报警事件

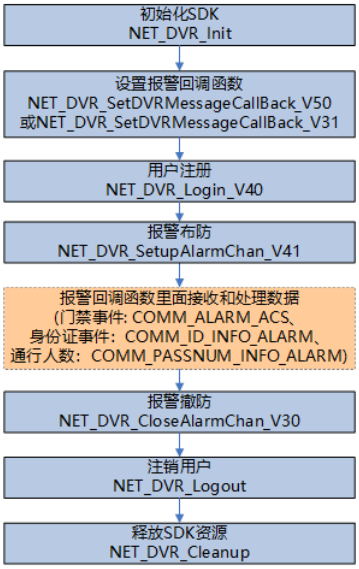
明眸门禁产品包括门禁主机、人员通道、明眸人证、梯控主机等，事件包含刷卡、刷脸、指纹、二维码认证结果和身份证刷卡事件以及测温信息等。

获取事件信息的方法有两种：1）报警布防，获取设备实时上传的事件信息；2）事件查询，事后查询设备本地存储的历史事件。

目录

一、报警布防

1.1 接口调用流程



- 初始化NET_DVR_Init接口在程序开始是调用，一个程序只需要调用一次。
- 用户注册即登录门口机，调用NET_DVR_Login_V40接口，每一台设备只需要登录一次。
- 报警布防方式实现方法：
  1) 先调用NET_DVR_SetDVRMessageCallBack_V50设置报警回调函数（V31接口也支持，不能使用V30接口），在SDK初始化之后即可以调用，多台设备对接时也只需要调用一次设置一个回调函数，回调函数里面接收数据之后可以通过报警设备信息(NET_DVR_ALARMER)中lUserID等参数判断区分设备。
  2) 每台设备分别登录，分别调用NET_DVR_SetupAlarmChan_V41进行布防，布防即建立设备跟客户端之间报警上传的连接通道，这样设备发生报警之后通过该连接上传报警信息，SDK在报警回调函数中接收和处理报警信息数据即可。
  3) 程序退出前或者不需要接收报警信息时调用NET_DVR_CloseAlarmChan_V30进行撤防，释放资源，此时连接断开，设备将不再上传报警信息。
- 刷卡、刷脸、指纹等认证方式对应门禁主机报警信息，回调函数中获取的报警类型(lCommand)为COMM_ALARM_ACS，报警信息(pAlarmInfo)对应结构体：NET_DVR_ACS_ALARM_INFO，其中pAcsEventInfoExtendV20指向一个NET_DVR_ACS_EVENT_INFO_EXTEND_V20结构体，里面包含测温温度信息。身份证刷卡事件对应门禁身份证刷卡信息，回调函数中获取的报警类型(lCommand)为COMM_ID_INFO_ALARM，报警信息(pAlarmInfo)对应结构体：NET_DVR_ID_CARD_INFO_ALARM，其中pIDCardInfoExtend指向一个NET_DVR_ID_CARD_INFO_EXTEND结构体，里面包含测温温度信息。报警信息结构体中dwMajor、dwMinor表示报警主类型和次类型，通过这两个参数值来判断实际的事件，比如是刷脸还是刷卡、成功还是失败，对应不同的事件次类型。
- 退出程序时调用NET_DVR_Logout注销设备，每一台设备调用一次。最后调用NET_DVR_Cleanup释放SDK所有资源。

1.2 示例代码

```
#include <stdio.h>
#include <iostream>
#include "Windows.h"
#include "HCNetSDK.h"
using namespace std;

int iNum = 0; //已保存的图片个数
void CALLBACK MessageCallback(LONG lCommand, NET_DVR_ALARMER *pAlarmer, char *pAlarmInfo, DWORD dwBufLen, void* pUser)
{
    switch (lCommand)
```

