How to integrate with Hikvision LPR function via SDK

1 Overview

Vehicle Detection and Mixed-traffic Detection are available for the road traffic monitoring. In Vehicle Detection, the passed vehicle can be detected and the picture of its license plate can be captured; besides, the vehicle color, vehicle logo and other information can be recognized automatically.
In Mixed-traffic Detection, the pedestrian, motor vehicle and non-motor vehicle can be detected, and the picture of the object (for pedestrian/non-motor vehicle/motor vehicle without license plate) or license plate (for motor vehicle with license plate) can be captured. You can send alarm signal to notify the surveillance center and upload the captured picture to FTP server.
Note:
Road traffic function varies according to different camera models.

2 APIs of LPR Funtion

2.1 Picture Searching

2.1.1 API

```
/************************************************
Function:     NET_DVR_FindPicture
Description:  Search the JPEG picture of DVR
Input:
lUserID: The return value of NET_DVR_Login()
pFindParam： The structure of picture information to be found
Output:       NULL
Remarks:
Return FALSE on failure -1, other value will be act as the parameter of
NET_DVR_FindNextPicture.
************************************************/
NET_DVR_API HPR_INT32 CALLBACK NET_DVR_FindPicture(HPR_INT32 lUserID,
NET_DVR_FIND_PICTURE_PARAM* pFindParam)

/************************************************
Function:          NET_DVR_FindNextPicture
Description:  This API is used to get picture one by one.
Input:
lFindHandle : Handle, the return value of NET_DVR_FindPicture()
Output:
lpFindData: Pointer for saving picture information
Return value:
```

Return FALSE on failure -1, other status are as follows:
#define NET_DVR_FILE_SUCCESS 1000 // Get the file directory information successfully
#define NET_DVR_FILE_NOFIND 1001 // No file directory found
#define NET_DVR_ISFINDING 1002 // Searching, please wait
#define NET_DVR_NOMOREFILE 1003 // No more file directory found, search is finished
#define NET_DVR_FILE_EXCEPTION 1004 // Exception when search file directory
***/
NET_DVR_API HPR_INT32 CALLBACK NET_DVR_FindNextPicture(HPR_INT32 lFindHandle,LPNET_DVR_FIND_PICTURE lpFindData)

/***
Function: NET_DVR_CloseFindPicture
Description: Close NET_DVR_FindFile and release the resource
Input:
lFindHandle : Handle of finding picture, the return value of NET_DVR_FindFile()
Output: NULL
Return value:
Return HPR_TRUE on success, return HPR_FALSE on failure
***/

NET_DVR_API HPR_BOOL CALLBACK NET_DVR_CloseFindPicture(HPR_INT32 lFindHandle);

2.1.2 Parameter Definition

lUserID	pFindParam	lFindHandle	lpFindData
The return value of NET_DVR_Login()	NET_DVR_FIND_PICTURE_PARAM	The return value of NET_DVR_FindPicture()	NET_DVR_FIND_PICTURE

2.1.3 Macro Definition and Structure

```
typedef enum _VCA_OPERATE_TYPE_
{
        VCA_LICENSE_TYPE            = 0x1,    //plate number
        VCA_PLATECOLOR_TYPE         = 0x2,    //plate color
        VCA_CARDNO_TYPE             = 0x4,    //card number
        VCA_PLATETYPE_TYPE          = 0x8,    //plate type
        VCA_LISTTYPE_TYPE           = 0x10, //plate list types
        VCA_INDEX_TYPE              = 0x20, //data index 2014-02-25
        VCA_OPERATE_INDEX_TYPE = 0x40    //operation index 2014-03-03
    }VCA_OPERATE_TYPE;
    typedef enum _VCA_VEHICLE_TYPE_
    {
        VCA_OTHER_TYPE              = 0x1,    //other type
        VCA_SMALLCAR_TYPE           = 0x2,    //small car
        VCA_BIGCAR_TYPE             = 0x4,    //big car
        VCA_BUS_TYPE                = 0x8,    //bus
        VCA_TRUCK_TYPE              = 0x10,   //truck
        VCA_CAR_TYPE                = 0x20,     //car
        VCA_MINIBUS_TYPE            = 0x40,     //minibus
        VCA_SMALL_TRUCK_TYPE        = 0x80    //small truck
    }VCA_VEHICLE_TYPE;
    typedef struct tagNET_DVR_FIND_PICTURE_PARAM
    {
        DWORD    dwSize;                 // Structure size
        LONG     lChannel;             // channel number
```

/* Image type to search: 0- scheduled capture, 1- motion detection capture, 2- alarm capture, 3- motion detection or alarm capture, 3-motion detection and alarm capture, 6- manual capture, 9- intelligent capture, 10- PIR alarm, 11- wireless alarm, 12- calling for help alarm, 0xa- snapshot when live view, 0xd-face detection, 0xe- line crossing detection, 0xf- intrusion detection,

0x10- scene change detection, 0x11- screenshot when playback on local device, 0x12- intelligent detection, 0x13- region entrance detection, 0x14- region exit detection, 0x15- Loitering detection, 0x16-People gathering detection, 0x17-quick movement detection, 0x18-park detection,

