT_TRAFFIC_TURNRIGHT_INFO
- 交通违章-违章掉头事件
  - ◇ 交通违章-违章掉头事件对应的枚举值：EVENT_IVS_TRAFFIC_UTURN
  - ◇ 交通违章-违章掉头事件对应的结构体：DEV_EVENT_TRAFFIC_UTURN_INFO
- 交通违章-低速事件
  - ◇ 交通违章-低速事件对应的枚举值：EVENT_IVS_TRAFFIC_UNDERSPEED
  - ◇ 交通违章-低速事件对应的结构体：DEV_EVENT_TRAFFIC_UNDERSPEED_INFO

- 交通违章-违章停车
  - ◇ 交通违章-违章停车对应的枚举值：EVENT_IVS_TRAFFIC_PARKING
  - ◇ 交通违章-违章停车对应的结构体：DEV_EVENT_TRAFFIC_PARKING_INFO
- 交通违章-不按车道行驶事件
  - ◇ 交通违章-不按车道行驶事件对应的枚举值：EVENT_IVS_TRAFFIC_WRONGROUTE
  - ◇ 交通违章-不按车道行驶事件对应的结构体：DEV_EVENT_TRAFFIC_WRONGROUTE_INFO
- 交通违章-违章变道事件
  - ◇ 交通违章-违章变道事件对应的枚举值：EVENT_IVS_TRAFFIC_CROSSLANE
  - ◇ 交通违章-违章变道事件对应的结构体：DEV_EVENT_TRAFFIC_CROSSLANE_INFO
- 交通违章-压黄线事件
  - ◇ 交通违章-压黄线事件对应的枚举值：EVENT_IVS_TRAFFIC_OVERYELLOWLINE
  - ◇ 交通违章-压黄线事件对应的结构体：DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO
- 交通违章-黄牌车占道事件
  - ◇ 交通违章-黄牌车占道事件对应的枚举值：EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE
  - ◇ 交通违章-黄牌车占道事件对应的结构体：DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO
- 交通违章-斑马线行人优先事件
  - ◇ 交通违章-斑马线行人优先事件对应的枚举值：EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY
  - ◇ 交通违章-斑马线行人优先事件对应的结构体：DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO
- 交通手动抓拍事件
  - ◇ 交通手动抓拍事件对应的枚举值：EVENT_IVS_TRAFFIC_MANUALSNAP
  - ◇ 交通手动抓拍事件对应的结构体：DEV_EVENT_TRAFFIC_MANUALSNAP_INFO
- 有车占道事件
  - ◇ 有车占道事件对应的枚举值：EVENT_IVS_TRAFFIC_VEHICLEINROUTE
  - ◇ 有车占道事件对应的结构体：DEV_EVENT_TRAFFIC_VEHICLEINROUTE_INFO
- 交通违章--占用公交车道事件
  - ◇ 交通违章-占用公交车道事件对应的枚举值：EVENT_IVS_TRAFFIC_VEHICLEINBUSROUTE
  - ◇ 交通违章-占用公交车道事件对应的结构体：DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INFO
- 交通违章--违章倒车事件
  - ◇ 交通违章-违章倒车事件对应的枚举值：EVENT_IVS_TRAFFIC_BACKING
  - ◇ 交通违章-违章倒车事件对应的结构体：DEV_EVENT_IVS_TRAFFIC_BACKING_INFO
- 车位有车事件
  - ◇ 车位有车事件对应的枚举值：EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING
  - ◇ 车位有车事件对应的结构体：DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO
- 车位无车事件
  - ◇ 车位无车事件对应的枚举值：EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING
  - ◇ 车位无车事件对应的结构体：DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO

- 交通违章--交通未系安全带事件
  - ◇ 交通违章-交通未系安全带事件对应的枚举值：EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT
  - ◇ 交通违章-交通未系安全带事件对应的结构体：DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT

10.2 示例代码

```
/*
 * 智能报警事件回调
 */
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) {

          // 获取识别对象 车身对象 事件发生时间 车道号等信息
          GetStuObject(dwAlarmType, pAlarmInfo);
```

```
        // 保存图片，获取图片缓存
          savePlatePic(pBuffer, dwBufSize, trafficInfo);

        // 列表、图片界面显示
          EventQueue eventQueue =Toolkit.getDefaultToolkit().getSystemEventQueue();
          if (eventQueue != null)
            {
          eventQueue.postEvent(new TrafficEvent(target,snapImage,plateImage,trafficInfo));
                }
            }

            return 0;
        }

        // 获取识别对象 车身对象 事件发生时间 车道号等信息
        private void GetStuObject(int dwAlarmType, Pointer pAlarmInfo)    {
            if(pAlarmInfo == null) {
                return;
            }

            switch(dwAlarmType) {
                case NetSDKLib.EVENT_IVS_TRAFFICJUNCTION: ///< 交通卡口事件
                {
                    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: ///< 闯红灯事件
            {
                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_RUNR
                 EDLIGHT);
                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: ///< 压车道线事件
            {
                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_OVER
                 LINE);
                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: ///< 逆行事件
            {
               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_RETR
                OGRADE);
                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_TURNLEFT: ///< 违章左转
            {
                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_TURNL
                EFT);
                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: ///< 违章右转
            {
                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_TURN
                RIGHT);
                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: ///< 违章掉头
            {
                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_UTUR
                N);
                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: ///< 超速
            {
                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_OVER
                SPEED);
                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: ///< 低速
            {
                NetSDKLib.DEV_EVENT_TRAFFIC_UNDERSPEED_INFO msg = new
                NetSDKLib.DEV_EVENT_TRAFFIC_UNDERSPEED_INFO();
                ToolKits.GetPointerData(pAlarmInfo, msg);

                trafficInfo.m_EventName =
                Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_UNDE
                RSPEED);
                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: ///< 违章停车
            {
                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_PAR
                KING);
                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: ///<  不按车道行驶
            {
                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_WRON
                GROUTE);
                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: ///< 违章变道
            {
                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_CROS
                SLANE);
                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: ///< 压黄线
            {
                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_OVER
                YELLOWLINE);
                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:
            ///< 黄牌车占道事件
            {
                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_YELLO
                WPLATEINLANE);
                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:
            ///< 斑马线行人优先事件
            {
                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_PEDES
                TRAINPRIORITY);
                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:
```

```
                ///< 交通手动抓拍事件
                {
                    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_MANU
                    ALSNAP);
                    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_VEHICLEINROUTE:
                ///< 有车占道事件
                {
                  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_VEHIC
                LEINROUTE);
                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:
            ///<  占用公交车道事件
            {
                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_VEHIC
                LEINBUSROUTE);
```

```
                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: ///< 违章倒车事件
            {
                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_BACKI
                NG);
                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:
            ///< 车位有车事件
            {
                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_PARKI
                NGSPACEPARKING);
                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_PARKINGSPACENOPARKING:
            ///< 车位无车事件
            {

                NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_IN
                FO msg = new
                NetSDKLib.DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_IN
                FO();
                ToolKits.GetPointerData(pAlarmInfo, msg);

                trafficInfo.m_EventName =
                Res.string().getEventName(NetSDKLib.EVENT_IVS_TRAFFIC_PARKI
                NGSPACENOPARKING);
                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:
            ///< 交通未系安全带事件
            {
                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_WITHO
                UT_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;
            }
        }
```

第 11 章 人证比对

11.1 事件订阅

11.1.1 简介

将检测到的人员信息与证件信息比较，检测人员是否匹配。

11.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

11.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_CITIZEN_PICTURE_COMPARE
- 事件对应的结构体：DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO

11.2 示例代码

```
        /* 智能报警事件回调 */
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:
    //人证比对事件
    {
        DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO msg = new
        DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO();
        ToolKits.GetPointerData(pAlarmInfo, msg);

        try {
            System.out.println("事件发生时间：" + msg.stuUTC.toString());
            System.out.println("事件名称 :" + new String(msg.szName,
            "GBK").trim());

            // 人证比对结果,相似度大于等于阈值认为比对成功, 1-表示成功, 0-
            表示失败
            System.out.println("比对结果:" + msg.bCompareResult);

            System.out.println("两张图片的相似度:" + msg.nSimilarity);
            System.out.println("检测阈值:" + msg.nThreshold);

            if (msg.emSex == 1) {
                System.out.println("性别:男");
            }else if (msg.emSex == 2){
                System.out.println("性别:女");
            }else {
                System.out.println("性别:未知");
            }



            System.out.println("居民姓名:" + new String(msg.szCitizen,
            "GBK").trim());
            System.out.println("住址:" + new String(msg.szAddress,
            "GBK").trim());
            System.out.println("身份证号:" + new String(msg.szNumber).trim());
            System.out.println("签发机关:" + new String(msg.szAuthority,
            "GBK").trim());

            System.out.println("出生日期:" + msg.stuBirth.toStringTimeEx());
            System.out.println("有效起始日期:" +
            msg.stuValidityStart.toStringTimeEx());
```

```
                if (msg.bLongTimeValidFlag == 1) {
                        System.out.println("有效截止日期：永久");
                }else{
                        System.out.println("有效截止日期:"+
                msg.stuValidityEnd.toStringTimeEx());
                }
                System.out.println("IC 卡号：" + new String(msg.szCardNo,
                "GBK").trim());
        } catch (Exception e) {
                e.printStackTrace();
        }

        // 拍摄照片
                String strFileName = path + "\\" + System.currentTimeMillis() +
        "citizen_snap.jpg";
                        byte[] snapBuffer =
         pBuffer.getByteArray(msg.stuImageInfo[0].dwOffSet,
         msg.stuImageInfo[0].dwFileLenth);
                        ByteArrayInputStream snapArrayInputStream = new
         ByteArrayInputStream(snapBuffer);
         try {
                snapBufferedImage = ImageIO.read(snapArrayInputStream);
                if(snapBufferedImage == null) {
                        return 0;
                }
                ImageIO.write(snapBufferedImage, "jpg", new File(strFileName));

         } catch (IOException e) {
                e.printStackTrace();
         }

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

```
                    break;
                }
                default:
                    break;
            }

            return 0;
        }
    }
```

第 12 章 工装检测事件

12.1 事件订阅

12.1.1 简介

工装(安全帽/工作服等)检测事件

12.1.2 流程说明

本章节只介绍针对具体事件的回调处理。事件的订阅和接收流程，请参见“2.4 智能事件订阅”。

12.1.3 枚举和结构体

- 事件对应的枚举值：EVENT_IVS_WORKCLOTHES_DETECT
- 事件对应的结构体：DEV_EVENT_WORKCLOTHES_DETECT_INFO

12.2 示例代码

```
/* 智能报警事件回调 */
    private static class AnalyzerDataCB implements NetSDKLib.fAnalyzerDataCallBack {
        private final File picturePath;
        private static AnalyzerDataCB instance;

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

        public static AnalyzerDataCB getInstance() {
            if (instance == null) {
                synchronized (AnalyzerDataCB.class) {
                    if (instance == null) {
                        instance = new AnalyzerDataCB();
                    }
                }
            }
            return instance;
        }
```

```
        public int invoke(LLong lAnalyzerHandle, int dwAlarmType, Pointer pAlarmInfo,
Pointer pBuffer, int dwBufSize,
                Pointer dwUser, int nSequence, Pointer reserved) {
            if (lAnalyzerHandle == null || lAnalyzerHandle.longValue() == 0) {
                return -1;
            }
            System.out.println("dwAlarmType:"+dwAlarmType);
            switch
(Objects.requireNonNull(EM_EVENT_IVS_TYPE.getEventType(dwAlarmType))) {
            case EVENT_IVS_WORKCLOTHES_DETECT: {// 工装(安全帽/工作服等)检测事
件(对应 DEV_EVENT_WORKCLOTHES_DETECT_INFO)
                NetSDKLib.DEV_EVENT_WORKCLOTHES_DETECT_INFO msg = new
NetSDKLib.DEV_EVENT_WORKCLOTHES_DETECT_INFO();
                ToolKits.GetPointerData(pAlarmInfo, msg);
                if (msg.stuSceneImage != null && msg.stuSceneImage.nLength > 0) {
                    String bigPicture = picturePath + "\\" +
"EVENT_IVS_WORKCLOTHES_DETECT" + System.currentTimeMillis()
                            + ".jpg";
                    ToolKits.savePicture(pBuffer, msg.stuSceneImage.nOffSet,
msg.stuSceneImage.nLength, bigPicture);
                    if (msg.stuHumanImage != null && msg.stuHumanImage.nLength > 0) {
                        String smallPicture = picturePath + "\\" +
"EVENT_IVS_WORKCLOTHES_DETECT"
                                + System.currentTimeMillis() + "small.jpg";
                        ToolKits.savePicture(pBuffer, msg.stuHumanImage.nOffSet,
msg.stuHumanImage.nLength,
                                smallPicture);
                    }
                }
                System.out.println(" 工装(安全帽/工作服等)检测事件(UTC)： " + msg.UTC +
" 通道号:" + msg.nChannelID + " 工作裤属性颜色:"
                        + msg.stuWorkPantsAttribute.emWorkPantsColor + " 事件 ID:" +
msg.nEventID);
                // 口罩相关属性状态信息
                System.out.println("口罩相关属性状态信息----");
                NET_MASK_ATTRIBUTE  stuMask = msg.stuMask;
                System.out.println("是否有戴口
罩:"+EM_WEARING_STATE.getNoteByValue(stuMask.emHasMask));// 是否有戴口罩,{@link
EM_WEARING_STATE}
                System.out.println("口罩检测结
果:"+EM_COMPLIANCE_STATE.getNoteByValue(stuMask.emHasLegalMask));// 口罩检测结
果,{@link EM_COMPLIANCE_STATE}
                break;
            }
            default:
                System.out.println("其他事件----------" + dwAlarmType);
```