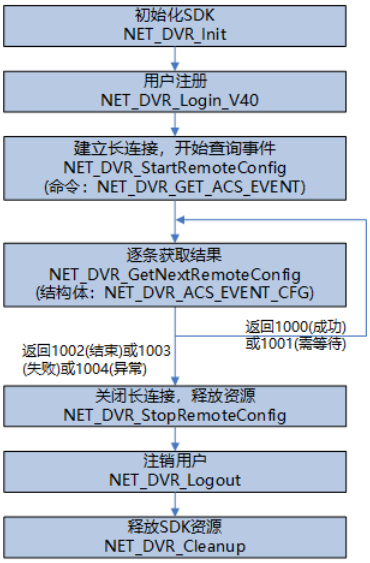
```
{
case COMM_ALARM_ACS: //门禁主机报警信息
{
    NET_DVR_ACS_ALARM_INFO struAcsAlarmInfo = { 0 };
    memcpy(&struAcsAlarmInfo, pAlarmInfo, sizeof(NET_DVR_ACS_ALARM_INFO));
    printf("门禁主机报警信息[0x5002]: struTim{%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d}, dwMajor[0x%x], dwMinor[0x%x], byCardNo[%s], dwEmployeeNo[%d], dwCardReaderNo[%d]\n",
        struAcsAlarmInfo.struTime.dwYear, struAcsAlarmInfo.struTime.dwMonth,struAcsAlarmInfo.struTime.dwDay, struAcsAlarmInfo.struTime.dwHour,
        struAcsAlarmInfo.struTime.dwMinute, struAcsAlarmInfo.struTime.dwSecond, struAcsAlarmInfo.dwMajor, struAcsAlarmInfo.dwMinor,
        struAcsAlarmInfo.struAcsEventInfo.byCardNo, struAcsAlarmInfo.struAcsEventInfo.dwEmployeeNo, struAcsAlarmInfo.struAcsEventInfo.dwCardReaderNo);

    //扩展信息，包含以人为中心下发人员触发事件中的工号参数
    if (struAcsAlarmInfo.byAcsEventInfoExtend == 1)
    {
        NET_DVR_ACS_EVENT_INFO_EXTEND  struAcsEventInfoExtend = { 0 };
        memset(&struAcsEventInfoExtend, 0, sizeof(struAcsEventInfoExtend));
        memcpy(&struAcsEventInfoExtend, struAcsAlarmInfo.pAcsEventInfoExtend, sizeof(struAcsEventInfoExtend));

        printf("门禁扩展事件信息: dwFrontSerialNo[%d], byUserType[%d], byEmployeeNo[%s]\n", struAcsEventInfoExtend.dwFrontSerialNo,
            struAcsEventInfoExtend.byUserType, (char *)struAcsEventInfoExtend.byEmployeeNo);
    }

    //扩展信息，包含人体测温温度数据
    if (struAcsAlarmInfo.byAcsEventInfoExtendV20 == 1)
    {
        NET_DVR_ACS_EVENT_INFO_EXTEND_V20 struAcsEventInfoExtendV20 = { 0 };
        memcpy(&struAcsEventInfoExtendV20, struAcsAlarmInfo.pAcsEventInfoExtendV20, sizeof(struAcsEventInfoExtendV20));

        if (struAcsEventInfoExtendV20.byRemoteCheck != 0)
        {
            //需要远程核验
            printf("remote check:%d\n", struAcsEventInfoExtendV20.byRemoteCheck);
        }
        if (struAcsEventInfoExtendV20.fCurrTemperature != 0)
        {
            printf("temperature[%f]fX[%f]fY[%f]byThermometryUnit[%d]byIsAbnomalTemperature[%d]\n",
                struAcsEventInfoExtendV20.fCurrTemperature, struAcsEventInfoExtendV20.struRegionCoordinates.fX,
                struAcsEventInfoExtendV20.struRegionCoordinates.fY, struAcsEventInfoExtendV20.byThermometryUnit,
                struAcsEventInfoExtendV20.byIsAbnomalTemperature);
        }
    }

    //保存报警抓拍图片
    if (struAcsAlarmInfo.dwPicDataLen > 0 && struAcsAlarmInfo.pPicData != NULL)
    {
        char cFilename[256] = { 0 };

        char chTime[128];
        sprintf(chTime, "%4.4d%2.2d%2.2d%2.2d%2.2d%2.2d", struAcsAlarmInfo.struTime.dwYear, struAcsAlarmInfo.struTime.dwMonth,
            struAcsAlarmInfo.struTime.dwDay, struAcsAlarmInfo.struTime.dwHour, struAcsAlarmInfo.struTime.dwMinute,
            struAcsAlarmInfo.struTime.dwSecond);