0x19- Unattended baggage, 0x1a- object removal, 0x1b- plate detection, 0x1c-MixColumn detection, 0xff- all types*/

```
        BYTE      byFileType;
        BYTE      byNeedCard;         // whether need the card or not
        BYTE      byProvince;         //Province index
        BYTE      byRes;    //Whether need to return the coordinate info in the result,
0-no, 1- yes;
        BYTE      sCardNum[CARDNUM_LEN_V3/*49*/0];         // card number
        NET_DVR_TIME    struStartTime;//Start time
        NET_DVR_TIME    struStopTime;// Stop time
        //ITC3.7 New added
        DWORD         dwTrafficType; //effect item, please refer to VCA_OPERATE
```

```
_TYPE
          DWORD              dwVehicleType;  //Vehicle  type,  please  refer  to
VCA_VEHICLE_TYPE
         //illegal type, please refer to VCA_ILLEGAL_TYPE(not support multiple choice )
          DWORD       dwIllegalType;
          BYTE        byLaneNo;   //Lane number (1~99)
           BYTE         bySubHvtType  ;//0-reserved,  1-motor  vehicle,  2-  Non-motor
vehicle, 3-pedestrian
           BYTE        byRes2[2];
          char       sLicense[MAX_LICENSE_LEN/*16*/];  //License number
          BYTE        byRegion;        // Region index: 0-reserved, 1- Europe Region, 2-
Russian Region, 0xff- All;
          /*Nation index: 0- not supported, 1- CZ - Czech Republic)，  2- FRA – France, 3-
DE - Germany), 4- E – Spain, 5-IT – Italy, 6-NL – Netherlands, 7- PL – Poland, 8- SVK -
Slovakia), 9- BY – Belorussia, 10- MDA – Moldova, 11- RU – Russia, 12- UA – Ukraine,
0xfe- Unrecognized, 0xff- All;
          */
          BYTE        byCountry;        // Nation index
          BYTE        byRes3[6];        // Reserved
     }NET_DVR_FIND_PICTURE_PARAM, *LPNET_DVR_FIND_PICTURE_PARAM;
     //time correction
     typedef struct
     {
          DWORD dwYear;      //Year
          DWORD dwMonth;         //Month
          DWORD dwDay;       //Day
          DWORD dwHour;          //hour
          DWORD dwMinute;        //minute
          DWORD dwSecond;        //second
     }NET_DVR_TIME, *LPNET_DVR_TIME;
     typedef enum _VCA_PLATE_COLOR_
     {
          VCA_BLUE_PLATE    = 0,     //blue plate
          VCA_YELLOW_PLATE,          //yellow plate
          VCA_WHITE_PLATE,           //white plate
          VCA_BLACK_PLATE,           //black plate
          VCA_GREEN_PLATE,           //green plate
          VCA_BKAIR_PLATE,           //black plate of civi     a
          VCA_OTHER = 0xff           //else
     }VCA_PLATE_COLOR;

     //Result
     typedef enum _VTR_RESULT
     {
        VTR_RESULT_OTHER          =   0,   //Unknow
        VTR_RESULT_BUS            =   1,   //bus
        VTR_RESULT_TRUCK          =   2,   //truck
        VTR_RESULT_CAR            =   3,   //car
        VTR_RESULT_MINIBUS        =   4,   //minibus
        VTR_RESULT_SMALLTRUCK =   5,   //small truck
        VTR_RESULT_HUMAN          =   6,   //human
        VTR_RESULT_TUMBREL        =   7,   //tumbrel
        VTR_RESULT_TRIKE          =   8,   //trike
        VTR_RESULT_SUV_MPV        =   9,   //SUV/MPV
        VTR_RESULT_MEDIUM_BUS =   10,   //medium bus
        VTR_RESULT_MOTOR_VEHICLE = 11, //motor vehicle
        VTR_RESULT_NON_MOTOR_VEHICLE = 12    //non- motor vehicle
    }VTR_RESULT;
    typedef struct
    {
        char       sFileName[PICTURE_NAME_LEN/*64*/];// picture name
        NET_DVR_TIME struTime;//time
        DWORD dwFileSize;//picture size
        char       sCardNum[CARDNUM_LEN_V30/*40*/];  //card number
        BYTE      byPlateColor ;//please refer to VCA_PLATE_COLOR
        BYTE      byVehicleLogo;//please refer to VLR_VEHICLE_CLASS
  BYTE      byEventSearchStatus; //If there is continuous picture in the result: 0- there
  is no picture behind, 1- there is picture behind.
        BYTE      byRecogResult ;//Please refer to VTR_RESULT
        char      sLicense[MAX_LICENSE_LEN/*16*/];    //license number
        BYTE      byRes[12];
    }NET_DVR_FIND_PICTURE,*LPNET_DVR_FIND_PICTURE;
```

2.1.4 Remark

NULL

2.2 LPR Configuration

2.2.1 API

```
/************************************************
Function:           NET_DVR_GetDeviceConfig
Description: This API is used to get configuration of the device(batch).
Input:     iUserID:    The return value of NET_DVR_Login_V30()
                 dwCommand: NET_DVR_GET_TRIGGEREX_CFG command
```

dwCount: The count to be set, both 0 and 1 mean one, 2 means two, and so forth, the max value is 64

lpInBuffer: The buffer pointer of NET_DVR_TRIGGER_COND

dwInBufferSize: The buffer size of the NET_DVR_TRIGGER_COND

dwOutBufferSize: The size of dwCount*NET_ITC_TRIGGERCFG

Output:

lpStatusList：Error message list, corresponding to the channel to be query, e.g. lpStatusList[2] corresponds to lpInBuffer[2], memory allocated by user. The size of one error message is 4 bytes(32 bit unsigned integer value), the value: 0-successful, >0- failed lpOutBuffer：Buffer pointer of NET_ITC_TRIGGERCFG. The parameter must be corresponding to the channel to be query. If lpStatusList that corresponds to the channel is larger than 0, the content of corresponding lpOutBuffer is invalid.

Return value: HPR_TRUE: success, but it dose not represent all configuration successful, it requires to check lpStatusList[n] to see whether the configuration is succesful or failed.

HPR_FALSE: FALSE means all configuration failed

```
**************************************************/
NET_DVR_API HPR_BOOL CALLBACK NET_DVR_GetDeviceConfig(
HPR_INT32 iUserID,
HPR_UINT32 dwCommand,
HPR_UINT32 dwCount,
HPR_VOIDPTR lpInBuffer,
HPR_UINT32 dwInBufferSize,
HPR_VOIDPTR lpStatusList,
HPR_VOIDPTR lpOutBuffer,
HPR_UINT32 dwOutBufferSize)

/**************************************************
```

Function: NET_DVR_SetDeviceConfig

Description: This API is used to set configuration of the device(batch).

Input: iUserID: The return value of NET_DVR_Login_V30()

dwCommand: NET_DVR_SET_TRIGGEREX_CFG command

dwCount: The count to be set, both 0 and 1 mean one, 2 means two, and so forth, the max value is 64

lpInBuffer: The buffer pointer of NET_DVR_TRIGGER_COND

dwInBufferSize: The buffer size of the NET_DVR_TRIGGER_COND

lpInParamBuffer：Buffer pointer of NET_ITC_TRIGGERCFG. The parameter must be corresponding to the channel to be query. If lpStatusList that corresponds to the channel is larger than 0, the content of corresponding lpInBufferis invalid.

dwInParamBufferSize：The buffer size of NET_ITC_TRIGGERCFG

Input/Output parameter:

lpStatusList：Error message list, corresponding to the channel to be query, e.g. lpStatusList[2] corresponds to lpInBuffer[2], memory allocated by user. The size of

one error message is 4 bytes(32 bit unsigned integer value), the value: 0-successful, >0- failed
Return value: HPR_TRUE： success, but it dose not represent all configuration successful, it requires to check lpStatusList[n] to see whether the configuration is succesful or failed.
HPR_FALSE： FALSE means all configuration failed
```
**************************************************/
NET_DVR_API HPR_BOOL CALLBACK NET_DVR_SetDeviceConfig(
HPR_INT32 iUserID,
HPR_UINT32 dwCommand,
HPR_UINT32 dwCount,
HPR_VOIDPTR lpInBuffer,
HPR_UINT32 dwInBufferSize,
HPR_VOIDPTR lpStatusList,
HPR_VOIDPTR lpInParamBuffer,
HPR_UINT32 dwInParamBufferSize)
```