```
                    break;
                }
                return 0;
            }

        }
```

第 13 章 接口说明

13.1 SDK 初始化

13.1.1 SDK 初始化 CLIENT_Init

表13-1 SDK 初始化 CLIENT_Init

选项	说明	
描述	对整个 SDK 进行初始化	
方法	public boolean CLIENT_Init(Callback cbDisConnect, Pointer dwUser);	
参数	[in]cbDisConnect	断线回调函数
	[in]dwUser	断线回调函数的用户参数
返回值	成功返回 TRUE，失败返回 FALSE	
说明	● 调用网络 SDK 其他函数的前提 ● 回调函数设置成 NULL 时，设备断线后不会回调给用户	

13.1.2 SDK 清理 CLIENT_Cleanup

表13-2 SDK 清理 CLIENT_Cleanup

选项	说明
描述	清理 SDK
方法	public void CLIENT_Cleanup();
参数	无
返回值	无
说明	SDK 清理接口，在结束前最后调用

13.1.3 设置断线重连回调函数 CLIENT_SetAutoReconnect

表13-3 设置断线重连回调函数 CLIENT_SetAutoReconnect

选项	说明	
描述	设置自动重连回调函数	
方法	public void CLIENT_SetAutoReconnect(Callback cbAutoConnect, Pointer dwUser);	
参数	[in]cbAutoConnect	断线重连回调函数

选项	说明	
	[in]dwUser	断线重连回调函数的用户参数
返回值	无	
说明	设置断线重连回调接口。如果回调函数设置为 NULL，则不自动重连	

13.1.4 设置网络参数 CLIENT_SetNetworkParam

表13-4 设置网络参数 CLIENT_SetNetworkParam

选项	说明	
描述	设置网络环境相关参数	
方法	public void CLIENT_SetNetworkParam(NET_PARAM pNetParam);	
参数	[in]pNetParam	网络延迟、重连次数、缓存大小等参数
返回值	无	
说明	可根据实际网络环境，调整参数	

13.2 设备登录

13.2.1 用户登录设备 CLIENT_LoginWithHighLevelSecurity

表13-5 用户登录设备 CLIENT_LoginWithHighLevelSecurity

选项	说明	
描述	用户登录设备	
方法	public LLong CLIENT_LoginWithHighLevelSecurity(NET_IN_LOGIN_WITH_HIGHLEVEL_SECURITY pstInParam, NET_OUT_LOGIN_WITH_HIGHLEVEL_SECURITY pstOutParam);	
参数	[in]pstInParam	输入参数
	[out]pstOutParam	输出参数
返回值	成功返回登录句柄，失败返回 0	
说明	此方法封装在 NetSDKLib 接口中，通常通过以下方式调用： CLIENT_LoginWithHighLevelSecurity(pstInParam, pstOutParam);	

13.2.2 用户登出设备 CLIENT_Logout

表13-6 用户登出设备 CLIENT_Logout

选项	说明	
描述	用户登出设备	
方法	public boolean CLIENT_Logout(LLong lLoginID);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
返回值	成功返回 TRUE，失败返回 FALSE	
说明	此方法封装在 NetSDKLib 接口中，通常通过以下方式调用 CLIENT_Logout(m_hLoginHandle);	

13.3 实时预览

13.3.1 打开预览 CLIENT_RealPlayEx

表13-7 打开预览 CLIENT_RealPlayEx

选项	说明	
描述	打开实时预览	
方法	public LLong CLIENT_RealPlayEx(LLong lLoginID, int nChannelID, Pointer hWnd, int rType);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]nChannelID	视频通道号，从 0 开始递增的整数
	[in]hWnd	窗口句柄，仅在 Windows 系统下有效
	[in]rType	预览类型
返回值	成功返回非 0，失败返回 0	
说明	在 Windows 环境下： ● hWnd 为有效值时，在对应窗口显示画面 ● hWnd 为 NULL 时，表示取流方式，通过设置回调函数来获取视频数据，交由用户处理	

预览类型及含义请参见表 13-8。

表13-8 预览类型说明

预览类型	含义
DH_RType_Realplay	实时预览
DH_RType_Multiplay	多画面预览
DH_RType_Realplay_0	实时预览-主码流，等同于 DH_RType_Realplay
DH_RType_Realplay_1	实时预览-辅码流 1
DH_RType_Realplay_2	实时预览-辅码流 2
DH_RType_Realplay_3	实时预览-辅码流 3
DH_RType_Multiplay_1	多画面预览－1 画面
DH_RType_Multiplay_4	多画面预览－4 画面
DH_RType_Multiplay_8	多画面预览－8 画面
DH_RType_Multiplay_9	多画面预览－9 画面
DH_RType_Multiplay_16	多画面预览－16 画面
DH_RType_Multiplay_6	多画面预览－6 画面
DH_RType_Multiplay_12	多画面预览－12 画面
DH_RType_Multiplay_25	多画面预览－25 画面
DH_RType_Multiplay_36	多画面预览－36 画面

13.3.2 关闭预览 CLIENT_StopRealPlayEx

表13-9 关闭预览 CLIENT_StopRealPlayEx

选项	说明	
描述	关闭实时预览	
方法	public boolean CLIENT_StopRealPlayEx(LLong lRealHandle);	
参数	[in]lRealHandle	CLIENT_RealPlayEx 的返回值

选项	说明
返回值	成功返回 TRUE，失败返回 FALSE
说明	无

13.4 智能订阅

13.4.1 开始智能事件订阅 CLIENT_RealLoadPictureEx

表13-10 开始智能事件订阅 CLIENT_RealLoadPictureEx

选项	说明	
描述	开始智能事件订阅	
方法	public LLong CLIENT_RealLoadPictureEx(LLong lLoginID, int nChannelID, int dwAlarmType, int bNeedPicFile, StdCallCallback cbAnalyzerData, Pointer dwUser, Pointer Reserved);	
参数	[in]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[in]nChannelID	设备通道号(从 0 开始)
	[in]dwAlarmType	订阅报警事件类型
	[in]bNeedPicFile	是否订阅图片文件
	[in]cbAnalyzerData	智能事件回调函数
	[in]dwUser	用户自定义数据类型
	[in]Reserved	保留字段
返回值	成功返回 LLONG 类型订阅句柄，失败返回 0	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

智能报警事件类型说明请参见表 13-11。

表13-11 智能事件类型说明

dwAlarmType 宏定义	宏定义值	含义	回调 pAlarmInfo 对应结构体
EVENT_IVS_ALL	0x00000001	所有事件	无
EVENT_IVS_CROSSFENCEDETECTION	0x0000011F	穿越围栏	DEV_EVENT_CROSSFENCEDETECTION_INFO
EVENT_IVS_CROSSLINEDETECTION	0x00000002	绊线入侵	DEV_EVENT_CROSSLINE_INFO
EVENT_IVS_CROSSREGIONDETECTION	0x00000003	区域入侵	DEV_EVENT_CROSSREGION_INFO
EVENT_IVS_LEFTDETECTION	0x00000005	物品遗留	DEV_EVENT_LEFT_INFO
EVENT_IVS_PRESERVATION	0x00000008	物品保全	DEV_EVENT_PRESERVATION_INFO
EVENT_IVS_TAKENAWAYDETECTION	0x00000115	物品搬移	DEV_EVENT_TAKENAWAYDETECTION_INFO
EVENT_IVS_WANDERDETECTION	0x00000007	徘徊事件	DEV_EVENT_WANDER_INFO

dwAlarmType 宏定义	宏定义值	含义	回调 pAlarmInfo 对应结构体
EVENT_IVS_VIDEOABNORMALDETECTION	0x00000013	视频异常	DEV_EVENT_VIDEOABNORMALDETECTION_INFO
EVENT_IVS_AUDIO_ABNORMALDETECTION	0x00000126	声音异常	DEV_EVENT_IVS_AUDIO_ABNORMALDETECTION_INFO
EVENT_IVS_CLIMBDETECTION	0x00000128	攀高检测	DEV_EVENT_IVS_CLIMB_INFO
EVENT_IVS_FIGHTDETECTION	0x0000000E	斗殴检测	DEV_EVENT_FLOWSTAT_INFO
EVENT_IVS_LEAVEDETECTION	0x00000129	离岗检测	DEV_EVENT_IVS_LEAVE_INFO
EVENT_IVS_PSRISEDETECTION	0x0000011E	起身检测	DEV_EVENT_PSRISEDETECTION_INFO
EVENT_IVS_PASTEDETECTION	0x00000004	非法黏贴物贴条检测	DEV_EVENT_PASTE_INFO

13.4.2 停止智能事件订阅 CLIENT_StopLoadPic

表13-12 停止智能事件订阅 CLIENT_StopLoadPic

选项	说明	
描述	停止智能事件订阅	
方法	public boolean CLIENT_StopLoadPic(LLong lAnalyzerHandle);	
参数	[in]lAnalyzerHandle	智能事件订阅句柄
返回值	BOOL 类型 ● 成功：TRUE ● 失败：FALSE	
说明	接口返回失败，请使用 CLIENT_GetLastError 获取错误码	

13.5 订阅客流量统计

13.5.1 客流量事件订阅 CLIENT_AttachVideoStatSummary

表13-13 客流量事件订阅 CLIENT_AttachVideoStatSummary

选项	说明	
描述	客流量事件订阅	
函数	public LLong CLIENT_AttachVideoStatSummary(LLong lLoginID, NET_IN_ATTACH_VIDEOSTAT_SUM pInParam, NET_OUT_ATTACH_VIDEOSTAT_SUM pOutParam, int nWaitTime);	
参数	[in] lLoginID	登录句柄
	[in] pInParam	订阅客流输入参数

选项	说明	
	[out] pOutParam	订阅客流输出参数
	[in] nWaitTime	超时时间
返回值	客流订阅句柄	
说明	无	

13.5.2 取消订阅客流量事件 CLIENT_DetachVideoStatSummary

表13-14 取消订阅客流量事件 CLIENT_DetachVideoStatSummary

选项	说明	
描述	取消订阅客流量事件	
函数	public boolean CLIENT_DetachVideoStatSummary(LLong lAttachHandle);	
参数	[in] lAttachHandle	客流订阅句柄
返回值	成功返回 TRUE，失败返回 FALSE	
说明	无	

第 14 章 回调函数

14.1 注意事项

回调函数建议写成静态单例模式，否则可能会使内存过大致使程序崩溃。

14.2 断线回调函数 fDisConnectCallBack

表14-1 断线回调函数 fDisConnectCallBack

选项	说明	
描述	断线回调函数	

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

参数	[out]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[out]pchDVRIP	断线的设备 IP
	[out]nDVRPort	断线的设备端口
	[out]dwUser	回调函数的用户参数
返回值	无	
说明	无	

14.3 断线重连回调函数 fHaveReConnectCallBack

表14-2 断线重连回调函数 fHaveReConnectCallBack

选项	说明	
描述	断线重连回调函数	

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

参数	[out]lLoginID	CLIENT_LoginWithHighLevelSecurity 的返回值
	[out]pchDVRIP	断线后重连成功的设备 IP
	[out]nDVRPort	断线后重连成功的设备端口
	[out]dwUser	回调函数的用户参数
返回值	无	
说明	无	

14.4 实时预览数据回调函数 fRealDataCallBackEx