        sprintf(cFilename, "COMM_ALARM_ACS_CapPic[%d][%s].jpg", pAlarmer->lUserID, chTime);

        FILE* fSnapPicPlate = fopen(cFilename, "wb");
        fwrite(struAcsAlarmInfo.pPicData, struAcsAlarmInfo.dwPicDataLen, 1, fSnapPicPlate);
        fclose(fSnapPicPlate);
        iNum++;
    }
}
break;
case COMM_ID_INFO_ALARM: //门禁主机报警信息
{
    NET_DVR_ID_CARD_INFO_ALARM struIDCardInfo = { 0 };
    memcpy(&struIDCardInfo, pAlarmInfo, sizeof(NET_DVR_ID_CARD_INFO_ALARM));
    printf("门禁身份证刷卡信息[0x5200]: struTim{%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d}, dwMajor[0x%x], dwMinor[0x%x], byName[%s], byIDNum[%s], dwCardReaderNo[%d]\n",
        struIDCardInfo.struSwipeTime.wYear, struIDCardInfo.struSwipeTime.byMonth, struIDCardInfo.struSwipeTime.byDay, struIDCardInfo.struSwipeTime.byHour,
        struIDCardInfo.struSwipeTime.byMinute, struIDCardInfo.struSwipeTime.bySecond, struIDCardInfo.dwMajor, struIDCardInfo.dwMinor,
        (char *)struIDCardInfo.struIDCardCfg.byName, (char *)struIDCardInfo.struIDCardCfg.byIDNum, struIDCardInfo.dwCardReaderNo);

    //扩展信息，包含人体测温温度数据
    if (struIDCardInfo.byIDCardInfoExtend == 1)
    {
        NET_DVR_ID_CARD_INFO_EXTEND struIDCardExtendV20 = { 0 };
        memcpy(&struIDCardExtendV20, struIDCardInfo.pIDCardInfoExtend, sizeof(struIDCardExtendV20));

        if (struIDCardExtendV20.byRemoteCheck != 0)
        {
            //需要远程核验
            printf("remote check:%d\n", struIDCardExtendV20.byRemoteCheck);
        }
        if (struIDCardExtendV20.fCurrTemperature != 0)
        {
```

```
            printf("temperature[%f]fX[%f]fY[%f]byThermometryUnit[%d]byIsAbnomalTemperature[%d]\n",
                struIDCardExtendV20.fCurrTemperature, struIDCardExtendV20.struRegionCoordinates.fX,
                struIDCardExtendV20.struRegionCoordinates.fY, struIDCardExtendV20.byThermometryUnit,
                struIDCardExtendV20.byIsAbnomalTemperature);
        }
    }

    //保存身份证图片数据大小
    if (struIDCardInfo.dwPicDataLen > 0 && struIDCardInfo.pPicData != NULL)
    {
        char cCardFilename[256] = { 0 };

        char chTime[128];
        sprintf(chTime, "%4.4d%2.2d%2.2d%2.2d%2.2d%2.2d", struIDCardInfo.struSwipeTime.wYear, struIDCardInfo.struSwipeTime.byMonth,
            struIDCardInfo.struSwipeTime.byDay, struIDCardInfo.struSwipeTime.byHour,
            struIDCardInfo.struSwipeTime.byMinute, struIDCardInfo.struSwipeTime.bySecond);

        sprintf(cCardFilename, "COMM_ID_INFO_ALARM_IDCardPic[%d][%s].jpg", pAlarmer->lUserID, chTime);

        FILE* fSnapPicPlate = fopen(cCardFilename, "wb");
        fwrite(struIDCardInfo.pPicData, struIDCardInfo.dwPicDataLen, 1, fSnapPicPlate);
        fclose(fSnapPicPlate);
        iNum++;
    }

    //保存抓拍图片数据大小