2.2.2 Parameter Definition

```
#define NET_DVR_GET_TRIGGEREX_CFG        5074    //Get the ITC extension configuration
#define NET_DVR_SET_TRIGGEREX_CFG        5075    //Set the ITC extension configuration
```

dwCommand	lpInBuffer	lpInParamBuffer	lpOutBuffer
NET_DVR_GET_TRIGGEREX_CFG	NET_DVR_TRIGGER_COND	NULL	NET_ITC_TRIGGERCFG
NET_DVR_SET_TRIGGEREX_CFG	NET_DVR_TRIGGER_COND	NET_ITC_TRIGGERCFG	NULL

2.2.3 Macro Definition and Structure

- Macro Definition

```
#define NET_DVR_GET_TRIGGEREX_CFG        5074    //Get ITC extension configuration
#define NET_DVR_SET_TRIGGEREX_CFG        5075    //set ITC extension
```

configuration

Structure

```
typedef enum _ITC_TRIGGERMODE_TYPE_
{
      ITC_POST_IOSPEED_TYPE                    = 0x1, // IO speed measuring
(intelligent monitoring and recording system of vehicles)
      ITC_POST_SINGLEIO_TYPE                   = 0x2, // IO speed measuring
(intelligent monitoring and recording system of vehicles)
      ITC_POST_RS485_TYPE                      = 0x4, // RS485 magnetic vehicle
detector trigger (intelligent monitoring and recording system of vehicles)
      ITC_POST_RS485_RADAR_TYPE                = 0x8, // RS485 radar trigger
(intelligent monitoring and recording system of vehicles)
      ITC_POST_VIRTUALCOIL_TYPE                = 0x10,    // Virtual coil trigger
(intelligent monitoring and recording system of vehicles
      ITC_POST_HVT_TYPE_V50                    = 0x20,  //HVT V50
      ITC_POST_MPR_TYPE                        = 0x40,    //multiframe recongnition
(intelligent monitoring and recording system of vehicles)(Ver3.7)
       ITC_POST_PRS_TYPE                       = 0x80,    //video detection trigger
      ITC_EPOLICE_IO_TRAFFICLIGHTS_TYPE = 0x100, // IO traffic lights (electronic
police)
      ITC_EPOLICE_RS485_TYPE                   = 0x200, // RS485 magnetic vehicle
detector trigger (electronic police)
      ITC_POST_HVT_TYPE                   = 0x400, //HVT (intelligent monitoring and
recording system of vehicles)
      ITC_PE_RS485_TYPE                        = 0x10000, // RS485 magnetic vehicle
detector trigger(electronic police and intelligent monitoring and recording system of
vehicles)     ITC_VIDEO_EPOLICE_TYPE                  = 0x20000, // Video trigger
(electronic police and intelligent monitoring and recording system of vehicles)
      ITC_VIA_VIRTUALCOIL_TYPE                 = 0x40000, //VIA trigger
      ITC_POST_IMT_TYPE                   =  0x80000,     //intelligent surveillance
configuration
       IPC_POST_HVT_TYPE                       = 0x100000  //HVT of IPC
}ITC_TRIGGERMODE_TYPE;

typedef struct tagNET_DVR_TRIGGER_COND
{
      DWORD   dwSize;           //Structure size
      DWORD   dwChannel;        //Channel number
      DWORD   dwTriggerMode;//trigger mode, refer to ITC_TRIGGERMODE_TYPE
      BYTE    byDetSceneID;// scene ID [1,4], IPC- 0(by default)
      BYTE    byRes[63];      //reserved
}NET_DVR_TRIGGER_COND,*LPNET_DVR_TRIGGER_COND;
//Point
```

```
typedef struct tagNET_VCA_POINT
{
      float fX;                                    // X, 0.001~1
      float fY;                                    //Y, 0.001~1
}NET_VCA_POINT, *LPNET_VCA_POINT;
//Line
typedef struct tagNET_VCA_LINE
{
      NET_VCA_POINT struStart;        //start point
      NET_VCA_POINT struEnd;           //end point
}NET_VCA_LINE, *LPNET_VCA_LINE;
//Line definition
typedef enum _ITC_LINE_TYPE_
{
      ITC_LINT_UNKNOW          = 0,      //unknown
      ITC_LINE_WHITE           = 1, //white
      ITC_LINE_STOP            = 2, //stop line
      ITC_LINE_SINGLE_YELLOW = 3, //single yellow line
      ITC_LINE_DOUBLE_YELLOW = 4, //double yellow line
      ITC_LINE_GUARD_RAIL = 5, //there is guard rail on the lane
      ITC_LINE_NO_CROSS = 6, //no cross lane
      ITC_LINE_DOTTED = 7 //dotted
}ITC_LINE_TYPE;

//Structure of video electronic police line
typedef struct tagNET_ITC_LINE
{
      NET_VCA_LINE struLine; //line parameter
      BYTE byLineType; //line type, please refer t ITC_LINE_TYPE
      BYTE byRes[7];
}NET_ITC_LINE, *LPNET_ITC_LINE;

//Plate recognition parameter
typedef struct tagNET_ITC_PLATE_RECOG_PARAM
{
       BYTE byDefaultCHN[MAX_CHJC_NUM/*3*/]; /* Chinese characters of province*/
       BYTE byEnable; //Whether enable the Plate recognition of region, 0-n    o, 1-yes
       DWORD dwRecogMode;    /*
              bit0- Back Plate: 0- Front Plate, 1- Back Plate;
             bit1- Small-Size Plate Recognition or Large-Size Plate Recognition: 0- Small-
Size Plate Recognition, 1- Large-Size Plate Recognition;
             bit2- Vehicle Color Recognization:0- disable in Back Plate and Small-Size Plate
Recognition; 1- enable;
             bit3- Agricultural Vehicle Recognition: 0-disbale,1-enable;
```

```
            bit4- Fuzzy Recognization: 0-disable, 1-enable;
            bit5- frame alignment or Field orientation: 0- frame alignment, 1- Field
orientation;
            bit6- frame identification or Field identification: 0- frame identification, 1-
Field identification;
            bit7-Night or daylight: 0-daylight, 1-night;
            bit8- motorcycle identification: 0- disable, 1- enable;
            bit9-scene mode: 0-electronic police/multiframe, 1- Checkpoint;
            bit10-tiny palte: 0-disable, 1-enable (pixel 60～80)
            bit11- safety belt detection: 0- disable, 1- enable
              bit12- plate recognition of civil aviation: 0- disable, 1- enable;
              bit13- plate Excessive tilt: 0- disable, 1- enable(PRS)
            bit14-oversized plate identification: 0- disable, 1- enable (PRS)
              bit15- sun shield detection: 0- disable, 1- enable
            bit16- Yellow Label Car detection: 0- disable, 1- enable
            bit17- hazardous article vehicle detection: 0- disable, 1- enable;
     */
    BYTE byVehicleLogoRecog;//whether enable vehicle logo recognition: 0-disable,
1-enable;
    BYTE byProvince;
    BYTE byRegion;        // region index value: 0-reserved, 1- Europe Region, 2-