表14-3 实时预览数据回调函数 fRealDataCallBackEx

选项	说明	
描述	实时预览数据回调函数	

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

参数	[out]lRealHandle	CLIENT_RealPlayEx 的返回值
	[out]dwDataType	数据类型，0 表示原始数据，2 表示 YUV 数据
	[out]pBuffer	预览数据块地址
	[out]dwBufSize	预览数据块的长度，单位：字节
	[out]param	回调数据参数结构体，dwDataType 值不同类型不同 ● dwDataType 为 0 时，param 为空指针 ● dwDataType 为 2 时，param 为 tagCBYUVDataParam 结构体指针
	[out]dwUser	回调函数的用户参数
返回值	无	
说明	无	

14.5 智能事件回调函数 fAnalyzerDataCallBack

表14-4 智能事件回调函数 fAnalyzerDataCallBack

选项	说明	
描述	远程设备状态回调函数	

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

参数	[out]lAnalyzerHandle	CLIENT_RealLoadPictureEx 返回值
	[out]dwEventType	智能事件类型
	[out] pAlarmInfo	事件信息缓存
	[out]pBuffer	图片缓存
	[out]dwBufSize	图片缓存大小
	[out]dwUser	用户数据
	[out]nSequence	序列号
	[out]reserved	保留
返回值	无	
说明	订阅远程设备智能事件后，如果前端设备有相关的智能事件，会上报发生事件的相关信息	

14.6 客流量事件订阅回调 fVideoStatSumCallBack

表14-5 客流量事件订阅回调 fVideoStatSumCallBack

选项	说明	
描述	客流量事件订阅回调	

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

参数	[out] lAttachHandle	客流订阅句柄
	[out] pBuf	客流返回数据
	[out] dwBufLen	返回数据长度
	[out] dwUser	用户数据
返回值	无	
说明	无	

附录1 智能事件

智能事件名称	序号	含义
EVENT_IVS_ALL	0x00000001	订阅所有事件
EVENT_IVS_CROSSLINEDETECTION	0x00000002	警戒线事件(对应 DEV_EVENT_CROSSLINE_INFO)
EVENT_IVS_CROSSREGIONDETECTION	0x00000003	警戒区事件(对应 DEV_EVENT_CROSSREGION_INFO)
EVENT_IVS_LEFTDETECTION	0x00000005	物品遗留事件(对应 DEV_EVENT_LEFT_INFO)
EVENT_IVS_STAYDETECTION	0x00000006	停留事件(对应 DEV_EVENT_STAY_INFO)
EVENT_IVS_WANDERDETECTION	0x00000007	徘徊事件(对应 DEV_EVENT_WANDER_INFO)
EVENT_IVS_PRESERVATION	0x00000008	物品保全事件(对应 DEV_EVENT_PRESERVATION_INFO)
EVENT_IVS_MOVEDETECTION	0x00000009	移动事件(对应 DEV_EVENT_MOVE_INFO)
EVENT_IVS_TAILDETECTION	0x0000000A	尾随事件(对应 DEV_EVENT_TAIL_INFO)
EVENT_IVS_RIOTERDETECTION	0x0000000B	聚众事件(对应 DEV_EVENT_RIOTERL_INFO)
EVENT_IVS_FIREDETECTION	0x0000000C	火警事件(对应 DEV_EVENT_FIRE_INFO)
EVENT_IVS_SMOKEDETECTION	0x0000000D	烟雾报警事件(对应 DEV_EVENT_SMOKE_INFO)
EVENT_IVS_FIGHTDETECTION	0x0000000E	斗殴事件(对应 DEV_EVENT_FIGHT_INFO)
EVENT_IVS_FLOWSTAT	0x0000000F	流量统计事件(对应 DEV_EVENT_FLOWSTAT_INFO)
EVENT_IVS_NUMBERSTAT	0x00000010	数量统计事件(对应 DEV_EVENT_NUMBERSTAT_INFO)
EVENT_IVS_CAMERACOVERDDETECTION	0x00000011	摄像头覆盖事件(保留)
EVENT_IVS_CAMERAMOVEDDETECTION	0x00000012	摄像头移动事件(保留)
EVENT_IVS_VIDEOABNORMALDETECTION	0x00000013	视频异常事件(对应 DEV_EVENT_VIDEOABNORMALDETECTION_INFO)
EVENT_IVS_VIDEOBADDETECTION	0x00000014	视频损坏事件(保留)
EVENT_IVS_TRAFFICCONTROL	0x00000015	交通管理事件（对应 DEV_EVENT_TRAFFICCONTROL_INFO）

智能事件名称	序号	含义
EVENT_IVS_TRAFFICACCIDENT	0x00000016	交通事故事件(对应 DEV_EVENT_TRAFFICACCIDENT_INFO)
EVENT_IVS_TRAFFICJUNCTION	0x00000017	交通路口事件----老规则(对应 DEV_EVENT_TRAFFICJUNCTION_INFO)
EVENT_IVS_TRAFFICGATE	0x00000018	交通卡口事件----老规则(对应 DEV_EVENT_TRAFFICGATE_INFO)
EVENT_TRAFFICSNAPSHOT	0x00000019	交通抓拍事件（对应 DEV_EVENT_TRAFFICSNAPSHOT_INFO）
EVENT_IVS_FACEDETECT	0x0000001A	目标检测事件(对应 DEV_EVENT_FACEDETECT_INFO)
EVENT_IVS_TRAFFICJAM	0x0000001B	交通拥堵事件(对应 DEV_EVENT_TRAFFICJAM_INFO)
EVENT_IVS_TRAFFIC_NONMOTORINMOTORROUTE	0x0000001C	非机动车占机动车车道事件(对应 DEV_EVENT_TRAFFIC_NONMOTORINMOTORROUTE_INFO)
EVENT_IVS_TRAFFIC_RUNREDLIGHT	0x00000100	交通违章-闯红灯事件(对应 DEV_EVENT_TRAFFIC_RUNREDLIGHT_INFO)
EVENT_IVS_TRAFFIC_OVERLINE	0x00000101	交通违章-压车道线事件(对应 DEV_EVENT_TRAFFIC_OVERLINE_INFO)
EVENT_IVS_TRAFFIC_RETROGRADE	0x00000102	交通违章-逆行事件(对应 DEV_EVENT_TRAFFIC_RETROGRADE_INFO)
EVENT_IVS_TRAFFIC_TURNLEFT	0x00000103	交通违章-违章左转(对应 DEV_EVENT_TRAFFIC_TURNLEFT_INFO)
EVENT_IVS_TRAFFIC_TURNRIGHT	0x00000104	交通违章-违章右转(对应 DEV_EVENT_TRAFFIC_TURNRIGHT_INFO)
EVENT_IVS_TRAFFIC_UTURN	0x00000105	交通违章-违章掉头(对应 DEV_EVENT_TRAFFIC_UTURN_INFO)
EVENT_IVS_TRAFFIC_OVERSPEED	0x00000106	交通违章-超速(对应 DEV_EVENT_TRAFFIC_OVERSPEED_INFO)
EVENT_IVS_TRAFFIC_UNDERSPEED	0x00000107	交通违章-低速(对应 DEV_EVENT_TRAFFIC_UNDERSPEED_INFO)
EVENT_IVS_TRAFFIC_PARKING	0x00000108	交通违章-违章停车(对应 DEV_EVENT_TRAFFIC_PARKING_INFO)
EVENT_IVS_TRAFFIC_WRONGROUTE	0x00000109	交通违章-不按车道行驶(对应 DEV_EVENT_TRAFFIC_WRONGROUTE_INFO)
EVENT_IVS_TRAFFIC_CROSSLANE	0x0000010A	交通违章-违章变道(对应 DEV_EVENT_TRAFFIC_CROSSLANE_INFO)
EVENT_IVS_TRAFFIC_OVERYELLOWLINE	0x0000010B	交通违章-压黄线(对应 DEV_EVENT_TRAFFIC_OVERYELLOWLINE_INFO)
EVENT_IVS_TRAFFIC_DRIVINGONSHOULDER	0x0000010C	交通违章-路肩行驶事件（对应 EVENT_IVS_TRAFFIC_DRIVINGONSHOULDER）

智能事件名称	序号	含义
EVENT_IVS_TRAFFIC_YELLOWPLATEINLANE	0x0000010E	交通违章-黄牌车占道事件(对应 DEV_EVENT_TRAFFIC_YELLOWPLATEINLANE_INFO)
EVENT_IVS_TRAFFIC_PEDESTRAINPRIORITY	0x0000010F	交通违章-礼让行人/斑马线行人优先事件(对应 DEV_EVENT_TRAFFIC_PEDESTRAINPRIORITY_INFO)
EVENT_IVS_TRAFFIC_NOPASSING	0x00000111	交通违章-禁止通行事件(对应 DEV_EVENT_TRAFFIC_NOPASSING_INFO)
EVENT_IVS_ABNORMALRUNDETECTION	0x00000112	异常奔跑事件(对应 DEV_EVENT_ABNORMALRUNDETECTION_INFO)
EVENT_IVS_RETROGRADEDETECTION	0x00000113	人员逆行事件(对应 DEV_EVENT_RETROGRADEDETECTION_INFO)
EVENT_IVS_TAKENAWAYDETECTION	0x00000115	物品搬移事件(对应 DEV_EVENT_TAKENAWAYDETECTION_INFO)
EVENT_IVS_PARKINGDETECTION	0x00000116	非法停车事件(对应 DEV_EVENT_PARKINGDETECTION_INFO)
EVENT_IVS_FACERECOGNITION	0x00000117	目标识别事件(对应 DEV_EVENT_FACERECOGNITION_INFO)
EVENT_IVS_TRAFFIC_MANUALSNAP	0x00000118	交通手动抓拍事件(对应 DEV_EVENT_TRAFFIC_MANUALSNAP_INFO)
EVENT_IVS_TRAFFIC_FLOWSTATE	0x00000119	交通流量统计事件(对应 DEV_EVENT_TRAFFIC_FLOW_STATE)
EVENT_IVS_TRAFFIC_STAY	0x0000011A	交通滞留事件（对应 DEV_EVENT_TRAFFIC_STAY_INFO）
EVENT_IVS_TRAFFIC_VEHICLEINROUTE	0x0000011B	有车占道事件(对应 DEV_EVENT_TRAFFIC_VEHICLEINROUTE_INFO)
EVENT_ALARM_MOTIONDETECT	0x0000011C	视频移动侦测事件(对应 DEV_EVENT_ALARM_INFO)
EVENT_ALARM_LOCALALARM	0x0000011D	外部报警事件(对应 DEV_EVENT_ALARM_INFO)
EVENT_IVS_PSRISEDETECTION	0x0000011E	囚犯起身事件(对应 DEV_EVENT_PSRISEDETECTION_INFO)
EVENT_IVS_TRAFFIC_TOLLGATE	0x00000120	交通违章-卡口事件----新规则(对应 DEV_EVENT_TRAFFICJUNCTION_INFO)
EVENT_IVS_DENSITYDETECTION	0x00000121	人员密集度检测(对应 DEV_EVENT_DENSITYDETECTION_INFO)
EVENT_IVS_VIDEODIAGNOSIS	0x00000122	视频诊断结果事件(对应 NET_VIDEODIAGNOSIS_COMMON_INFO 和 NET_REAL_DIAGNOSIS_RESULT)
EVENT_IVS_TRAFFIC_VEHICLEINBUSROUTE	0x00000124	占用公交车道事件(对应 DEV_EVENT_TRAFFIC_VEHICLEINBUSROUTE_INFO)

智能事件名称	序号	含义
EVENT_IVS_TRAFFIC_BACKING	0x00000125	违章倒车事件(对应 DEV_EVENT_IVS_TRAFFIC_BACKING_INFO)
EVENT_IVS_AUDIO_ABNORMALDETECTION	0x00000126	声音异常检测(对应 DEV_EVENT_IVS_AUDIO_ABNORMALDETECTION_INFO)
EVENT_IVS_TRAFFIC_RUNYELLOWLIGHT	0x00000127	交通违章-闯黄灯事件(对应 DEV_EVENT_TRAFFIC_RUNYELLOWLIGHT_INFO)
EVENT_IVS_CLIMBDETECTION	0x00000128	攀高检测事件(对应 DEV_EVENT_IVS_CLIMB_INFO)
EVENT_IVS_LEAVEDETECTION	0x00000129	离岗检测事件(对应 DEV_EVENT_IVS_LEAVE_INFO)
EVENT_IVS_TRAFFIC_PARKINGONYELLOWBOX	0x0000012A	黄网格线抓拍事件(对应 DEV_EVENT_TRAFFIC_PARKINGONYELLOWBOX_INFO)