    if (struIDCardInfo.dwCapturePicDataLen  > 0 && struIDCardInfo.pCapturePicData != NULL)
    {
        char cFilename[256] = { 0 };

        char chTime[128];
        sprintf(chTime, "%4.4d%2.2d%2.2d%2.2d%2.2d%2.2d", struIDCardInfo.struSwipeTime.wYear, struIDCardInfo.struSwipeTime.byMonth,
            struIDCardInfo.struSwipeTime.byDay, struIDCardInfo.struSwipeTime.byHour,
            struIDCardInfo.struSwipeTime.byMinute, struIDCardInfo.struSwipeTime.bySecond);

        sprintf(cFilename, "COMM_ID_INFO_ALARM_CapPic[%d][%s].jpg", pAlarmer->lUserID, chTime);

        FILE* fSnapPicPlate = fopen(cFilename, "wb");
        fwrite(struIDCardInfo.pCapturePicData, struIDCardInfo.dwCapturePicDataLen, 1, fSnapPicPlate);
        fclose(fSnapPicPlate);
        iNum++;
    }
  }
  break;
  default:
    printf("其他报警，报警信息类型: %d\n", lCommand);
    break;
  }

  return;
}

void main() {
  //---------------------------------------
  // 初始化
  NET_DVR_Init();
  //设置连接时间与重连时间
  NET_DVR_SetConnectTime(2000, 1);
  NET_DVR_SetReconnect(10000, true);

  //---------------------------------------
  // 注册设备
  LONG lUserID;

  //登录参数，包括设备地址、登录用户、密码等
  NET_DVR_USER_LOGIN_INFO struLoginInfo = { 0 };
  struLoginInfo.bUseAsynLogin = 0; //同步登录方式
  strcpy(struLoginInfo.sDeviceAddress, "10.17.36.2"); //设备IP地址
  struLoginInfo.wPort = 8000; //设备服务端口
  strcpy(struLoginInfo.sUserName, "admin"); //设备登录用户名
  strcpy(struLoginInfo.sPassword, "abcd1234"); //设备登录密码

  //设备信息, 输出参数
  NET_DVR_DEVICEINFO_V40 struDeviceInfoV40 = { 0 };

  lUserID = NET_DVR_Login_V40(&struLoginInfo, &struDeviceInfoV40);
  if (lUserID < 0)
  {
    printf("Login failed, error code: %d\n", NET_DVR_GetLastError());
    NET_DVR_Cleanup();
    return;
  }
```

```
//设备登录成功，获取设备字符集编码类型
printf("Login successfully, byCharEncodeType: %d\n", struDeviceInfoV40.byCharEncodeType);


//设置报警回调函数
NET_DVR_SetDVRMessageCallBack_V50(0, MessageCallback, NULL);

//启用布防
LONG lHandle;
NET_DVR_SETUPALARM_PARAM  struAlarmParam = { 0 };
struAlarmParam.dwSize = sizeof(struAlarmParam);
//不需要设置其他报警布防参数，不支持

lHandle = NET_DVR_SetupAlarmChan_V41(lUserID, &struAlarmParam);
if (lHandle < 0)
{
    printf("NET_DVR_SetupAlarmChan_V41 failed, error code:  %d\n", NET_DVR_GetLastError());
    NET_DVR_Logout(lUserID);
    NET_DVR_Cleanup();
    return;
}

//事件信息在回调函数里面获取
//控制台输入q退出程序，否则一直运行
char c = 0;
while ('q' != c)
{
    printf("input 'q' to quit\n");
    printf("input: ");
    scanf("%c", &c);
}

//----------------------------------------
//退出程序

//撤销布防上传通道
if (!NET_DVR_CloseAlarmChan_V30(lHandle))
{
    printf("NET_DVR_CloseAlarmChan_V30 failed, error code: %d\n", NET_DVR_GetLastError());
    NET_DVR_Logout(lUserID);
    NET_DVR_Cleanup();
    return;
}

//注销用户
NET_DVR_Logout(lUserID);

//释放SDK资源
NET_DVR_Cleanup();
return;
}
```