Russian Region;
    BYTE byRes[29];
}NET_ITC_PLATE_RECOG_PARAM, *LPNET_ITC_PLATE_RECOG_PARAM;
// Structure of polygon
typedef struct tagNET_ITC_POLYGON
{
    DWORD dwPointNum; //Valid point, >=3, if three points are on a straight line, or
the lines cross, it is considered to invalid area
    NET_VCA_POINT    struPos[ITC_MAX_POLYGON_POINT_NUM/*20*/]; //Polygon
boundary point, the max number is 20
}NET_ITC_POLYGON, *LPNET_ITC_POLYGON;
//drive direction definition not supported now
typedef enum _ITC_LANE_CAR_DRIVE_DIRECT_
{
     ITC_LANE_DRIVE_UNKNOW         = 0, //unknown
     ITC_LANE_DRIVE_UP_TO_DOWN = 1, //drive from up to down
     ITC_LANE_DRIVE_DOWN_TO_UP = 2    //drive from down to up
}ITC_LANE_CAR_DRIVE_DIRECT;
//Definition of lane direction     not supported now
typedef enum _ITC_RELA_LANE_DIRECTION_TYPE_
{
     ITC_RELA_LANE_DIRECTION_UNKNOW           = 0,     //else
    ITC_RELA_LANE_EAST_WEST                   = 1,      //from east to west
    ITC_RELA_LANE_WEST_EAST                    = 2,      //from west to east
    ITC_RELA_LANE_SOUTH_NORTH                  = 3,      //from south to north
    ITC_RELA_LANE_NORTH_SOUTH                  = 4,      //from north to south
    ITC_RELA_LANE_EASTSOUTH_WESTNORTH   = 5,      //from east south to west
north
    ITC_RELA_LANE_WESTNORTH_EASTSOUTH   = 6,      //from west north to east
south
    ITC_RELA_LANE_EASTNORTH_WESTSOUTH   = 7,      //from east north to west
south
    ITC_RELA_LANE_WESTSOUTH_EASTNORTH   = 8       //from west south to east
north
} ITC_RELA_LANE_DIRECTION_TYPE;
typedef struct tagNET_ITC_LANE_MPR_PARAM
{
    BYTE byLaneNO;
    union
    {
        BYTE    uLen[4];
        struct
        {
        BYTE    byIONo;//IO number x[1, byIoInNum+1], start from 1, and the max
  value can be relate to the byIoInNum of NET_DVR_SNAP_ABILITY.
            BYTE    byTriggerType;//0-falling edge, 1-rising edge
            BYTE    byRes1[2];
        }struIO;//valid in IO mode
        struct
        {
            BYTE    byRelateChan;// relevance number [1,16] of magnetic vehicle
detector.
            BYTE    byRes2[3];
        }struRS485;
    }uTssParamInfo;
    BYTE byCarDriveDirect; //vehicle drive direction, ITC_LANE_CAR_DRIVE_DIRECT
    BYTE byRes[58];
    NET_ITC_LINE struLaneLine;//lane line
    NET_ITC_POLYGON struPlateRecog;//license recognition region
    //Relevance lane direction type, refer to ITC_RELA_LANE_DIRECTION_TYPE
    //The lane direction paramer corresponding with the relevance lane number.
    BYTE byRelaLaneDirectionType;
     BYTE byRes1[255];
}NET_ITC_LANE_MPR_PARAM,*LPNET_ITC_LANE_MPR_PARAM;
typedef struct tagNET_ITC_POST_MPR_PARAM
{
    BYTE byEnable;
```

```
        BYTE byLaneNum;
        BYTE bySourceType; //0-MPR, 1-IO (FVNP), 2-RS485
        BYTE byRes[61];
        NET_ITC_LINE struLaneBoundaryLine;//Boundary line of lane(far left)
        NET_ITC_PLATE_RECOG_PARAM struPlateRecog;//plate recognition parameter 40
        NET_ITC_LANE_MPR_PARAM struLaneParam[MAX_ITC_LANE_NUM/*6*/];
         char szSceneName[NAME_LEN/*32*/]; //name of scene
         BYTE byRes1[408];
}NET_ITC_POST_MPR_PARAM,*LPNET_ITC_POST_MPR_PARAM;
typedef union tagNET_ITC_TRIGGER_PARAM_UNION
{
        DWORD    uLen[1070];                //parameter
        NET_ITC_POST_IOSPEED_PARAM          struIOSpeed;       //IO speed measuring
parameter (intelligent monitoring and recording system of vehicles)
        NET_ITC_POST_SINGLEIO_PARAM          struSingleIO;      //Single IO parameter
(intelligent monitoring and recording system of vehicles)
        NET_ITC_POST_RS485_PARAM                 struPostRs485;    //RS485 magnetic
vehicle detector parameter (intelligent monitoring and recording system of vehicles)
        NET_ITC_POST_RS485_RADAR_PARAM      struPostRadar;      // RS485 radar
parameter (intelligent monitoring and recording system of vehicles)
        NET_ITC_POST_VTCOIL_PARAM           struVtCoil;        // Virtual coil parameter
(intelligent monitoring and recording system of vehicles)
        NET_ITC_POST_HVT_PARAM                  struHvt;          //HVT          parameter
(intelligent monitoring and recording system of vehicles)
        NET_ITC_EPOLICE_IOTL_PARAM          struIOTL;          //   IO   traffic   light
parameter (electronic police)
        NET_ITC_EPOLICE_RS485_PARAM           struEpoliceRs485;  // RS485 magnetic
vehicle detector parameter (electronic police)
        NET_ITC_EPOLICE_RS485_PARAM        struPERs485;    //     RS485     magnetic
vehicle detector parameter (electronic police for intelligent monitoring and recording
system of vehicles)    NET_ITC_POST_MPR_PARAM                              struPostMpr;
//Multiframe detection trigger(MPR)
        NET_DVR_VIA_VTCOIL_PARAM              struViaVtCoil;    //(VIA) video detection
parameter
        NET_ITC_POST_IMT_PARAM                    struPostImt;// intelligent surveillance
trigger
         NET_ITC_POST_PRS_PARAM                  struPostPrs;//video detection trigger
         NET_IPC_POST_HVT_PARAM                  struIpcHvt;//(IPC) HVT parameters
        NET_ITC_POST_HVT_PARAM_V50            struHvtV50;    /* HVT parameter V50
(intelligent monitoring and recording system of vehicles) */
}NET_ITC_TRIGGER_PARAM_UNION, *LPNET_ITC_TRIGGER_PARAM_UNION;
//Structure of single trigger parameter
typedef struct tagNET_ITC_SINGLE_TRIGGERCFG
{
    BYTE    byEnable;//whether need to enable, 0-no, 1-yes
    BYTE    byRes1[3];
    DWORD   dwTriggerType;  //trigger type, please see details in ITC_TRIGGERMODE_TYPE
    NET_ITC_TRIGGER_PARAM_UNION uTriggerParam; //trigger parameter
    BYTE    byRes[64];
}NET_ITC_SINGLE_TRIGGERCFG, *LPNET_ITC_SINGLE_TRIGGERCFG;
// Structure of trigger parameter
typedef struct tagNET_ITC_TRIGGERCFG
{
    DWORD     dwSize;              // Structure length
    NET_ITC_SINGLE_TRIGGERCFG   struTriggerParam;   //single trigger parameter
    BYTE       byRes[32];
}NET_ITC_TRIGGERCFG, *LPNET_ITC_TRIGGERCFG;
```

