

---

# 门禁产品 集成指导手册

---

# 目录

1. 简介 .....	4
1.1 编写目的 .....	4
1.2 术语和缩写 .....	4
2. 集成流程介绍 .....	4
2.1 整体流程 .....	4
2.1.1 权限管理 .....	4
2.1.2 远程控制 .....	5
2.1.3 事件管理 .....	5
2.2 卡管理 .....	5
2.2.1 功能介绍 .....	5
2.2.2 功能示例 .....	5
2.2.3 集成流程 .....	6
2.2.4 备注 .....	7
2.3 指纹管理 .....	8
2.3.1 功能介绍 .....	8
2.3.2 功能示例 .....	8
2.3.3 集成流程 .....	8
2.3.4 备注 .....	11
2.4 人脸管理 .....	11
2.4.1 功能介绍 .....	12
2.4.2 功能示例 .....	12
2.4.3 集成流程 .....	12
2.4.4 备注 .....	14
2.5 远程控门 .....	14
2.5.1 功能介绍 .....	14
2.5.2 功能示例 .....	15
2.5.3 集成流程 .....	15
2.5.4 备注 .....	15
2.6 设备事件主动上传（布防） .....	16
2.6.1 功能介绍 .....	16
2.6.2 功能示例 .....	16
2.6.3 集成流程 .....	16
2.6.4 备注 .....	17
2.7 主动获取设备事件 .....	17
2.7.1 功能介绍 .....	17
2.7.2 功能示例 .....	17
2.7.3 集成流程 .....	18
2.7.4 备注 .....	19
2.8 卡权限计划模板管理 .....	19
2.8.1 功能介绍 .....	19

2.8.2	功能示例.....	19
2.8.3	集成流程.....	20
2.8.4	备注.....	20
<b>3.</b>	<b>接口附录.....</b>	<b>21</b>
3.1	卡管理.....	21
3.1.1	卡获取（获取所有）.....	21
3.1.2	卡获取（按条件获取）.....	23
3.1.3	卡下发.....	26
3.1.4	卡删除.....	29
3.2	指纹管理.....	31
3.2.1	指纹获取.....	31
3.2.2	指纹下发.....	33
3.2.3	指纹删除.....	36
3.2.4	指纹采集.....	39
3.3	人脸管理.....	41
3.3.1	人脸获取.....	41
3.3.2	人脸下发.....	43
3.3.3	人脸删除.....	46
3.3.4	人脸采集.....	48
3.4	远程控门.....	50
3.4.1	接口函数.....	50
3.5	设备事件主动上传（布防）.....	50
3.5.1	接口函数.....	50
3.5.2	命令码.....	52
3.5.3	宏定义及结构体.....	52
3.5.4	备注.....	53
3.6	主动获取设备事件.....	53
3.6.1	接口函数.....	53
3.6.2	命令码.....	55
3.6.3	宏定义及结构体.....	55
3.7	卡权限计划模板管理.....	56
3.7.1	卡权限周计划配置.....	57
3.7.2	卡权限假日计划配置.....	59
3.7.3	卡权限假日组配置.....	61
3.7.4	卡权限计划模板配置.....	62
3.8	门禁事件主次类型.....	64
3.9	错误码表.....	68
<b>4.</b>	<b>FAQ.....</b>	<b>69</b>

---

## 1. 简介

### 1.1 编写目的

此文档制定了产品线项目开发中门禁产品接入网络功能的集成说明；

阅读人员主要是：

应用客户端，或者通用平台，行业平台；技术支持人员，测试人员。

### 1.2 术语和缩写

术语/缩写	含 义
HCNetSDK	海康网络功能软件开发包

## 2. 集成流程介绍

### 2.1 整体流程

#### 2.1.1 权限管理

权限管理的整体流程如下图所示：

- 1.卡管理和计划模板管理之间通过计划模板编号（CardRightPlan）关联；
- 2.卡管理和指纹管理、人脸管理之间通过卡号（CardNo）关联；
- 3.在下发人员权限时，首先要下发卡权限，然后才能下发指纹权限、人脸权限（如实际使用场景中  
没有卡号的需求，也需要虚拟一个卡号下发（要保证卡号在设备内是唯一的））；
- 4.在删除人员权限时，首先要删除指纹权限、人脸权限，然后再删除卡权限。