EVENT_IVS_TRAFFIC_PARKINGSPACEPARKING	0x0000012B	车位有车事件(对应 DEV_EVENT_TRAFFIC_PARKINGSPACEPARKING_INFO)
EVENT_IVS_TRAFFIC_PARKINGSPACENOPARKING	0x0000012C	车位无车事件(对应 DEV_EVENT_TRAFFIC_PARKINGSPACENOPARKING_INFO)
EVENT_IVS_TRAFFIC_PEDESTRAIN	0x0000012D	交通行人事件(对应 DEV_EVENT_TRAFFIC_PEDESTRAIN_INFO)
EVENT_IVS_TRAFFIC_THROW	0x0000012E	交通抛洒物品事件(对应 DEV_EVENT_TRAFFIC_THROW_INFO)
EVENT_IVS_TRAFFIC_IDLE	0x0000012F	交通空闲事件（对应 DEV_EVENT_TRAFFIC_IDLE_INFO）
EVENT_ALARM_VEHICLE_TURNOVER	0x00000131	车辆侧翻报警事件（对应 DEV_EVENT_VEHICEL_ALARM_INFO）
EVENT_ALARM_VEHICLE_COLLISION	0x00000132	车辆撞车报警事件（对应 DEV_EVENT_VEHICEL_ALARM_INFO）
EVENT_ALARM_VEHICLE_LARGE_ANGLE	0x00000133	车载摄像头大角度扭转事件
EVENT_IVS_TRAFFIC_PARKINGSPACEOVERLINE	0x00000134	车位压线事件(对应 DEV_EVENT_TRAFFIC_PARKINGSPACEOVERLINE_INFO)
EVENT_IVS_MULTISCENESWITCH	0x00000135	多场景切换事件（对应 DEV_EVENT_IVS_MULTI_SCENE_SWICH_INFO）
EVENT_IVS_TRAFFIC_RESTRICTED_PLATE	0x00000136	受限车牌事件（对应 DEV_EVENT_TRAFFIC_RESTRICTED_PLATE）
EVENT_IVS_TRAFFIC_OVERSTOPLINE	0x00000137	压停止线事件(对应 DEV_EVENT_TRAFFIC_OVERSTOPLINE)
EVENT_IVS_TRAFFIC_WITHOUT_SAFEBELT	0x00000138	交通未系安全带事件(对应 DEV_EVENT_TRAFFIC_WITHOUT_SAFEBELT)

智能事件名称	序号	含义
EVENT_IVS_TRAFFIC_DRIVER_SMOKING	0x00000139	驾驶员抽烟事件(对应 DEV_EVENT_TRAFFIC_DRIVER_SMOKING)
EVENT_IVS_TRAFFIC_DRIVER_CALLING	0x0000013A	驾驶员打电话事件(对应 DEV_EVENT_TRAFFIC_DRIVER_CALLING)
EVENT_IVS_TRAFFIC_PEDESTRAINRUNREDLIGHT	0x0000013B	行人闯红灯事件(对应 DEV_EVENT_TRAFFIC_PEDESTRAINRUNREDLIGHT_INFO)
EVENT_IVS_TRAFFIC_PASSNOTINORDER	0x0000013C	未按规定依次通行(对应 DEV_EVENT_TRAFFIC_PASSNOTINORDER_INFO)
EVENT_IVS_OBJECT_DETECTION	0x00000141	物体特征检测事件
EVENT_ALARM_ANALOGALARM	0x00000150	模拟量报警通道的报警事件(对应 DEV_EVENT_ALARM_ANALOGALRM_INFO)
EVENT_IVS_CROSSLINEDETECTION_EX	0x00000151	警戒线扩展事件（对应 DEV_EVENT_CROSSLINE_INFO_EX ）
EVENT_ALARM_VIDEOBLIND	0x00000153	视频遮挡事件(对应 DEV_EVENT_ALARM_VIDEOBLIND)
EVENT_ALARM_VIDEOLOSS	0x00000154	视频丢失事件
EVENT_IVS_GETOUTBEDDETECTION	0x00000155	下床事件（对应 DEV_EVENT_GETOUTBED_INFO）
EVENT_IVS_PATROLDETECTION	0x00000156	巡逻检测事件（对应 DEV_EVENT_PATROL_INFO）
DEV_EVENT_ONDUTY_INFO	0x00000157	站岗检测事件（对应 DEV_EVENT_ONDUTY_INFO）
EVENT_IVS_NOANSWERCALL	0x00000158	门口机呼叫未响应事件
EVENT_IVS_STORAGENOTEXIST	0x00000159	存储组不存在事件
EVENT_IVS_STORAGELOWSPACE	0x0000015A	硬盘空间低报警事件
EVENT_IVS_STORAGEFAILURE	0x0000015B	存储错误事件
EVENT_IVS_PROFILEALARMTRANSMIT	0x0000015C	报警传输事件（对应 DEV_EVENT_PROFILE_ALARM_TRANSMIT_INFO）
EVENT_IVS_VIDEOSTATIC	0x0000015D	视频静态检测事件（对应 DEV_EVENT_ALARM_VIDEOSTATIC_INFO）
EVENT_IVS_VIDEOTIMING	0x0000015E	视频定时检测事件（对应 DEV_EVENT_ALARM_VIDEOTIMING_INFO）
EVENT_IVS_HEATMAP	0x0000015F	热度图（对应 CFG_IVS_HEATMAP_INFO）
EVENT_IVS_CITIZENIDCARD	0x00000160	证件信息读取事件（对应 DEV_EVENT_ALARM_CITIZENIDCARD_INFO）

智能事件名称	序号	含义
EVENT_IVS_PICINFO	0x00000161	图片信息事件（对应 DEV_EVENT_ALARM_PIC_INFO）
EVENT_IVS_NETPLAYCHECK	0x00000162	上网登记事件（对应 DEV_EVENT_ALARM_NETPLAYCHECK_INFO）
EVENT_IVS_TRAFFIC_JAM_FORBID_INTO	0x00000163	车辆拥堵禁入事件(对应 DEV_EVENT_ALARM_JAMFORBIDINTO_INFO)
EVENT_IVS_SNAPBYTIME	0x00000164	定时抓图事件(对应 DEV_EVENT_SNAPBYTIME)
EVENT_IVS_PTZ_PRESET	0x00000165	云台转动到预置点事件（对应 DEV_EVENT_ALARM_PTZ_PRESET_INFO）
EVENT_IVS_RFID_INFO	0x00000166	红外线检测信息事件（对应 DEV_EVENT_ALARM_RFID_INFO）
EVENT_IVS_RFID_INFO	0x00000167	人起立检测事件
EVENT_IVS_QSYTRAFFICCARWEIGHT	0x00000168	交通卡口称重事件(对应 DEV_EVENT_QSYTRAFFICCARWEIGHT_INFO)
EVENT_IVS_TRAFFIC_COMPAREPLATE	0x00000169	卡口前后车牌合成事件（对应 DEV_EVENT_TRAFFIC_COMPAREPLATE_INFO）
EVENT_IVS_SHOOTINGSCORERECOGNITION	0x0000016A	打靶像机事件（对应 DEV_EVENT_SHOOTING_SCORE_RECOGNITION_INFO,CFG_IVS_SHOOTINGSCORERECOGNITION_INFO）
EVENT_IVS_TRAFFIC_FCC	0x0000016B	加油站提枪、挂枪事件（对应 DEV_EVENT_TRAFFIC_FCC_INFO）
EVENT_IVS_TRAFFIC_TRANSFINITE	0x0000016C	违章超限抓图上报事件（对应 DEV_EVENT_TRAFFIC_TRANSFINITE_INFO）
EVENT_IVS_SCENE_CHANGE	0x0000016D	场景变更事件(对应 DEV_ALRAM_SCENECHANGE_INFO)
EVENT_IVS_LETRACK	0x0000016E	简单跟踪事件(暂未有具体事件)
EVENT_IVS_OBJECT_ACTION	0x0000016F	物体检测事件(暂未有具体事件)
EVENT_IVS_TRAFFIC_ANALYSE_PRESNAP	0x00000170	预分析抓拍图片事件（物体检测事件(暂未有具体事件)）
EVENT_ALARM_EQSTATE	0x00000171	智能插座电量状态上报(暂未有具体事件)
EVENT_IVS_ALARM_IPC	0x00000172	DVR/NVR 设备上的 IPC 报警（对应 DEV_EVENT_ALARM_IPC_INFO）
EVENT_IVS_POS_RECORD	0x00000173	POS 录像查询事件(暂未有具体事件)
EVENT_IVS_NEAR_DISTANCE_DETECTION	0x00000174	近距离接触事件（对应 DEV_EVENT_NEAR_DISTANCE_DETECTION_INFO）
EVENT_IVS_OBJECTSTRUCTLIZE_PERSON	0x00000175	行人特征检测事件（对应 DEV_EVENT_OBJECTSTRUCTLIZE_PERSON_INFO）
EVENT_IVS_OBJECTSTRUCTLIZE_NONMOTOR	0x00000176	非机动车特征检测事件（对应 DEV_EVENT_OBJECTSTRUCTLIZE_NONMOTOR_INFO）

智能事件名称	序号	含义
EVENT_IVS_TUMBLE_DETECTION	0x00000177	倒地报警事件(对应 DEV_EVENT_TUMBLE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_ALL	0x000001FF	所有以 traffic 开头的事件
EVENT_IVS_VIDEOANALYSE	0x00000200	所有智能分析事件
EVENT_IVS_LINKSD	0x00000201	LinkSD 事件（对应 DEV_EVENT_LINK_SD）
EVENT_IVS_VEHICLEANALYSE	0x00000202	车辆特征检测分析（对应 DEV_EVENT_VEHICLEANALYSE）
EVENT_IVS_FLOWRATE	0x00000203	流量使用情况事件（对应 DEV_EVENT_FLOWRATE_INFO）
EVENT_IVS_ACCESS_CTL	0x00000204	门禁事件(对应 DEV_EVENT_ACCESS_CTL_INFO)
EVENT_IVS_SNAPMANUAL	0x00000205	SnapManual 事件(对应 DEV_EVENT_SNAPMANUAL)
EVENT_IVS_TRAFFIC_ELETAGINFO	0x00000206	RFID 电子车牌标签事件(对应 DEV_EVENT_TRAFFIC_ELETAGINFO_INFO)
EVENT_IVS_TRAFFIC_TIREDPHYSIOLOGICAL	0x00000207	生理疲劳驾驶事件(对应 DEV_EVENT_TIREDPHYSIOLOGICAL_INFO)
EVENT_IVS_TRAFFIC_BUSSHARPTURN	0x00000208	EVENT_IVS_TRAFFIC_BUSSHARPTURN（对应 DEV_EVENT_BUSSHARPTURN_INFO）
EVENT_IVS_CITIZEN_PICTURE_COMPARE	0x00000209	人证比对事件(对应 DEV_EVENT_CITIZEN_PICTURE_COMPARE_INFO)
EVENT_IVS_TRAFFIC_TIREDLOWERHEAD	0x0000020A	开车低头报警事件(对应 DEV_EVENT_TIREDLOWERHEAD_INFO)
EVENT_IVS_TRAFFIC_DRIVERLOOKAROUND	0x0000020B	开车左顾右盼报警事件(对应 DEV_EVENT_DRIVERLOOKAROUND_INFO)
EVENT_IVS_TRAFFIC_DRIVERLEAVEPOST	0x0000020C	开车离岗报警事件(对应 DEV_EVENT_DRIVERLEAVEPOST_INFO)
EVENT_IVS_MAN_STAND_DETECTION	0x0000020D	立体视觉站立事件(对应 DEV_EVENT_MANSTAND_DETECTION_INFO)
EVENT_IVS_MAN_NUM_DETECTION	0x0000020E	立体视觉区域内人数统计事件(对应 DEV_EVENT_MANNUM_DETECTION_INFO)
EVENT_IVS_STEREO_NUMBERSTAT	0x0000020F	客流量统计事件（暂未有具体事件）
EVENT_IVS_TRAFFIC_DRIVERYAWN	0x00000210	开车打哈欠事件(对应 DEV_EVENT_DRIVERYAWN_INFO)
EVENT_IVS_NUMBERSTAT_PLAN	0x00000211	客流量统计计划(暂未有具体事件,球机使用,对应规则配置结构体 CFG_NUMBERSTAT_INFO)
EVENT_IVS_HEATMAP_PLAN	0x00000212	热度图计划(暂未有具体事件,球机使用,对应规则配置结构体 CFG_IVS_HEATMAP_INFO)
呼叫无答应事件	0x00000213	呼叫无答应事件

智能事件名称	序号	含义
EVENT_IVS_IGNOREINVITE	0x00000214	无视邀请事件
EVENT_IVS_HUMANTRAIT	0x00000215	人体特征事件(对应 DEV_EVENT_HUMANTRAIT_INFO)
EVENT_ALARM_LE_HEADDETECTION	0x00000216	乐橙人头检测事件
EVENT_IVS_FACEANALYSIS	0x00000217	人脸分析事件(暂未有具体事件)
EVENT_IVS_TRAFFIC_TURNLEFTAFTERSTRAIGHT	0x00000218	左转不礼让直行事件（对应 DEV_EVENT_TURNLEFTAFTERSTRAIGHT_INFO）