2.2.4 Remarks

Null

2.3 Intelligent Control Configuration Ability

2.3.1 API

```
/***************************************************
Function:        NET_DVR_GetDeviceAbility
Description: Get capability set of the device
Input:
    lUserID: The return value of NET_DVR_Login()
    dwAbilityType: Type of capability, details listed below:
    pInBuf: Pointer of the input buffer (according to description mode of capability parameter which is defined by device,it supports XML text or structure format)
    dwInLength: Length of input buffer
Output:
    pOutBuf： Pointer of the output buffer
    dwOutLength: Length of the output buffer
Return value:
    Returns TRUE on success, FALSE on failure.
***************************************************/
NET_DVR_API BOOL __stdcall NET_DVR_GetDeviceAbility(LONG lUserID, DWORD dwAbilityType, char* pInBuf, DWORD dwInLength, char* pOutBuf, DWORD dwOutLength);
```

2.3.2 Macro Definition and Structure

- Macro Definition

```
#define    VCA_DEV_ABILITY     0x100 //intelligent device ability
```

- Structure

```
//VCA Ability
typedef struct tagNET_VCA_DEV_ABILITY
{
    DWORD dwSize;                    //Structure size
    BYTE byVCAChanNum;               //The total number of intelligent channels
    BYTE byPlateChanNum;             //The total number of plate channels
    BYTE byBBaseChanNum;             //The total number of basic behaviour version
channels
    BYTE byBAdvanceChanNum;          //The total number of advanced behaviour
version channels
    BYTE byBFullChanNum;             //The total number of complete behaviour
version channels
    BYTE byATMChanNum;               //The total number of intelligent ATM channels
    BYTE byPDCChanNum;               //The total number of pedestrian counting
channels
    BYTE byITSChanNum;               //The total number of traffic event channels
    BYTE byBPrisonChanNum;           //The total number of behaviour prison version
channels
    BYTE byFSnapChanNum;             // The total number of face snapshot channels
    BYTE byFSnapRecogChanNum;        // The total number of face snapshot and
recognition channels
    BYTE byFRetrievalChanNum;    // The total number of face backward retrieval
    BYTE bySupport;                  //ability, 0- not supported, 1- support
                                     //bySupport & 0x1- whether support intelligent
trace 2012-3-22
                                     //bySupport & 0x2- whether support 128
  channels stream extension 2012-12-27
    BYTE byFRecogChanNum;            //Channel number of face detection
    BYTE byBPPerimeterChanNum; // Channel number of behavior in jail (perimeter)
    BYTE byTPSChanNum;               // Channel number of Traffic Guidance
    BYTE byTFSChanNum;               // Channel number of Violation Forensics
    BYTE byFSnapBFullChanNum; //channel number of face snapshot and behavioural
analysis
    BYTE byHeatMapChanNum;           // Channel number of Heatmap channel
    BYTE bySmartVehicleNum;          // Channel number of SMART event and vehicle
detection
    BYTE bySmartHVTNum;              // Channel number of SMART event and HVT
    BYTE bySmartNum;             //number of SMART event
    BYTE byVehicleNum;          //number of vehicle detection channel
    BYTE byRes[17];
}NET_VCA_DEV_ABILITY, *LPNET_VCA_DEV_ABILITY;
```

2.3.3 Remarks

NULL

2.4 Get/Set Intelligent Control Parameters APIs

2.4.1 API

```
/*************************************************
Function:           NET_DVR_GetDVRConfig
Description:  Get DVR parameter
Input:      nUserID: The return value of NET_DVR_Login()
nCommand: Configuration command: NET_DVR_GET_VCA_CTRLCFG
lChannel: Channel number
nOutBufferSize： The length of the buffer: NET_VCA_CTRLCFG
Output:
lpOutBuffer：       Buffer pointer, NET_VCA_CTRLCFG
lpBytesReturned：  The size of the returned buffer, it can't be NULL
Return value:
Returns HPR_TRUE on success, HPR_FALSE on failure.
*************************************************/
NET_DVR_API HPR_BOOL CALLBACK NET_DVR_GetDVRConfig(HPR_INT32 nUserID,
HPR_UINT32 nCommand, HPR_INT32 nChannel, HPR_VOIDPTR lpOutBuffer,
HPR_UINT32 nOutBufferSize, HPR_UINT32 *lpBytesReturned)

/*************************************************
Function:           NET_DVR_SetDVRConfig
Description:  Set DVR parameter
Input:      nUserID: The return value of NET_DVR_Login()
nCommand: Configuration command, NET_DVR_SET_VCA_CTRLCFG
lChannel: Channel number
lpInBuffer: Buffer pointer, NET_VCA_CTRLCFG
nInBufferSize: The length of the buffer, NET_VCA_CTRLCFG
Output:     null
Return value:           Returns HPR_TRUE on success, HPR_FALSE on failure.
*************************************************/
NET_DVR_API HPR_BOOL CALLBACK NET_DVR_SetDVRConfig(HPR_INT32 nUserID,
HPR_UINT32 nCommand, HPR_INT32 nChannel, HPR_VOIDPTR lpInBuffer,
```

HPR_UINT32 nInBufferSize)

2.4.2 Parameter Definition

```
#define     NET_DVR_SET_VCA_CTRLCFG 164   //Set intelligent control parameter
#define     NET_DVR_GET_VCA_CTRLCFG 165   //Get intelligent control parameter
```

nUserID	dwCommand	lpInBuffer	lpOutBuffer
NET_DVR_Login()	NET_DVR_GET_VCA_CTRLCFG		NET_VCA_CTRLCFG
NET_DVR_Login()	NET_DVR_SET_VCA_CTRLCFG	NET_VCA_CTRLCFG	