注：计划模板编号 1 为人脸门禁设备的默认计划模板，为 24 小时全天有权限（如没有特殊权限控制要求，卡管理可以直接使用计划模板编号 1）。

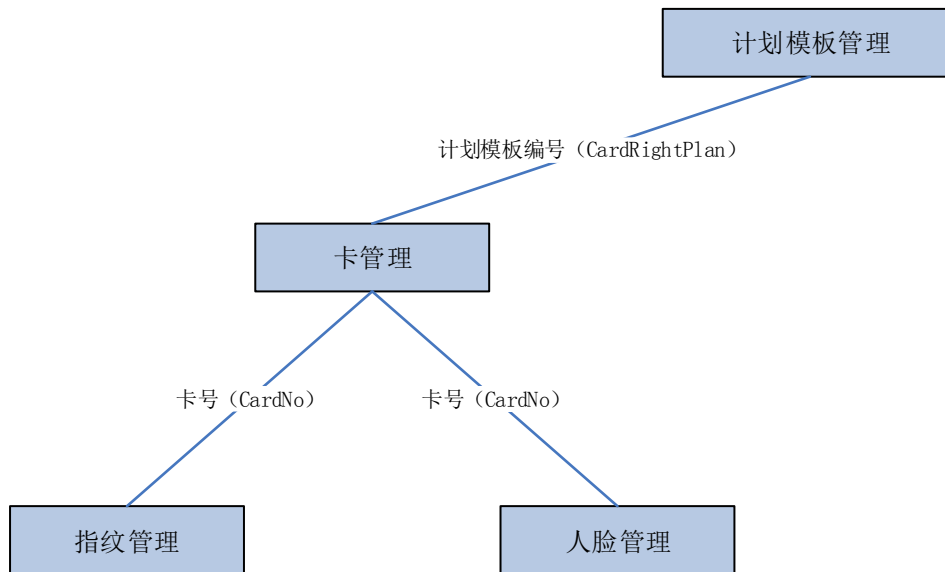


图 1 权限管理

### 2.1.2 远程控制

远程控制主要为远程控制门的开、关、常开、常闭。此处不做重点介绍，详情请见远程控门章节。

### 2.1.3 事件管理

事件管理主要分为设备事件主动上传（布防）、主动获取设备事件两种方式：

- 1.设备事件主动上传（布防）实时性较好，在门禁设备触发相关事件（如：刷卡开门、人脸认证通过等）时，会立即上报相关事件给平台；
- 2.主动获取设备事件用于事后查询事件记录，但要注意，由于门禁设备储存空间有限，只能存储固定条数（如 10W 条）的事件。