二、事件查询

2.1 接口调用流程



- 初始化NET_DVR_Init接口在程序开始是调用，一个程序只需要调用一次。
- 用户注册即登录门口机，调用NET_DVR_Login_V40接口，每一台设备只需要登录一次。
- 查询事件是指查找设备本地存储的历史事件，先调用NET_DVR_StartRemoteConfig(命令: NET_DVR_GET_ACS_EVENT)建立长连接，指定查找条件(NET_DVR_ACS_EVENT_COND，然后循环调用NET_DVR_GetNextRemoteConfig逐条获取查找的结果信息(NET_DVR_ACS_EVENT_CFG)，查询结束之后调用NET_DVR_StopRemoteConfig关闭长连接，释放资源。
- 退出程序时调用NET_DVR_Logout注销设备，每一台设备调用一次。最后调用NET_DVR_Cleanup释放SDK所有资源。

2.2 示例代码

```
#include "stdafx.h"
#include "include\HCNetSDK.h"
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <string>
using namespace std;

int DoLogin();
void GetEventInfo(int lLoginID);

int _tmain(int argc, _TCHAR* argv[])
{

   //初始化
   NET_DVR_Init();

   //登录设备
   int lUserID = DoLogin();
   if (lUserID < 0)
   {
      return 0;
   }

   //查询事件信息
   GetEventInfo(lUserID);

   Sleep(500);

   //注销登录
   NET_DVR_Logout(lUserID);

   //反初始化，释放资源
   NET_DVR_Cleanup();
   return 1;
}

//设备登录
int DoLogin()
{
   //注册参数
   NET_DVR_USER_LOGIN_INFO struLoginInfo = { 0 };
   struLoginInfo.bUseAsynLogin = 0; //同步登录方式
   strcpy(struLoginInfo.sDeviceAddress, "10.17.36.2"); //设备IP地址
   struLoginInfo.wPort = 8000; //设备服务端口
   strcpy(struLoginInfo.sUserName, "admin"); //设备登录用户名
```

```
    strcpy(struLoginInfo.sPassword, "abcd1234"); //设备登录密码

    //设备参数结构体
    NET_DVR_DEVICEINFO_V40 struDeviceInfoV40;
    memset(&struDeviceInfoV40, 0, sizeof(NET_DVR_DEVICEINFO_V40));

    //设备注册
    int lLoginID = NET_DVR_Login_V40(&struLoginInfo, &struDeviceInfoV40);
    if (lLoginID< 0)
    {
        cout << ("Login failed, error code: \n", NET_DVR_GetLastError()) << endl;
        NET_DVR_Cleanup();
    }
    if (lLoginID >= 0)
    {
        cout << "注册成功" << endl;

        //设备登录成功，获取设备字符集编码类型
        cout << "设备字符集编码类型: " << struDeviceInfoV40.byCharEncodeType << endl;
    }
    return lLoginID;
}

// 获取事件信息
void GetEventInfo(int lLoginID)
{
    //门禁事件查询条件
    NET_DVR_ACS_EVENT_COND struEventCond = {0};
    struEventCond.dwSize = sizeof(struEventCond);
    struEventCond.dwMajor = 0x5; //5表示查询事件，0表示查询全部
    struEventCond.dwMajor = 0; //查询全部

    //查询开始时间
    struEventCond.struStartTime.dwYear = 2021;
    struEventCond.struStartTime.dwMonth = 1;
    struEventCond.struStartTime.dwDay = 26;
    struEventCond.struStartTime.dwHour = 0;
    struEventCond.struStartTime.dwMinute = 0;
    struEventCond.struStartTime.dwSecond = 0;

    //查询结束时间
    struEventCond.struEndTime.dwYear = 2021;
    struEventCond.struEndTime.dwMonth = 1;
    struEventCond.struEndTime.dwDay = 26;
    struEventCond.struEndTime.dwHour = 23;
    struEventCond.struEndTime.dwMinute = 59;