EVENT_IVS_TRAFFIC_BIGBENDSMALLTURN	0x00000219	大弯小转事件（对应 DEV_EVENT_BIGBENDSMALLTURN_INFO）
EVENT_IVS_ROAD_BLOCK	0x0000021A	道路施工监测事件（对应 DEV_EVENT_ROAD_CONSTRUCTION_INFO）
EVENT_IVS_ROAD_BLOCK	0x0000021B	路障检测事件（对应 DEV_EVENT_ROAD_BLOCK_INFO）
EVENT_IVS_TRAFFIC_QUEUEJUMP	0x0000021C	车辆加塞事件(对应 DEV_EVENT_TRAFFIC_QUEUEJUMP_INFO)
EVENT_IVS_VEHICLE_SUSPICIOUSCAR	0x0000021D	嫌疑车辆事件（对应 DEV_EVENT_VEHICLE_SUSPICIOUSCAR_INFO）
EVENT_IVS_TRAFFIC_TURNRIGHTAFTERSTRAIGHT	0x0000021E	右转不礼让直行事件（对应 DEV_EVENT_TURNRIGHTAFTERSTRAIGHT_INFO ）
EVENT_IVS_TRAFFIC_TURNRIGHTAFTERPEOPLE	0x0000021F	右转不礼让直行行人（对应 DEV_EVENT_TURNRIGHTAFTERPEOPLE_INFO）
EVENT_IVS_INSTALL_CARDREADER	0x00000220	安装读卡器事件（对应 DEV_EVENT_INSTALL_CARDREADER_INFO）
EVENT_ALARM_YALE_DROPBOX_BADTOKEN	0x00000221	Yale token 失效事件，只用于订阅手机推送
EVENT_IVS_ACC_OFF_SNAP	0x00000222	车载设备断电前抓拍上传事件（对应 DEV_EVENT_ACC_OFF_SNAP_INFO）
EVENI_IVS_XRAY_DETECTION	0x00000223	X 光检测事件(对应 DEV_EVENT_XRAY_DETECTION_INFO)
EVENT_IVS_NOTCLEARCAR	0x00000224	未清车告警（对应 DEV_EVENT_NOTCLEARCAR_INFO）
EVENT_IVS_SOSALEART	0x00000225	sos 求救报警(对应 DEV_EVENT_SOSALEART_INFO)
EVENT_IVS_OVERLOAD	0x00000226	超载抓图(对应 DEV_EVENT_OVERLOAD_INFO)
EVENT_IVS_NONWORKINGTIME	0x00000227	非工作时间告警(对应 DEV_EVENT_NONWORKINGTIME_INFO)
EVENT_IVS_TRAFFIC_HIGH_BEAM	0x00000228	远光灯违章事件（对应 DEV_EVENT_TRAFFIC_HIGH_BEAM_INFO）

智能事件名称	序号	含义
EVENT_IVS_TRAFFIC_TRUCKFORBID	0x00000229	禁止货车事件(对应 DEV_EVENT_TRAFFICTRUCKFORBID_INFO)
EVENT_IVS_DRIVINGWITHOUTCARD	0x0000022A	无卡驾驶报警事件（对应 DEV_EVENT_DRIVINGWITHOUTCARD_INFO）
EVENT_IVS_HIGHSPEED	0x0000022B	车辆超速报警事件(对应 DEV_EVENT_HIGHSPEED_INFO)
EVENT_IVS_CROWDDETECTION	0x0000022C	人群密度检测事件(对应结构体 DEV_EVENT_CROWD_DETECTION_INFO)
EVENT_IVS_TRAFFIC_CARDISTANCESHORT	0x0000022D	车间距过小报警事件(对应 DEV_EVENT_TRAFFIC_CARDISTANCESHORT_INFO)
EVENT_IVS_PEDESTRIAN_JUNCTION	0x00000230	行人卡口事件(对应 DEV_EVENT_PEDESTRIAN_JUNCTION_INFO)
EVENT_IVS_VEHICLE_RECOGNITION	0x00000231	车牌对比事件(对应 DEV_EVENT_VEHICLE_RECOGNITION_INFO)
EVENT_IVS_PASS_CHANGE	0x00000232	预置点图片变化事件（对应 DEV_EVENT_PASS_CHANGE_INFO）
EVENT_IVS_TRAFFIC_PARKING_SPACEDETECTION	0x00000233	违停相机单球车位检测规则事件
EVENT_IVS_TRAFFIC_WAITINGAREA	0x00000234	违章进入待行区事件（对应 DEV_EVENT_TRAFFIC_WAITINGAREA_INFO）
EVENT_IVS_TRAFFIC_BAN	0x00000235	机动车违法禁令标识事件(对应 DEV_EVENT_TRAFFIC_BAN_INFO)
EVENT_IVS_POS_EXCHANGE	0x00000236	POS 机交易事件（对应 DEV_EVENT_POS_EXCHANGE_INFO）
EVENT_IVS_STEREO_FIGHTDETECTION	0x00000237	立体行为分析打架/剧烈运动检测规则（仅用于规则配置，对应事件 EVENT_IVS_FIGHTDETECTION）
EVENT_IVS_STEREO_DISTANCE_DETECTION	0x00000238	立体行为分析间距异常/人员靠近检测（仅用于规则配置，对应事件）
EVENT_IVS_STEREO_STEREOFALLDETECTION	0x00000239	立体行为分析跌倒检测规则（仅用于规则配置，对应事件 EVENT_IVS_TUMBLE_DETECTION）
EVENT_IVS_STEREO_STAYDETECTION	0x0000023A	立体行为分析人员滞留检测规则（仅用于规则配置，对应事件 EVENT_IVS_STAYDETECTION）
EVENT_IVS_BANNER_DETECTION	0x0000023B	拉横幅事件(对应 DEV_EVENT_BANNER_DETECTION_INFO)
EVENT_IVS_NORMAL_FIGHTDETECTION	0x0000023C	普通打架事件(只用于普通打架规则,事件采用 EVENT_IVS_FIGHTDETECTION)
EVENT_IVS_ELEVATOR_ABNORMAL	0x0000023D	电动扶梯运行异常事件(对应 DEV_EVENT_ELEVATOR_ABNORMAL_INFO)

智能事件名称	序号	含义
EVENT_IVS_NONMOTORDETECT	0x0000023E	非机动车检测（对应结构体 DEV_EVENT_NONMOTORDETECT_INFO）
EVENT_IVS_VEHICLEDETECT	0x0000023F	机动车检测（只用于规则配置，事件采用 EVENT_IVS_TRAFFICJUNCTION）
EVENT_IVS_TRAFFIC_PARKING_B	0x00000240	交通违章-B 类违章停车(对应 DEV_EVENT_TRAFFIC_PARKING_B_INFO)
EVENT_IVS_TRAFFIC_PARKING_C	0x00000241	交通违章-C 类违章停车(对应 DEV_EVENT_TRAFFIC_PARKING_C_INFO)
EVENT_IVS_TRAFFIC_PARKING_D	0x00000242	交通违章-D 类违章停车(对应 DEV_EVENT_TRAFFIC_PARKING_D_INFO)
EVENT_IVSS_FACEATTRIBUTE	0x00000243	IVSS 目标检测事件 (暂未有具体事件)
EVENT_IVSS_FACECOMPARE	0x00000244	IVSS 目标识别事件 (暂未有具体事件)
EVENT_IVS_FIREWARNING	0x00000245	火警事件(对应 DEV_EVENT_FIREWARNING_INFO)
EVENT_IVS_SHOPPRESENCE	0x00000246	商铺占道经营事件(对应 DEV_EVENT_SHOPPRESENCE_INFO)
EVENT_IVS_WASTEDUMPED	0x00000247	垃圾违章倾倒事件（对应 DEV_EVENT_WASTEDUMPED_INFO）
EVENT_IVS_SPILLEDMATERIAL_DETECTION	0x00000248	抛洒物检测事件（对应 DEV_EVENT_SPILLEDMATERIAL_DETECTION_INFO）
EVENT_IVS_STEREO_MANNUM_DETECTION	0x00000249	立体行为分析人数异常检测（仅用于规则配置，对应事件 EVENT_IVS_MAN_NUM_DETECTION）
EVENT_IVS_DISTANCE_DETECTION	0x0000024A	异常间距事件(对应 DEV_EVENT_DISTANCE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_OVERLOAD	0x0000024B	非机动车超载事件(对应 DEV_EVENT_TRAFFIC_NONMOTOR_OVERLOAD_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_WITHOUTSAFEHAT	0x0000024C	非机动车未戴安全帽事件(对应 DEV_EVENT_TRAFFIC_NONMOTOR_WITHOUTSAFEHAT_INFO)
EVENT_IVS_TRAFFIC_JAM_STOP_ON_ZEBRACROSSING	0x0000024D	拥堵滞留斑马线事件（对应 DEV_EVENT_TRAFFIC_JAM_STOP_ON_ZEBRACROSSING_INFO）
EVENT_IVS_FLOWBUSINESS	0x0000024E	流动摊贩事件(对应 DEV_EVENT_FLOWBUSINESS_INFO)
EVENT_IVS_CITY_MOTORPARKING	0x0000024F	城市机动车违停事件(对应 DEV_EVENT_CITY_MOTORPARKING_INFO)
EVENT_IVS_CITY_NONMOTORPARKING	0x00000250	城市机非动车违停事件(对应 DEV_EVENT_CITY_NONMOTORPARKING_INFO)
EVENT_IVS_LANEDEPARTURE_WARNNING	0x00000251	车道偏移预警(对应 DEV_EVENT_LANEDEPARTURE_WARNNING_INFO)

智能事件名称	序号	含义
EVENT_IVS_FORWARDCOLLISION_WARNNING	0x00000252	前向碰撞预警(对应 DEV_EVENT_FORWARDCOLLISION_WARNNING_INFO)
EVENT_IVS_MATERIALSSTAY	0x00000253	物料堆放事件（对应 DEV_EVENT_MATERIALSSTAY_INF）
EVENT_IVS_TRAFFIC_NONMOTOR_HOLDUMBRELLA	0x00000254	非机动车装载伞具(对应 DEV_EVENT_TRAFFIC_NONMOTOR_HOLDUMBRELLA_INFO)
EVENT_IVS_JABLOTRON_ALARM	0x00000255	客户报警产品
EVENT_IVS_VIDEOUNFOCUS_ALARM	0x00000256	视频虚焦事件（对应 DEV_EVENT_VIDEOUNFOCUS_INFO）
EVENT_IVS_FLOATINGOBJECT_DETECTION	0x00000257	漂浮物检测事件(对应 DEV_EVENT_FLOATINGOBJECT_DETECTION_INFO)
EVENT_IVS_SHIP_DETECTION	0x00000258	船舶检测事件（对应 DEV_EVENT_SHIP_DETECTION_INFO）
EVENT_IVS_AIRPLANE_DETECTION	0x00000259	飞机行为检测事件（对应 DEV_EVENT_AIRPLANE_DETECTION_INFO）
EVENT_IVS_PHONECALL_DETECT	0x0000025A	打电话检测事件(对应 DEV_EVENT_PHONECALL_DETECT_INFO)
EVENT_IVS_SMOKING_DETECT	0x0000025B	吸烟检测事件(对应 DEV_EVENT_SMOKING_DETECT_INFO)
EVENT_IVS_RADAR_SPEED_LIMIT_ALARM	0x0000025C	雷达限速报警事件(对应 DEV_EVENT_RADAR_SPEED_LIMIT_ALARM_INFO)
EVENT_IVS_WATER_LEVEL_DETECTION	0x0000025D	水位检测事件(对应 DEV_EVENT_WATER_LEVEL_DETECTION_INFO)
EVENT_IVS_HOLD_UMBRELLA	0x0000025E	违规撑伞检测事件(对应 DEV_EVENT_HOLD_UMBRELLA_INFO)
EVENT_IVS_GARBAGE_EXPOSURE	0x0000025F	垃圾暴露检测事件(对应 DEV_EVENT_GARBAGE_EXPOSURE_INFO)
EVENT_IVS_DUSTBIN_OVER_FLOW	0x00000260	垃圾桶满溢检测事件(对应 DEV_EVENT_DUSTBIN_OVER_FLOW_INFO)
EVENT_IVS_DOOR_FRONT_DIRTY	0x00000261	门前脏乱检测事件(对应 DEV_EVENT_DOOR_FRONT_DIRTY_INFO)
EVENT_IVS_QUEUESTAY_DETECTION	0x00000262	排队滞留时间报警事件（对应 DEV_EVENT_QUEUESTAY_DETECTION_INFO）
EVENT_IVS_QUEUENUM_DETECTION	0x00000263	排队人数异常报警事件（对应 DEV_EVENT_QUEUENUM_DETECTION_INFO）
EVENT_IVS_GENERATEGRAPH_DETECTION	0x00000264	生成图规则事件（对应 DEV_EVENT_GENERATEGRAPH_DETECTION_INFO）