2.4.3 Macro Definition and Structure

- Macro Definition

```
#define     NET_DVR_SET_VCA_CTRLCFG 164   // Set intelligent control parameter
#define     NET_DVR_GET_VCA_CTRLCFG 165   // Get intelligent control parameter
```

- Structure

```
#define MAX_VCA_CHAN 16//max intelligent channel number
//intelligent channel type
typedef enum _VCA_CHAN_ABILITY_TYPE_
{
    VCA_BEHAVIOR_BASE          = 1,        //Basic behaviour analysis
    VCA_BEHAVIOR_ADVANCE = 2,              //Advanced behaviour analysis
    VCA_BEHAVIOR_FULL          = 3,        //Complete behaviour analysis
     VCA_PLATE                 = 4,        //Capacity of plate
    VCA_ATM                     = 5,        //ATM ability
     VCA_PDC                    = 6,        //Capacity of pedestrian counting
     VCA_ITS                    = 7,        //intelligent traffic event
     VCA_BEHAVIOR_PRISON       =  8,        // behaviour analysis jail version
(Dormitories)
    VCA_FACE_SNAP              = 9,        //face snapshot ability
    VCA_FACE_SNAPRECOG       =  10,        // face snapshot and recognition
ability
    VCA_FACE_RETRIEVAL       = 11,        //face backward retrieval ability
    VCA_FACE_RECOG             = 12,       // face recognition ability
    VCA_BEHAVIOR_PRISON_PERIMETER = 13, // behaviour analysis jail version (perimeter)
    VCA_TPS                    = 14,          // Traffic Guidance
    VCA_TFS                    = 15,          // Road peccancy forensics
    VCA_BEHAVIOR_FACESNAP = 16,              // face snapshot and behaviour analysis
    VCA_HEATMAP                = 17,          //heatmap
    VCA_SMART_VEHICLE_DETECTION = 18, // SMART event and vehicle detection
    VCA_SMART_HVT_DETECTION        = 19,    // SMART event and HVT
    VCA_SMART_EVENT                   = 20, // SMART event
    VCA_VEHICLE_DETECTION                = 21     // vehicle detection
} VCA_CHAN_ABILITY_TYPE;
typedef struct tagNET_VCA_CTRLINFO
{
    BYTE     byVCAEnable;          //whether enable VCA
    BYTE     byVCAType;        //VCA ability type, VCA_CHAN_ABILITY_TYPE
    BYTE     byStreamWithVCA; //whether there is VCA info in the stream
    BYTE     byMode;               //Mode, ATM ability: refer to VCA_CHAN_MODE_TYPE; TFS ability: refer to TFS_CHAN_MODE_TYPE
    BYTE     byControlType;     //, whether show the control type by bit: 0- no, 1-yes
                                 // byControlType &1 whether enable snapshot function
    BYTE     byPicWithVCA; // whether overlay target information on the picture: 0-no (by default), 1-yes;
    BYTE     byRes[2];           // Reserved, please set to 0
} NET_VCA_CTRLINFO, * LPNET_VCA_CTRLINFO;

// Structure of intelligent control
typedef struct tagNET_VCA_CTRLCFG
{
    DWORD dwSize;
  NET_VCA_CTRLINFO struCtrlInfo [MAX_VCA_CHAN];    //Control info, array 0 stands for the start channel of device
  BYTE byRes [16];
} NET_VCA_CTRLCFG, * LPNET_VCA_CTRLCFG;;
```

2.4.4 Remarks

NULL

2.5 Plate Recognition alarm uploading

2.5.1 Arming

2.5.1.1 API

```
/**************************************************
Function:       NET_DVR_SetDVRMessageCallBack_V30
Description:    register callback function to receive device alarm message(with user
data and callbacj the detailed device info)
Input:      fMessageCallBack: callback info
                        lCommand: alarm info, COMM_ITS_PLATE_RESULT
                        pAlarmer: alarmer info, NET_DVR_ALARMER
                        pAlarmInfo: alarm info, NET_ITS_PLATE_RESULT
                        dwBufLen: alarm length, length of NET_ITS_PLATE_RESULT
                        pUser: user data
Output:     NULL
Return value:   Returns HPR_TRUE on success, HPR_FALSE on failure.
**************************************************/
NET_DVR_API HPR_BOOL CALLBACK NET_DVR_SetDVRMessageCallBack_V30(
MSGCallBack fMessageCallBack,
  HPR_VOIDPTR pUser);
typedef void (CALLBACK *MSGCallBack)(LONG lCommand, NET_DVR_ALARMER
*pAlarmer, char *pAlarmInfo, DWORD dwBufLen, void* pUser);
/**************************************************
Function:           NET_DVR_SetupAlarmChan_V41
Description:  set alarm upload channel
Input:      lUserID: The return value of NET_DVR_Login().
lpSetupParam: Arm priority settings parameter
Output:
Return value：        -1 means false, other values are as handle parameters of function
NET_DVR_CloseAlarmChan.
**************************************************/
NET_DVR_API HPR_INT32 CALLBACK NET_DVR_SetupAlarmChan_V41 (HPR_INT32
lUserID, LPNET_DVR_SETUPALARM_PARAM lpSetupParam);
```

2.5.1.2 Parameter Definition

```
#define COMM_ITS_PLATE_RESULT                0x3050 //terminal picture
uploading
```

lCommand	pAlarmer	pAlarmInfo	dwBufLen

COMM_ITS_PLATE_RESULT	NET_DVR_ALARMER	NET_ITS_PLATE_RESULT	Length of NET_ITS_PLATE_RESULT

2.5.1.3 Macro Definition and Structure

- Macro Definition

```
#define MAX_LICENSE_LEN           16     //max length of plate number
#define DEVICE_ID_LEN              48     //length of device ID
#define MONITORSITE_ID_LEN        48     // length of camera ID
```