    struEventCond.struEndTime.dwSecond = 59;
    struEventCond.byTimeType = 0;  //时间类型：0-设备本地时间（默认），1-UTC时间（struStartTime和struEndTime的时间）

    struEventCond.byPicEnable = 0; //是否带图片，0-不带图片，1-带图片

    int m_lSetCardCfgHandle = -1;

    m_lSetCardCfgHandle = NET_DVR_StartRemoteConfig(lLoginID, NET_DVR_GET_ACS_EVENT, &struEventCond, sizeof(struEventCond), NULL, NULL);
    if (m_lSetCardCfgHandle == -1)
    {
        cout << ("建立查询事件长连接失败，错误码为:", NET_DVR_GetLastError()) << endl;
        return;
    }
    else
    {
        cout << "建立查询事件长连接成功！" << endl;
    }

    //逐条获取查询的结果
    NET_DVR_ACS_EVENT_CFG struEventCfg = { 0 };

    while (true)
    {
        int dwState = NET_DVR_GetNextRemoteConfig(m_lSetCardCfgHandle, &struEventCfg, sizeof(struEventCfg));
        if (dwState == -1)
        {
            cout << "NET_DVR_GetNextRemoteConfig获取事件调用失败，错误码：" << NET_DVR_GetLastError() << endl;
            break;
        }
        else if (dwState == NET_SDK_CONFIG_STATUS_NEEDWAIT)
        {
            cout << "配置等待" << endl;
            Sleep(4);
            continue;
        }
        else if (dwState == NET_SDK_CONFIG_STATUS_FAILED)
```

```
        {
            cout << "获取事件失败" << endl;
            break;
        }
        else if (dwState == NET_SDK_CONFIG_STATUS_EXCEPTION)
        {
            cout << "获取事件异常" << endl;
            break;
        }
        else if (dwState == NET_SDK_CONFIG_STATUS_SUCCESS){
            cout << "获取事件成功\n" << endl;
            printf("获取的门禁主机报警信息: struTim{%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d}, dwMajor[0x%x], dwMinor[0x%x], byCardNo[%s], dwCardReaderNo[%d], dwEmployeeNo[%d], byEmployeeNo[%s]\n",
                struEventCfg.struTime.dwYear, struEventCfg.struTime.dwMonth, struEventCfg.struTime.dwDay, struEventCfg.struTime.dwHour,
                struEventCfg.struTime.dwMinute, struEventCfg.struTime.dwSecond, struEventCfg.dwMajor, struEventCfg.dwMinor,
                struEventCfg.struAcsEventInfo.byCardNo, struEventCfg.struAcsEventInfo.dwCardReaderNo, struEventCfg.struAcsEventInfo.dwEmployeeNo,
                (char *)struEventCfg.struAcsEventInfo.byEmployeeNo);
            continue;

        }
        else if (dwState == NET_SDK_CONFIG_STATUS_FINISH) {
            cout << "获取人员参数完成" << endl;
            break;
        }
    }
    if (!NET_DVR_StopRemoteConfig(m_lSetCardCfgHandle)){
        cout << "NET_DVR_StopRemoteConfig接口调用失败，错误码：" << NET_DVR_GetLastError() << endl;
    }
    else{
        cout << "NET_DVR_StopRemoteConfig接口成功" << endl;
    }
}
```