智能事件名称	序号	含义
EVENT_IVS_TRAFFIC_PARKING_MANUAL	0x00000265	交通违章-手动取证(对应 DEV_EVENT_TRAFFIC_PARKING_MANUAL_INFO)
EVENT_IVS_HELMET_DETECTION	0x00000266	安全帽检测事件(对应 DEV_EVENT_HELMET_DETECTION_INFO)
EVENT_IVS_DEPOSIT_DETECTION	0x00000267	包裹堆积程度检测事件（对应 DEV_EVENT_DEPOSIT_DETECTION_INFO）
EVENT_IVS_HOTSPOT_WARNING	0x00000268	热点异常报警事件(对应 DEV_EVENT_HOTSPOT_WARNING_INFO)
EVENT_IVS_WEIGHING_PLATFORM_DETECTION	0x00000269	称重平台检测事件(对应 DEV_EVENT_WEIGHING_PLATFORM_DETECTION_INFO)
EVENT_IVS_CLASSROOM_BEHAVIOR	0x0000026A	课堂行为分析事件(对应 DEV_EVENT_CLASSROOM_BEHAVIOR_INFO)
EVENT_IVS_VEHICLE_DISTANCE_NEAR	0x0000026B	安全驾驶车距过近报警事件(对应 DEV_EVENT_VEHICLE_DISTANCE_NEAR_INFO)
EVENT_IVS_TRAFFIC_DRIVER_ABNORMAL	0x0000026C	驾驶员异常报警事件(对应 DEV_EVENT_TRAFFIC_DRIVER_ABNORMAL_INFO)
EVENT_IVS_TRAFFIC_DRIVER_CHANGE	0x0000026D	驾驶员变更报警事件（对应 DEV_EVENT_TRAFFIC_DRIVER_CHANGE_INFO）
EVENT_IVS_WORKCLOTHES_DETECT	0x0000026E	工装(安全帽/工作服等)检测事件(对应 DEV_EVENT_WORKCLOTHES_DETECT_INFO)
EVENT_IVS_SECURITYGATE_PERSONALARM	0x0000026F	安检门人员报警事件(对应 DEV_EVENT_SECURITYGATE_PERSONALARM_INFO)
EVENT_IVS_STAY_ALONE_DETECTION	0x00000270	单人独处事件(对应 DEV_EVENT_STAY_ALONE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_ROAD_BLOCK	0x00000271	交通路障检测事件(对应 DEV_EVENT_TRAFFIC_ROAD_BLOCK_INFO)
EVENT_IVS_TRAFFIC_ROAD_CONSTRUCTION	0x00000272	交通道路施工检测事件(对应 DEV_EVENT_TRAFFIC_ROAD_CONSTRUCTION_INFO)
EVENT_IVS_WORKSTATDETECTION	0x00000274	作业统计事件(对应 DEV_EVENT_WORKSTATDETECTION_INFO)
EVENT_IVS_INFRAREDBLOCK	0x00000275	红外阻断事件（对应 DEV_EVENT_INFRAREDBLOCK_INFO）
EVENT_IVS_FEATURE_ABSTRACT	0x00000276	特征提取事件(对应 DEV_EVENT_FEATURE_ABSTRACT_INFO)
EVENT_IVS_INTELLI_SHELF	0x00000277	智能补货事件(对应 DEV_EVENT_INTELLI_SHELF_INFO)
EVENT_IVS_PANORAMA_SHOT	0x00000278	全景抓拍事件（对应 DEV_EVENT_PANORAMA_SHOT_INFO）

智能事件名称	序号	含义
EVENT_ALARM_SMARTMOTION_HUMAN	0x00000279	智能视频移动侦测事件(人),(对应 DEV_EVENT_SMARTMOTION_HUMAN_INFO)
EVENT_ALARM_SMARTMOTION_VEHICLE	0x0000027A	智能视频移动侦测事件(车),(对应 DEV_EVENT_SMARTMOTION_VEHICLE_INFO)
EVENT_IVS_CAR_DRIVING_IN_OUT	0x0000027B	车辆驶入驶出状态事件(对应 DEV_EVENT_CAR_DRIVING_IN_OUT_INFO)
EVENT_IVS_PARKINGSPACE_STATUS	0x0000027C	停车位状态事件（对应 DEV_EVENT_PARKINGSPACE_STATUS_INFO）
EVENT_IVS_VIOLENT_THROW_DETECTION	0x0000027D	暴力抛物检测(对应 DEV_EVENT_VIOLENT_THROW_DETECTION_INFO)
EVENT_IVS_TRAMCARSECTIONS_DETECTION	0x0000027E	矿车超挂报警事件（对应 DEV_EVENT_TRAMCARSECTIONS_DETECTION_INFO）
EVENT_IVS_ALARM_BOX_ALARM	0x0000027F	报警盒通道的触发报警事件（目前只用于订阅手机推送）
EVENT_IVS_FACE_COMPARISION	0x00000280	人脸比对事件（专用于手机推送的目标识别事件，目前只用于订阅手机推送）
EVENT_IVS_FACEBODY_DETECT	0x00000281	人像检测事件（对应 DEV_EVENT_FACEBODY_DETECT_INFO）
EVENT_IVS_FACEBODY_ANALYSE	0x00000282	人像识别事件（对应 DEV_EVENT_FACEBODY_ANALYSE_INFO）
EVENT_IVS_GASSTATION_VEHICLE_DETECT	0x00000283	加油站车辆检测事件(对应 DEV_EVENT_GASSTATION_VEHICLE_DETECT_INFO)
EVENT_IVS_CONGESTION_DETECTION	0x00000284	道路场景车辆拥堵报警事件（对应 DEV_EVENT_CONGESTION_DETECTION_INFO）
EVENT_IVS_VEHICLELIMIT_DETECTION	0x00000285	停车场场景下停车车辆上限报警（对应 DEV_EVENT_VEHICLELIMIT_DETECTION_INFO）
EVENT_IVS_ANIMAL_DETECTION	0x00000286	动物检测事件,(对应 DEV_EVENT_ANIMAL_DETECTION_INFO)
EVENT_IVS_SHOP_WINDOW_POST	0x00000287	橱窗张贴事件(对应 DEV_EVENT_SHOP_WINDOW_POST_INFO)
EVENT_IVS_SHOP_SIGN_ABNORMAL	0x00000288	店招异常事件(对应 DEV_EVENT_SHOP_SIGN_ABNORMAL_INFO)
EVENT_IVS_BREED_DETECTION	0x00000289	智慧养殖检测事件(对应 DEV_EVENT_BREED_DETECTION_INFO)
EVENT_IVS_AIRPORT_VEHICLE_DETECT	0x0000028A	机场智能保障车辆检测事件(对应 DEV_EVENT_AIRPORT_VEHICLE_DETECT_INFO)
EVENT_IVS_PIG_TEMPERATURE_DETECT	0x0000028B	智慧养殖猪体温检测(只用于规则配置)

智能事件名称	序号	含义
EVENT_IVS_MAN_CAR_COEXISTANCE	0x0000028C	人车共存事件(对应 DEV_EVENT_MAN_CAR_COEXISTANCE_INFO)
EVENT_IVS_HIGH_TOSS_DETECT	0x0000028D	高空抛物检测(对应 DEV_EVENT_HIGH_TOSS_DETECT_INFO)
EVENT_IVS_ELECTRIC_GLOVE_DETECT	0x0000028E	电力检测手套检测事件（对应 DEV_EVENT_ELECTRIC_GLOVE_DETECT_INFO）
EVENT_IVS_ELECTRIC_LADDER_DETECT	0x0000028F	电力检测梯子检测事件(对应 DEV_EVENT_ELECTRIC_LADDER_DETECT_INFO)
EVENT_IVS_ELECTRIC_CURTAIN_DETECT	0x00000290	电力检测布幔检测事件(对应 DEV_EVENT_ELECTRIC_FENCE_DETECT_INFO)
EVENT_IVS_ELECTRIC_FENCE_DETECT	0x00000291	电力检测围栏检测事件(对应 DEV_EVENT_ELECTRIC_FENCE_DETECT_INFO)
EVENT_IVS_ELECTRIC_SIGNBOARD_DETECT	0x00000292	电力检测标识牌检测事件(对应 DEV_EVENT_ELECTRIC_SIGNBOARD_DETECT_INFO)
EVENT_IVS_ELECTRIC_BELT_DETECT	0x00000293	电力检测安全带检测事件(对应 DEV_EVENT_ELECTRIC_BELT_DETECT_INFO)
EVENT_IVS_RADAR_LINE_DETECTION	0x00000294	雷达警戒线/绊线检测（对应 DEV_EVENT_RADAR_LINE_DETECTION_INFO）
EVENT_IVS_RADAR_REGION_DETECTION	0x00000295	雷达警戒区检测事件(对应 DEV_EVENT_RADAR_REGION_DETECTION_INFO)
EVENT_IVS_AUDIO_INTENSITY	0x00000296	异常音事件（对应 DEV_EVENT_AUDIO_INTENSITY_INFO）
EVENT_IVS_PARKING_LOT_STATUS_DETECTION	0x00000297	室外停车位状态检测(对应 DEV_EVENT_PARKING_LOT_STATUS_DETECTION_INFO)
EVENT_IVS_VEHICLE_COMPARE	0x00000298	(只用于规则配置)
EVENT_IVS_DREGS_UNCOVERED	0x00000299	渣土车未遮盖载货检测事件(对应 DEV_EVENT_DREGS_UNCOVERED_INFO)
EVENT_IVS_WALK_DETECTION	0x0000029A	走动检测事件（对应 DEV_EVENT_WALK_DETECTION_INFO）
EVENT_IVS_BACK_TO_DETECTION	0x0000029B	背对检测事件(对应 DEV_EVENT_BACK_TO_DETECTION_INFO)
EVENT_IVS_WRITE_ON_THE_BOARD_DETECTION	0x0000029C	板书检测事件(对应 DEV_EVENT_WRITE_ON_THE_BOARD_DETECTION_INFO)
EVENT_IVS_SMART_KITCHEN_CLOTHES_DETECTION	0x0000029D	智慧厨房穿着检测事件（对不戴口罩、厨师帽以及颜色不符合规定的厨师服进行报警）（对应 DEV_EVENT_SMART_KITCHEN_CLOTHES_DETECTION_INFO）

智能事件名称	序号	含义
EVENT_IVS_SLEEP_DETECT	0x0000029E	睡觉检测事件(对应 DEV_EVENT_SLEEP_DETECT_INFO)
EVENT_IVS_WALK_AROUND_DETECT	0x0000029F	随意走动检测事件（对应 DEV_EVENT_WALK_AROUND_DETECT_INFO）
EVENT_IVS_PLAY_MOBILEPHONE	0x00000300	玩手机事件(对应 DEV_EVENT_PLAY_MOBILEPHONE_INFO)
EVENT_IVS_FINANCE_CONTRABAND_DETECT	0x00000301	智慧金融违规物品检测事件(对应 DEV_EVENT_FINANCE_CONTRABAND_DETECT_INFO)
EVENT_IVS_FINANCE_CASH_TRANSACTION	0x00000302	智慧金融现金交易检测事件（对应 DEV_EVENT_FINANCE_CASH_TRANSACTION_INFO）
EVENT_IVS_ANATOMY_TEMP_DETECT	0x00000303	人体温智能检测事件(对应 DEV_EVENT_ANATOMY_TEMP_DETECT_INFO)
EVENT_IVS_ACTIVITY_ANALYSE	0x00000304	活跃度统计规则（对应 DEV_EVENT_DOOR_STATUS_INFO）
EVENT_IVS_DOOR_STATUS	0x00000305	门状态事件(对应 DEV_EVENT_DOOR_STATUS_INFO)
EVENT_IVS_DHOP_CUSTOM	0x00000306	Dhop 自定义事件（start/stop, 对应 DEV_EVENT_DHOP_CUSTOM_INFO）
EVENT_IVS_DHOP_CUSTOM_ONCE	0x00000307	Dhop 自定义事件（Pulse, 对应 DEV_EVENT_DHOP_CUSTOM_INFO）
EVENT_IVS_FOG_DETECTION	0x00000308	起雾检测事件(对应 DEV_EVENT_FOG_DETECTION)
EVENT_IVS_TRAFFIC_VEHICLE_BC	0x00000309	飙车事件（对应 DEV_EVENT_TRAFFIC_VEHICLE_BC）
EVENT_IVS_TRAFFIC_MOTOR_OVERLOAD	0x0000030A	机动车超载（对应 DEV_EVENT_TRAFFIC_MOTOR_OVERLOAD_INFO）
EVENT_IVS_TRAFFIC_PLATE_OCCLUSION	0x0000030B	车牌污损（对应 DEV_EVENT_TRAFFIC_PLATE_OCCLUSION_INFO）