## 2.2 卡管理

### 2.2.1 功能介绍

用于下发及获取设备卡及人员相关信息：如卡号、工号、姓名、卡权限计划模板等相关信息。

### 2.2.2 功能示例

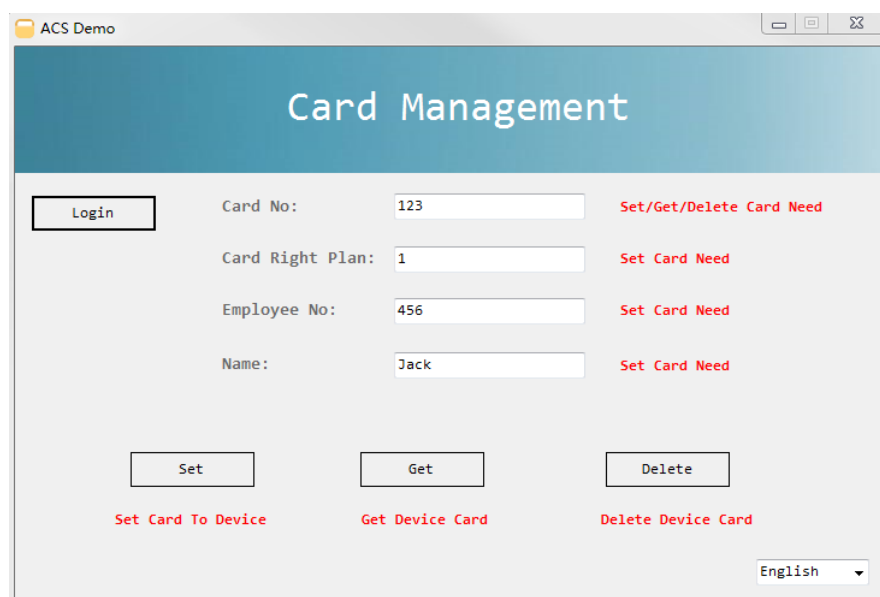
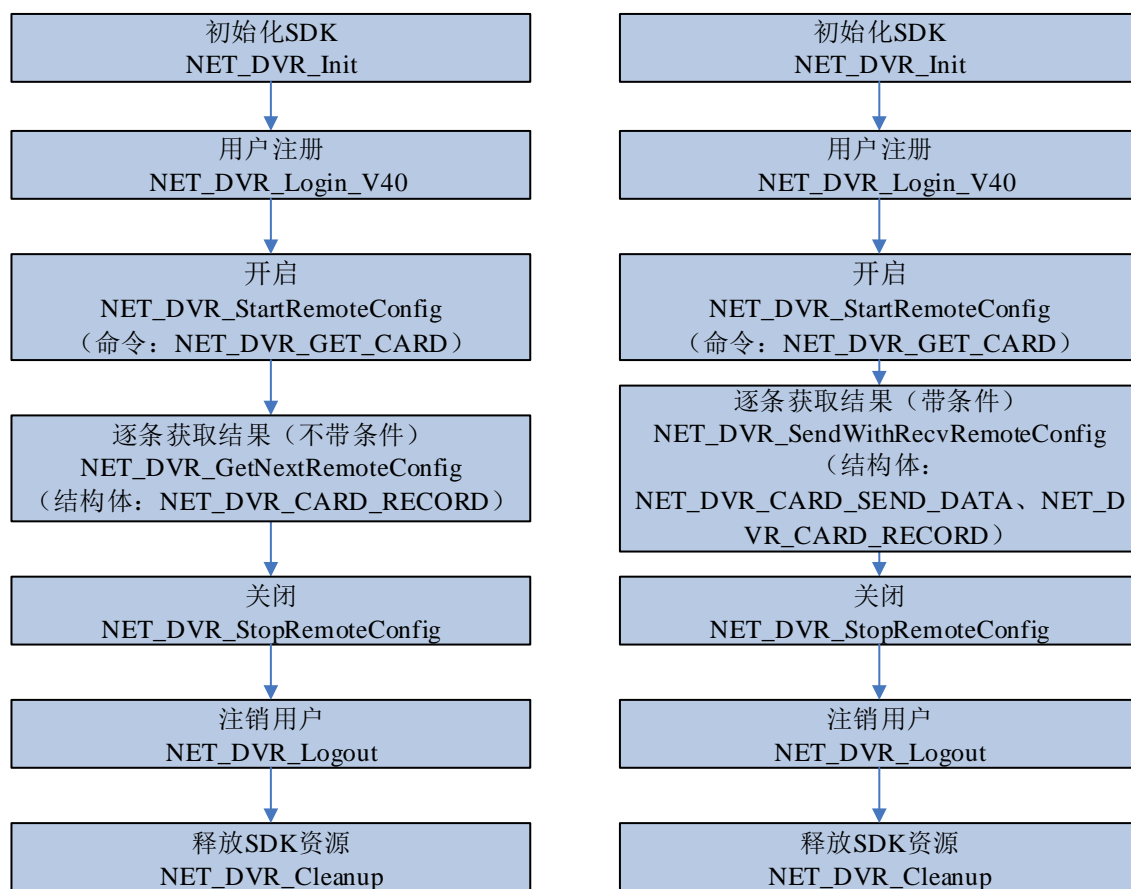