- Structure

```
typedef struct tagNET_DVR_SETUPALARM_PARAM
{
      DWORD dwSize;
   BYTE          byLevel; //Arming priority: 0- level one (high), 1- level two (medium),
2- level three (low)
   BYTE     byAlarmInfoType; //The type of the alarm information to upload (for
intelligent traffic camera): 0- old type (NET_DVR_PLATE_RESULT), 1- new type
(NET_ITS_PLATE_RESULT)
2012-9-28
    BYTE    byRetAlarmTypeV40;   //0—return   NET_DVR_ALARMINFO_V30   or
    NET_DVR_ALARMINFO, 1—return NET_DVR_ALARMINFO_V40 when device
    supports, while return NET_DVR_ALARMINFO_V30 or NET_DVR_ALARMINFO.
    BYTE    byRetDevInfoVersion; //version of CVR alarm callback structure, 0-
COMM_ALARM_DEVICE，  1-COMM_ALARM_DEVICE_V40
    BYTE          byRetVQDAlarmType;     //VQD     alarm     type    ，    0-
    NET_DVR_VQD_DIAGNOSE_INFO，1-NET_DVR_VQD_ALARM
   //1- INTER_FACE_DETECTION, 0- INTER_FACESNAP_RESULT
   BYTE     byFaceAlarmDetection;
    BYTE         bySupport; //Bit0- whether there is a need to upload picture in
secondary arm: 0-yes, 1-no
    BYTE    byRes;
   WORD  wTaskNo;  //task number (corresponding to the dwTaskNo of
NET_DVR_VEHICLE_RECOG_RESULT      and      the      dwTaskNo      of
ET_DVR_VEHICLE_RECOG_COND)
   BYTE     byRes1[5];
    BYTE    byCustomCtrl;//Bit0- support face Subgraph of copilot uploading: 0-no,
1-yes
} NET_DVR_SETUPALARM_PARAM, *LPNET_DVR_SETUPALARM_PARAM;
//frame structure of region
typedef struct tagNET_VCA_RECT
{
   float fX;                         // X coordinate of top left corner, 0.001~1
   float fY;                         // Y coordinate of top left corner, 0.001~1
```

```
    float fWidth;                    //width, 0.001~1
    float fHeight;                   //height, 0.001~1
} NET_VCA_RECT, *LPNET_VCA_RECT;
// sub-structure of plate recognition result
typedef struct tagNET_DVR_PLATE_INFO
{
    BYTE    byPlateType;                     //plate type
    BYTE    byColor;                         //plate color
    BYTE    byBright;                        //plate bright
    BYTE    byLicenseLen;                        //license length of plate
    BYTE    byEntireBelieve;                 //confidence coefficient of plate, -100
    BYTE    byRegion;         // region index, 0- resered, 1- Europe Region, 2- Russian
Region, 0xff-all
    /*Nation index value
0-  Not supported, 1-CZ - Czech Republic, 2-FRA – France, 3-DE - Germany, 4-E –
    Spain, 5-IT – Italy, 6-NL – Netherlands, 7-PL – Poland, 8-SVK - Slovakia, 9-BY -
    Belorussia, 10-MDA – Moldova, 11-RU – Russia, 12-UA - Ukraine, 0xfe- can't be
    recognized*/
    BYTE    byCountry;                           // region index value
    BYTE    byRes[33];                       //reserved
    NET_VCA_RECT  struPlateRect;                //plate position
    char sLicense[MAX_LICENSE_LEN];            //plate number
    BYTE byBelieve[MAX_LICENSE_LEN];   // confidence coefficient of each character
identification;
}NET_DVR_PLATE_INFO, *LPNET_DVR_PLATE_INFO;
//vehicle info
typedef struct tagNET_DVR_VEHICLE_INFO_
{
        DWORD dwIndex;                  //vehicle index
        BYTE        byVehicleType;       /vehicle type, 0-others, 1- small car, 2- oversize
vehicle, 3- pedestrian triggers, 4- cart triggers, 5- tricycle triggers (3.5Ver)
        BYTE        byColorDepth;           //color depth of car
        BYTE    byColor;                //color of car, refer to VCR_CLR_CLASS
        BYTE        byRes1;                 //
        WORD wSpeed;                   //unit:km/h
        WORD    wLength;                 // bodywork length of the previous car  /*
Violation type: 0-normal, 1-low speed, 2-over speed, 3- retrograde, 4-run the red
light, 5-cross the lane line, 6- not according to the guide line, 7- intersection congest,
8- motor vehicle takes up the non-motor vehicle lane, 9- change the traffic lane
illegally, 10- motor vehicle against the rules occupy the special lane, 11- violation of
the ban, 12- Intersection park, 13-park during the green lights, 14-not comity the
pedestrians (illegal code: 1357), 15- Violation park, 16- Violation turn around,17-
occupy the Emergency Vehicle Lane, 18-right forbidden, 19- left forbidden, 20- cross
the yellow line, 21- not wearing a seatbelt*/
```

```
        BYTE byIllegalType;
        BYTE byVehicleLogoRecog; //please refer to VLR_VEHICLE_CLASS
        BYTE byVehicleSubLogoRecog; //please refer to VSB_VOLKSWAGEN_CLASS
        BYTE byRes2; //
        BYTE byCustomInfo[16];    //customized info
        BYTE byRes3[16];
}NET_DVR_VEHICLE_INFO, *LPNET_DVR_VEHICLE_INFO;
typedef struct tagNET_DVR_TIME_V30
{
    WORD wYear;
    BYTE byMonth;
    BYTE byDay;
    BYTE byHour;
    BYTE byMinute;
    BYTE bySecond;
    BYTE byRes;
    WORD wMilliSec;
    BYTE byRes1[2];
}NET_DVR_TIME_V30, *LPNET_DVR_TIME_V30;
// result
typedef struct tagNET_ITS_PLATE_RESULT
{
    DWORD     dwSize;         //Structure size
    DWORD     dwMatchNo;        //The match number, composed of vehicle serial
number, data type and lane number;
    BYTE      byGroupNum;   //The total number of picture groups (the number of
pictures continuously captured when one car passed)
    BYTE      byPicNo;        //The sequence number of the picture (if
byPicNo==byGroupNum, it means finished to receive the last picture; if byPicNo!
=byGroupNum, the picture will be deleted or reserved as needed)
    BYTE      bySecondCam; // whether captured by the second camera (e.g. the
vista camera of the vision and close-up snapshot, or the rear camera of front and
rear snapshot, will be used in special projects)
    BYTE      byFeaturePicNo; // which picture is taken as close-up view (it is used for
automatic detecting system of vehicle violation of traffic signal), and 0xff means not
take any one
    BYTE     byDriveChan;            // the lane that triggered snapshot
    BYTE     byVehicleType;         // Vehicle type, refer to VTR_RESULT
    BYTE      byDetSceneID; //detected scene ID[1,4], IPC is 0 by default
    BYTE        byVehicleAttribute; // 0-no additional Properties, 1- Yellow Label
Car(Banner),2- Dangerous goods vehicles;
    WORD       wIllegalType;             // Illegal type, definition of GB
    BYTE      byIllegalSubType[8];     // Sub type of illegal behavior
    BYTE       byPostPicNo;        // which picture is taken to record for intelligent
```