EVENT_IVS_NONMOTOR_ENTRYING	0x0000030C	非机动车进入电梯(对应 DEV_EVENT_NONMOTOR_ENTRYING_INFO)
EVENT_IVS_WATER_STAGE_MONITOR	0x0000030D	水位监测事件,目前仅用于任务型智能分析(对应 DEV_EVENT_WATER_STAGE_MONITOR_INFO)
EVENT_IVS_TRAFFIC_ROAD_ALERT	0x0000030E	道路安全预警(对应 DEV_EVENT_TRAFFIC_ROAD_ALERT_INFO)
EVENT_IVS_BREAK_RULE_BUILDING_DETECTION	0x0000030F	违章建筑检测事件(对应 DEV_EVENT_BREAK_RULE_BUILDIING_DETECTION_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR_RUN_REDLIGHT	0x00000310	非机动车闯红灯(对应 DEV_EVENT_TRAFFIC_NONMOTOR_RUN_REDLIGHT_INFO)

智能事件名称	序号	含义
EVENT_IVS_TRAFFIC_VEHICLE_IN_EMERGENCY_LANE	0x00000311	占用应急车道事件(对应 DEV_EVENT_TRAFFIC_VEHICLE_IN_EMERGENCY_LANE_INFO)
EVENT_IVS_PRAM_DETECTION	0x00000312	婴儿车检测事件 (对应 DEV_EVENT_PRAM_DETECTION_INFO)
EVENT_IVS_STEREO_PRAM_DETECTION	0x00000313	立体行为婴儿车检测事件 (只用于规则配置)
EVENT_IVS_BIG_BAGGAGE_DETECTION	0x00000314	大件行李箱检测事件 (对应 DEV_EVENT_BIG_BAGGAGE_DETECTION_INFO)
EVENT_IVS_STEREO_BIG_BAGGAGE_DETECTION	0x00000315	立体行为大件行李箱检测事件 (只用于规则配置)
EVENT_IVS_TICKET_EVADE_DETECTION	0x00000316	逃票检测事件 (对应 DEV_EVENT_TICKET_EVADE_DETECTION_INFO)
EVENT_IVS_TRAFFIC_CAR_MEASUREMENT	0x00000320	交通卡口测量(车辆长、宽、高度、重量等)事件(对应 DEV_EVENT_TRAFFIC_CAR_MEASUREMENT_INFO)
EVENT_IVS_TRAFFIC_REAREND_ACCIDENT	0x00000322	交通事故事件（对应 DEV_EVENT_TRAFFIC_REAREND_ACCIDENT_INFO）
EVENT_IVS_FIRE_LANE_DETECTION	0x00000324	消防占道检测事件（对应 DEV_EVENT_FIRE_LANE_DETECTION_INFO）
EVENT_IVS_TRAFFIC_NON_MOTOR_RETROGRADE	0x00000328	非机动车逆行事件(对应 DEV_EVENT_TRAFFIC_NON_MOTOR_RETROGRADE_INFO)
EVENT_IVS_CAR_DRIVING_IN	0x00000330	车辆驶入事件(对应 DEV_EVENT_CAR_DRIVING_IN_INFO)
EVENT_IVS_CAR_DRIVING_OUT	0x00000331	车辆驶出事件(对应 DEV_EVENT_CAR_DRIVING_OUT_INFO)
EVENT_IVS_TRAFFIC_SPECIAL_VEHICLE_DETECT	0x00000333	特殊车辆检测(对应 DEV_EVENT_TRAFFIC_SPECIAL_VEHICLE_INFO)
EVENT_IVS_TRAFFIC_NONMOTOR	0x00000335	交通非机动车事件检测,智能服务器使用 (DEV_EVENT_TRAFFIC_NONMOTOR_INFO)
EVENT_IVS_TRAFFIC_VISIBILITY	0x00000337	交通能见度事件检测(对应 DEV_EVENT_TRAFFIC_VISIBILITY_INFO)
EVENT_IVS_TRAFFIC_VEHICLE_CLEANLINESS	0x00000338	交通车辆清洁度检测事件检测(对应 DEV_EVENT_TRAFFIC_VEHICLE_CLEANLINESS_INFO)
EVENT_IVS_TRUCKNOTCLEAN_FOR_PRMA	0x0000033A	工程车未清洗对应 DEV_EVENT_TRUCKNOTCLEAN_FOR_PRMA_INFO

智能事件名称	序号	含义
EVENT_IVS_CONVEYORBLOCK_DETECTION	0x0000033E	传送带阻塞报警事件 （对应 DEV_EVENT_CONVEYORBLOCK_DETECTION_INFO）
EVENT_IVS_ANYTHING_DETECT	0x0000033F	全物体类型检测事件(对应 DEV_EVENT_ANYTHING_DETECT_INFO)
EVENT_IVS_OBJECT_ABNORMAL	0x00000340	目标异常事件(对应 DEV_EVENT_OBJECT_ABNORMAL_INFO)
EVENT_IVS_ARTICLE_DETECTION	0x00000345	物品检测(只用于规则配置),对应事件 EVENT_IVS_LEFTDETECTION 或者 EVENT_IVS_TAKENAWAYDETECTION
EVENT_IVS_TRAFFIC_PARKINGSPACE_MANUALSNAP	0x00000346	路侧停车位手动抓图(对应 DEV_EVENT_PARKINGSPACE_MANUALSNAP_INFO)
EVENT_IVS_STREET_SUNCURE	0x00000347	沿街晾晒事件（对应 DEV_EVENT_STREET_SUNCURE_INFO）
EVENT_IVS_DISTRESS_DETECTION	0x0000034C	求救检测事件 (对应 DEV_EVENT_DISTRESS_DETECTION_INFO)
EVENT_IVS_OUTDOOR_ADVERTISEMENT	0x00000348	户外广告事件（对应 DEV_EVENT_OUTDOOR_ADVERTISEMENT_INFO ）
EVENT_IVS_HUDDLE_MATERIAL	0x00000349	乱堆物料检测事件（对应 DEV_EVENT_HUDDLE_MATERIAL_INFO）
EVENT_IVS_FIRE_LINE_DETECTION	0x0000034A	进入消防通道检测事件（对应 DEV_EVENT_FIRE_LINE_DETECTION_INFO）
EVENT_IVS_FOLLOW_CAR_ALARM	0x0000034F	跟车报警(对应 DEV_EVENT_FOLLOW_CAR_ALARM_INFO)
EVENT_IVS_CONVEYER_BELT_RUNOFF	0x00000353	传送带跑偏检测事件(对应 DEV_EVENT_CONVEYER_BELT_RUNOFF_INFO)
EVENT_IVS_TRAFFIC_PARKING_STATISTICS	0x0000035B	车位统计事件（对应 DEV_EVENT_TRAFFIC_PARKING_STATISTICS_INFO）
EVENT_IVS_HEAT_IMAGING_TEMPER	0x0000035C	热成像测温点温度异常报警事件(对应结构体 DEV_EVENT_HEAT_IMAGING_TEMPER_INFO)
EVENT_IVS_SEWAGE_DETECTION	0x00000362	排污检测事件（对应 DEV_EVENT_SEWAGE_DETECTION_INFO)
EVENT_IVS_WATERCOLOR_DETECTION	0x00000363	水体颜色事件（对应 DEV_EVENT_WATERCOLOR_DETECTION_INFO）
EVENT_IVS_TRAFFIC_MOTORCYCLE_FORBID	0x00000364	禁摩事件 (对应 DEV_EVENT_TRAFFIC_MOTORCYCLE_FORBID)
EVENT_IVS_VIDEO_NORMAL_DETECTION	0x00000365	视频正常事件,在视频诊断检测周期结束时,将未报错的诊断项上报正常事件 DEV_EVENT_VIDEO_NORMAL_DETECTION_INFO

智能事件名称	序号	含义
EVENT_IVS_OBJECT_PLACEMENT_DETECTION	0x00000369	物品放置检测事件(对应 DEV_EVENT_OBJECT_PLACEMENT_DETECTION_INFO)
EVENT_IVS_OBJECT_REMOVAL_DETECTION	0x0000036A	物品拿取检测事件(对应 DEV_EVENT_OBJECT_REMOVAL_DETECTION_INFO)
EVENT_IVS_FIRE_EXTINGUISHER_DETECTION	0x0000036C	灭火器检测事件(对应 DEV_EVENT_FIRE_EXTINGUISHER_DETECTION_INFO)
EVENT_IVS_DIALRECOGNITION	0x00000371	仪表检测事件(仅用于规则配置,对应 DEV_EVENT_DIALRECOGNITION_INFO)
EVENT_IVS_ELECTRICFAULT_DETECT	0x00000372	仪表类缺陷检测事件(对应 DEV_EVENT_ELECTRICFAULTDETECT_INFO)
EVENT_IVS_TRASH_WITHOUT_COVER_DETECTION	0x00000373	垃圾桶未盖盖子检测事件(对应 DEV_EVENT_TRASH_WITHOUT_COVER_DETECTION_INFO)
EVENT_IVS_WATER_SPEED_DETECTION	0x0000037B	水流速检测事件(对应 DEV_EVENT_WATER_SPEED_DETECTION_INFO)
EVENT_IVS_TRAFFIC_PARKING_BACKING	0x0000037C	出入口倒车驶离事件(对应 DEV_EVENT_TRAFFIC_PARKING_BACKING_INFO)
EVENT_IVS_BARELAND_DETECTION	0x00000380	裸土检测事件 (DEV_EVENT_BARELAND_DETECTION_INFO)
EVENT_IVS_CONSUMPTION_EVENT	0x00000381	消费事件(对应 DEV_EVENT_CONSUMPTION_EVENT_INFO)
EVENT_IVS_XRAY_UNPACKING_CHECK	0x00000384	X 光开包检查事件(对应 DEV_EVENT_XRAY_UNPACKING_CHECK_INFO)
EVENT_IVS_GENERAL_ATTITUDE_DETECTION	0x00000385	姿态检测事件（对应 DEV_EVENT_GENERAL_ATTITUDE_DETECTION_INFO）
EVENT_IVS_TRAFFIC_CHANGE_LANE_CONTINUES	0x00000387	机动车连续变道违法事件(对应 DEV_EVENT_TRAFFIC_CHANGE_LANE_CONTINUES_INFO)
EVENT_IVS_GENEAL_ATTITUDE	0x0000038C	通用姿态行为事件(对应 DEV_EVENT_GENEAL_ATTITUDE_INFO)
EVENT_IVS_LEAKAGE_DETECTION	0x0000038E	渗漏检测事件(对应 DEV_EVENT_LEAKAGE_DETECTION_INFO)
EVENT_IVS_FISHING_DETECTION	0x00000390	钓鱼检测事件(对应 DEV_EVENT_FISHING_DETECTION_INFO)
EVENT_IVS_CROWD_LEVEL_DETECTION	0x00000395	拥挤程度检测事件(对应 DEV_EVENT_CROWD_LEVEL_DETECTION_INFO)

智能事件名称	序号	含义
EVENT_IVS_DUSTBIN_DETECTION	0x00000397	垃圾桶检测事件(对应 DEV_EVENT_DUSTBIN_DETECTION_INFO)
EVENT_IVS_DIALRECOGNITION_EX	0x00000398	仪表检测事件(对应 DEV_EVENT_DIALRECOGNITION_INFO)
EVENT_IVS_OCR_DETECTION	0x00000399	OCR 检测事件(对应 DEV_EVENT_OCR_DETECTION_INFO)
EVENT_IVS_ROAD_CONDITIONS_DETECTION	0x0000039A	路面检测事件(对应 DEV_EVENT_ROAD_CONDITIONS_DETECTION_INFO)
EVENT_IVS_OPEN_INTELLI	0x0000039D	开放智能事件(对应 DEV_EVENT_OPEN_INTELLI_INFO)
EVENT_IVS_RIDING_MOTOR_CYCLE	0x00000401	摩托车骑跨检测事件(对应 DEV_EVENT_RIDING_MOTOR_CYCLE_INFO)
EVENT_IVS_TRAFFIC_SPEED_DROP_SHARPLY	0x00000404	车辆速度剧减事件(对应 DEV_EVENT_TRAFFIC_SPEED_DROP_SHARPLY_INFO)
EVENT_IVS_TRAFFIC_OVERTAKE_ONRIGHT	0x0000040A	右侧超车事件(对应 DEV_EVENT_TRAFFIC_OVERTAKE_ONRIGHT_INFO)
EVENT_IVS_TRAFFIC_TRUCK_OCCUPIED	0x0000040B	大车占道事件(对应 DEV_EVENT_TRAFFIC_TRUCK_OCCUPIED_INFO)
EVENT_IVS_HUMAN_ANIMAL_COEXISTENCE	0x00000411	人和动物检测事件（对应 DEV_EVENT_HUMAN_ANIMAL_COEXISTENCE_INFO）
EVENT_IVS_VEHICLE_STATE	0x00000422	车辆状态事件(对应 NET_DEV_EVENT_VEHICLE_STATE_INFO)