vehicle monitoring and recording system, 0xff means not take any one;
BYTE byChanIndex; // Channel number (Reserved)
WORD wSpeedLimit; // The upper limit of speed (valid when overspeed）
BYTE byRes2[2];
NET_DVR_PLATE_INFO struPlateInfo; // License plate information
NET_DVR_VEHICLE_INFO struVehicleInfo; // Vehicle information
BYTE byMonitoringSiteID[48]; // ID of monitoring point
BYTE byDeviceID[48]; // ID of the device
BYTE byDir; // Monitoring direction: 1- up-road, 2- down-road, 3- two-way, 4- from east to west, 5- from south to north, 6- from west to east, 7- from north to south, 8- else
BYTE byDetectType; // Detection type: 1- triggered by inductive coil, 2- triggered by video detection, 3- multi-frame recognition, 4- triggered by radar
// relevant lane direction type, please refer to ITC_RELA_LANE_DIRECTION_TYPE
//Act as the parameter of lane direction, corresponding to the relevant lane number;
BYTE byRelaLaneDirectionType;
BYTE byRes3; // reserved
//valid when wIllegalType is NULL. If the wIllegalType is not NULL, subject to wIllegalType.
DWORD dwCustomIllegalType; //illegal type definition(customized)
BYTE byRes4[9]; //reserved
BYTE byPilotSafebelt;//0-unknown,1- Fastened seat belt,2-without seat belt
BYTE byCopilotSafebelt;// 0-unknown,1- Fastened seat belt,2-without seat belt
BYTE byPilotSunVisor;//0- unknown,1- sun louver closed,2- sun louver open
BYTE byCopilotSunVisor;// 0- unknown,1- sun louver closed,2- sun louver open
BYTE byPilotCall;// 0- unknown, 1-don’t make a phone call,2-call up
//0- switch off, 1-non- switch off (Dedicated to the historical data in the camera after the match according to the black and white list, the flag of switch off success)
BYTE byBarrierGateCtrlType;
BYTE byAlarmDataType;//0-real time data, 1-history data
NET_DVR_TIME_V30 struSnapFirstPicTime;//time of the first picture captured (ms) DWORD dwIllegalTime;//illegal time of duration(ms) = time of the last picture captured - time of the first picture captured;
DWORD dwPicNum; // The number of pictures (different form icGroupNum, it is the number of pictures in this message)
NET_ITS_PICTURE_INFO struPicInfo[6]; // Picture information, up to 6 pitures
}NET_ITS_PLATE_RESULT, *LPNET_ITS_PLATE_RESULT;;

2.5.1.4 Remarks

NULL

2.5.2 Lisen

2.5.2.1 API

```
/************************************************
Function:           NET_DVR_StartListen_V30
Description: start listening and receive alarm information uploaded actively from device (muti-thread).
Input:      sLocalIP: Local IP
                wLocalPort: Local listening port number of PC, configured by user, should be consistent with that set in device
                fDataCallback: Callback function
                    lCommand: callback type, COMM_ITS_PLATE_RESULT
                  pAlarmer: alarmer info, NET_DVR_ALARMER
                    pAlarmInfo: alarm info, NET_ITS_PLATE_RESULT
                    dwBufLen: length of NET_ITS_PLATE_RESULT
                    pUser: User data
                pUserData: User data
Output:     null
Return value:           >=0: success and return listen handle, <0: failed
************************************************/
NET_DVR_API HPR_INT32 CALLBACK NET_DVR_StartListen_V30(
char *sLocalIP,
HPR_UINT16 wLocalPort,
MSGCallBack DataCallback,
HPR_VOIDPTR pUserData)

typedef void (CALLBACK *MSGCallBack)(
LONG lCommand,
NET_DVR_ALARMER *pAlarmer,
char *pAlarmInfo,
DWORD dwBufLen, void* pUser);
```

2.5.2.2 Parameter Definition

```
#define COMM_ITS_PLATE_RESULT                0x3050    // terminal picture uploading
```

lCommand	pAlarmer	pAlarmInfo	dwBufLen

COMM_ITS_PLATE_RESULT	NET_DVR_ALARMER	NET_ITS_PLATE_RESULT	Length of NET_ITS_PLATE_RESULT

2.5.2.3 Macro Definition and Structure

Please refer to NET_ITS_PLATE_RESULT

2.5.2.4 Remarks

NULL

3 Others

3.1 Device Type

NULL

3.2 Error code

NULL

3.3 Log protocol and types

```
//main type
#define MAJOR_ALARM                 0x1     //main alarm type
#define MAJOR_EXCEPTION             0x2     // major exception type
#define MAJOR_OPERATION             0x3     // major operation type
#define MAJOR_INFORMATION           0x4     //additional info
```

◆ Structure

```
//Date
typedef struct
{
      DWORD dwYear;        //year
      DWORD dwMonth;    //month
      DWORD dwDay;          //day
      DWORD dwHour;               //hour
      DWORD dwMinute;      //minute
      DWORD dwSecond;      //second
}INTER_TIME, *LPINTER_TIME;

typedef struct tagINTER_COMMON_APPEND_LOG
{
       DWORD dwParaType;//parameter type
      DWORD dwChannel;// channel number
      DWORD dwDiskNumber;//HDD number
       DWORD dwAlarmInPort;//alarm input port
```

```
        DWORD dwAlarmOutPort;//alarm output port
        DWORD dwInfoLen;      //length of log info
        char    sInfo[NORMAL_LOGLEN];//normal log with additional info
}INTER_COMMON_APPEND_LOG, *LPINTER_COMMON_APPEND_LOG;

typedef struct tagINTER_PDC_APPEND_LOG
{
        time_t  tPDCStopTime;                  //count the stop time
        DWORD    dwEnterNum;                    //number of entered people
        DWORD    dwLeaveNum;                   // number of left people
        BYTE byRes[NORMAL_LOGLEN];        //reserved
}INTER_PDC_APPEND_LOG, *LPINTER_PDC_APPEND_LOG;

typedef union tagINTER_APPEND_LOG_UNION
{
        INTER_COMMON_APPEND_LOG struCommonAppendLog; //log with additional
info
        INTER_PDC_APPEND_LOG           struPDCAppendLog;           //people   counting
statistics info
}INTER_APPEND_LOG_UNION, *LPINTER_APPEND_LOG_UNION;

typedef struct
{
        time_t tLogTime;                                  //time
        DWORD dwMajorType;                                //major type;
        DWORD dwMinorType;                        //minor type;
        BYTE sPanelUser[MAX_NAMELEN];      //panel user
        BYTE sNetUser[MAX_NAMELEN];        //network user
        U_IN_ADDR struRemoteHostAddr;             // remote host IP
        INTER_APPEND_LOG_UNION    uAppendLogInfo;//log with info union
}INTER_DVRLOG_V30, *LPINTER_DVRLOG_V30;
```

◆ Value

Macro Definition	Definition	Value
FILELISTOVER	File searching over	26
NEEDWAIT	Searching, please wait	25
RECVFILEINFO	Length of received file	27
NORMAL_LOGLEN	Max length of log info	4400
MAX_NAMELEN	Length of user name	16