EVENT_IVS_DOOR_STATE_DETECTION	0x00000424	开关门检测事件(对应 NET_DEV_EVENT_DOOR_STATE_DETECTION_INFO)
EVENT_IVS_WASTE_MIXED_INVEST	0x00000425	垃圾混投事件(对应 NET_DEV_EVENT_WASTE_MIXED_INVEST_INFO)
EVENT_IVS_UNBROKEN_TRASHBAG	0x00000426	垃圾袋未破袋检测事件(对应 NET_DEV_EVENT_UNBROKEN_TRASHBAG_INFO)"),//垃圾袋未破袋检测事件(对应 NET_DEV_EVENT_UNBROKEN_TRASHBAG_INFO)
EVENT_IVS_PERSON_CARRY_TRASHBAG	0x00000427	人员拎袋报警事件(对应 NET_DEV_EVENT_PERSON_CARRY_TRASHBAG_INFO)
EVENT_IVS_DROP_DETECTION	0x00000429	滴漏检测事件(对应 NET_DEV_EVENT_DROP_DETECTION_INFO)
EVENT_IVS_TEMPERATURE_ALARM	0x0000042A	温度报警事件(对应 NET_DEV_EVENT_TEMPERATURE_ALARM_INFO)
EVENT_IVS_HUMIDITY_ALARM	0x0000042B	湿度报警事件(对应 NET_DEV_EVENT_HUMIDITY_ALARM_INFO)
EVENT_IVS_ILLEGAL_CARRIAGE	0x0000042F	非法运输事件(对应 NET_DEV_EVENT_ILLEGAL_CARRIAGE_INFO)

智能事件名称	序号	含义
EVENT_IVS_REMOTE_APPROVAL_ALARM	0x00000438	金融远程审批事件(对应 NET_DEV_EVENT_REMOTE_APPROVAL_ALARM_INFO)
EVENT_IVS_ANTI_COUNTERFEIT	0x00000439	防造假检测事件(对应 NET_DEV_EVENT_ANTI_COUNTERFEIT_INFO)
EVENT_IVS_RAILING_PASS_DETECTION	0x0000043E	隔栏传物事件(对应结构体 NET_DEV_EVENT_RAILING_PASS_DETECTION_INFO)
EVENT_IVS_MULTI_MAN_NUM_DETECTION	0x0000043F	讯问会见室人数报警事件(对应 NET_DEV_EVENT_MULTI_MAN_NUM_DETECTION_INFO)
EVENT_IVS_OBJECT_QUANTITY_DETECTION	0x00000440	目标类型和数量检测报警事件(对应 NET_DEV_EVENT_OBJECT_QUANTITY_DETECTION_INFO)
EVENT_IVS_USERMANAGER_FOR_TWSDK	0x00000441	用户信息上报事件(对应 NET_DEV_EVENT_USERMANAGER_FOR_TWSDK_INFO)
EVENT_IVS_POSITION_SNAP	0x00000447	按位置抓图事件(对应 NET_DEV_EVENT_POSITION_SNAP_INFO)
EVENT_IVS_CIGARETTE_CASE_DETECTION	0x00000450	烟盒检测事件(对应 NET_DEV_EVENT_CIGARETTE_CASE_DETECTION_INFO)
EVENT_IVS_CONVEYOR_BELT_STATUS	0x00000451	传送带运动状态检测报警事件(对应 NET_DEV_EVENT_CONVEYOR_BELT_STATUS_INFO)
EVENT_IVS_COLD_SPOT_WARNING	0x00000455	冷点报警 (对应 NET_DEV_EVENT_COLD_SPOT_WARNING_INFO)
EVENT_IVS_TRAFFIC_ACCELERATION_RAPID	0x00000457	急加速事件(对应 NET_DEV_EVENT_TRAFFIC_ACCELERATION_RAPID_INFO)
EVENT_IVS_TRAFFIC_TURN_SHARP	0x00000458	急转弯事件(对应 NET_DEV_EVENT_TRAFFIC_TURN_SHARP_INFO)
EVENT_IVS_GARBAGE_PLASTICBAG	0x00000459	打包垃圾检测事件(对应 NET_DEV_EVENT_GARBAGE_PLASTICBAG_INFO)
EVENT_IVS_COLLISION_CONFLICT	0x0000045B	碰撞冲突事件(对应 NET_DEV_EVENT_COLLISION_CONFLICT_INFO)
EVENT_IVS_AUDIO_MUTATION	0x0000045E	声强突变事件(对应 NET_DEV_EVENT_AUDIO_MUTATION_INFO)
EVENT_IVS_OBJECT_APPEAR_DETECTION	0x0000045F	目标出现事件(对应 NET_DEV_EVENT_OBJECT_APPEAR_DETECTION_INFO)
EVENT_IVS_OBJECT_DISAPPEAR_DETECTION	0x00000460	目标消失事件(对应 NET_DEV_EVENT_OBJECT_DISAPPEAR_DETECTION_INFO)
EVENT_IVS_OBJECT_STATE_DETECTION	0x00000461	目标状态事件(对应 NET_DEV_EVENT_OBJECT_STATE_DETECTION_INFO)
EVENT_IVS_TRAPPED_IN_LIFT_DETECTION	0x00000462	电梯困人检测(对应 NET_DEV_EVENT_TRAPPED_IN_LIFT_DETECTION_INFO)

智能事件名称	序号	含义
EVENT_IVS_ACTION_COUNT	0x0000046E	行为自定义行为计数事件(对应 NET_DEV_EVENT_ACTION_COUNT_INFO)
EVENT_IVS_WADING_DETECTION	0x0000046F	涉水安全检测、水域监测报警(对应 NET_DEV_EVENT_WADING_DETECTION_INFO)
EVENT_IVS_SAME_OBJECT_SEARCH_DETECT	0x00000472	按图索骥物品检测事件(对应 NET_DEV_EVENT_SAME_OBJECT_SEARCH_DETECT_INFO)
EVENT_IVS_SAME_OBJECT_SEARCH_COUNT	0x00000480	按图索骥物品计数事件(对应 NET_DEV_EVENT_SAME_OBJECT_SEARCH_COUNT_INFO)
EVENT_IVS_GRAIN_HEIGHT_DETECTION	0x0000048F	动粮检测事件(对应 NET_DEV_EVENT_GRAIN_HEIGHT_DETECTION_INFO)
EVENT_IVS_ELEVATOR_WORK_INFO	0x00000493	上报电梯运行数据事件(对应 NET_DEV_EVENT_ELEVATOR_WORK_INFO_INFO)
EVENT_IVS_ELEVATOR_ALARM	0x00000494	电梯异常报警(对应 NET_DEV_EVENT_ELEVATOR_ALARM_INFO)
EVENT_IVS_TRAFFIC_TRUST_CAR	0x00000499	信任车辆事件(对应 NET_DEV_EVENT_TRAFFIC_TRUST_CAR_INFO)
EVENT_IVS_TRAFFIC_SUSPICIOUS_CAR	0x0000049A	嫌疑车辆事件(对应 NET_DEV_EVENT_TRAFFIC_SUSPICIOUS_CAR_INFO)
EVENT_IVS_GROUND_THING_DETECTION	0x000004A4	地物识别(对应 NET_DEV_EVENT_GROUND_THING_DETECTION_INFO)
EVENT_IVS_PERSONNEL_CATEGORY_COUNT	0x000004BE,	人员类型统计事件(对应结构体 NET_DEV_EVENT_PERSONNEL_CATEGORY_COUNT_INFO)
EVENT_IVS_SMART_MOTION_EQUIPMENT	0x000004BF	智能动检事件(对应结构体 NET_DEV_EVENT_SMART_MOTION_EQUIPMENT_INFO)

附录2 法律声明

商标声明

- HDMI™ HIGH-DEFINITION MULTIMEDIA INTERFACE：本声明适用所有产品。如本产品使用 HDMI 技术，词语 HDMI、HDMI High-Definition Multimedia Interface（高清晰度多媒体接口）、HDMI 商业外观和 HDMI 徽标均为 HDMI Licensing Administrator, Inc.的商标或注册商标。本产品已经获得 HDMI Licensing Administrator, Inc.授权使用 HDMI 技术。
- VGA 是 IBM 公司的商标。
- Windows 标识和 Windows 是微软公司的商标或注册商标。
- 在本文档中可能提及的其他商标或公司的名称，由其各自所有者拥有。

责任声明

- 在适用法律允许的范围内，在任何情况下，本公司都不对因本文档中相关内容及描述的产品而产生任何特殊的、附随的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉、文档丢失或预期节约的损失进行赔偿。
- 本文档中描述的产品均“按照现状”提供，除非适用法律要求，本公司对文档中的所有内容不提供任何明示或暗示的保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证。

隐私保护提醒

您安装了我们的产品，您可能会采集人脸、指纹、车牌等个人信息。在使用产品过程中，您需要遵守所在地区或国家的隐私保护法律法规要求，保障他人的合法权益。如，提供清晰、可见的标牌，告知相关权利人视频监控区域的存在，并提供相应的联系方式。

关于本文档

- 本文档供多个型号产品使用，产品外观和功能请以实物为准。
- 如果不按照本文档中的指导进行操作而造成的任何损失由使用方自己承担。
- 本文档会实时根据相关地区的法律法规更新内容，具体请参见产品的纸质、电子光盘、二维码或官网，如果纸质与电子档内容不一致，请以电子档为准。
- 本公司保留随时修改本文档中任何信息的权利，修改的内容将会在本文档的新版本中加入，恕不另行通知。
- 本文档可能包含技术上不准确的地方、或与产品功能及操作不相符的地方、或印刷错误，以公司最终解释为准。
- 如果获取到的 PDF 文档无法打开，请使用最新版本或最主流的阅读工具。

附录3 网络安全建议

保障设备基本网络安全的必须措施：

1. 使用复杂密码

请参考如下建议进行密码设置：

- 长度不小于 8 个字符。
- 至少包含两种字符类型，字符类型包括大小写字母、数字和符号。
- 不包含账户名称或账户名称的倒序。
- 不要使用连续字符，如 123、abc 等。
- 不要使用重叠字符，如 111、aaa 等。

2. 及时更新固件和客户端软件

- 按科技行业的标准作业规范，设备（如 NVR、DVR 和 IP 摄像机等）的固件需要及时更新至最新版本，以保证设备具有最新的功能和安全性。设备接入公网情况下，建议开启在线升级自动检测功能，便于及时获知厂商发布的固件更新信息。
- 建议您下载和使用最新版本客户端软件。

增强设备网络安全的建议措施：

1. 物理防护

建议您对设备（尤其是存储类设备）进行物理防护，比如将设备放置在专用机房、机柜，并做好门禁权限和钥匙管理，防止未经授权的人员进行破坏硬件、外接设备（例如 U 盘、串口）等物理接触行为。

2. 定期修改密码

建议您定期修改密码，以降低被猜测或破解的风险。

3. 及时设置、更新密码重置信息

设备支持密码重置功能，为了降低该功能被攻击者利用的风险，请您及时设置密码重置相关信息，包含预留手机号/邮箱、密保问题，如有信息变更，请及时修改。设置密保问题时，建议不要使用容易猜测的答案。

4. 开启账户锁定

出厂默认开启账户锁定功能，建议您保持开启状态，以保护账户安全。在攻击者多次密码尝试失败后，其对应账户及源 IP 将会被锁定。

5. 更改 HTTP 及其他服务默认端口

建议您将 HTTP 及其他服务默认端口更改为 1024～65535 间的任意端口，以减小被攻击者猜测服务端口的风险。

6. 使能 HTTPS

建议您开启 HTTPS，通过安全的通道访问 Web 服务。

7. MAC 地址绑定

建议您在设备端将其网关设备的 IP 与 MAC 地址进行绑定，以降低 ARP 欺骗风险。

8. 合理分配账户及权限

根据业务和管理需要，合理新增用户，并合理为其分配最小权限集合。

9. 关闭非必需服务，使用安全的模式

如果没有需要，建议您关闭 SNMP、SMTP、UPnP 等功能，以降低设备面临的风险。

如果有需要，强烈建议您使用安全的模式，包括但不限于：

- SNMP：选择 SNMP v3，并设置复杂的加密密码和鉴权密码。
- SMTP：选择 TLS 方式接入邮箱服务器。
- FTP：选择 SFTP，并设置复杂密码。
- AP 热点：选择 WPA2-PSK 加密模式，并设置复杂密码。

10. 音视频加密传输

如果您的音视频数据包含重要或敏感内容，建议启用加密传输功能，以降低音视频数据传输过程中被窃取的风险。

11. 安全审计

- 查看在线用户：建议您不定期查看在线用户，识别是否有非法用户登录。
- 查看设备日志：通过查看日志，可以获知尝试登录设备的 IP 信息，以及已登录用户的关键操作信息。

12. 网络日志

由于设备存储容量限制，日志存储能力有限，如果您需要长期保存日志，建议您启用网络日志功能，确保关键日志同步至网络日志服务器，便于问题回溯。

13. 安全网络环境的搭建

为了更好地保障设备的安全性，降低网络安全风险，建议您：

- 关闭路由器端口映射功能，避免外部网络直接访问路由器内网设备的服务。
- 根据实际网络需要，对网络进行划区隔离：若两个子网间没有通信需求，建议使用 VLAN、网闸等方式对其进行网络分割，达到网络隔离效果。
- 建立 802.1x 接入认证体系，以降低非法终端接入专网的风险。
- 开启设备 IP/MAC 地址过滤功能，限制允许访问设备的主机范围。