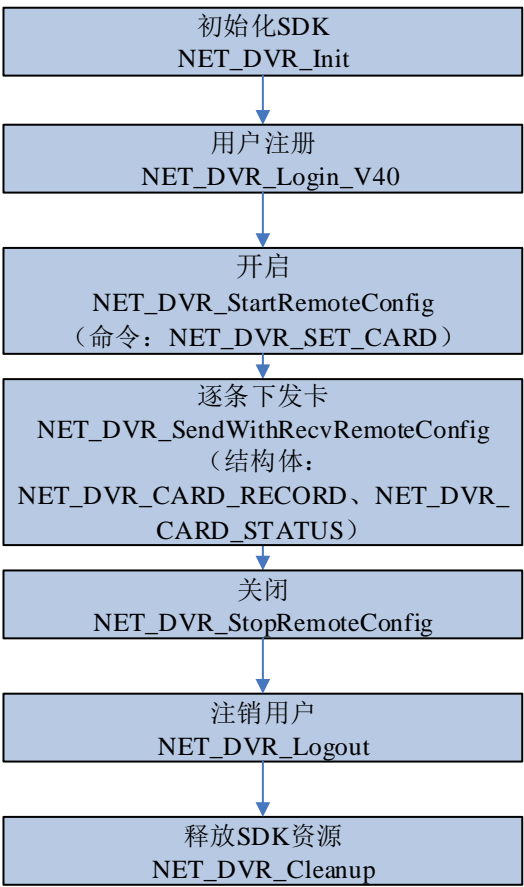
图 2 卡管理示例

## 2.2.3 集成流程

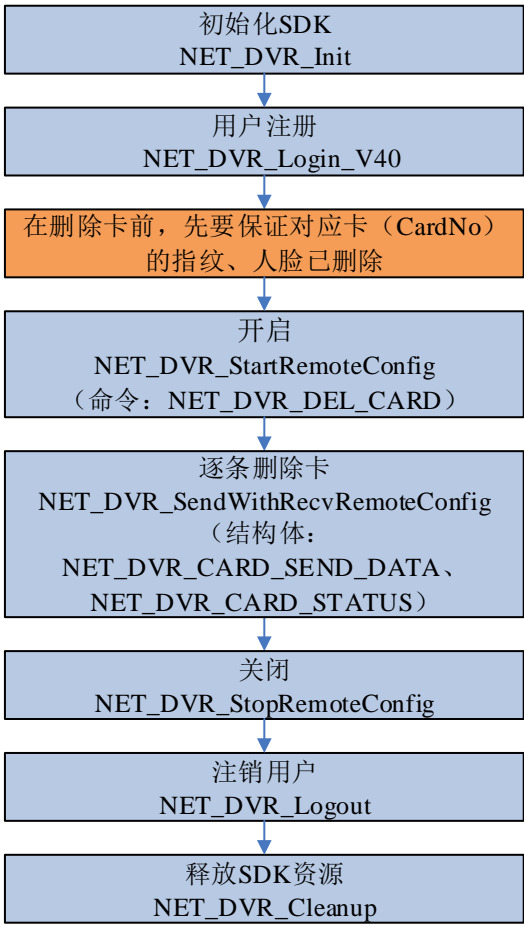


流程 1 卡获取（获取所有）

流程 2 卡获取（按条件获取）



流程 3 卡下发



流程 4 卡删除

## 2.2.4 备注

在下发卡时:

1.卡号的整型值不能重复 (比如不能同时含有 1 和 01 两种卡号), 这两个卡号对于设备来说属于相同的卡号;

2.要注意: NET\_DVR\_CARD\_RECORD 中不同的卡号 (byCardNo) 的工号 (dwEmployeeNo) 要是唯一的, 如对于不同的卡号 byCardNo, 设置相同的 dwEmployeeNo, 设备是会报错的 (NET\_ERR\_NOT\_SUPPORT\_ONE\_MORE\_CARD);

3.计划模板 (CardRightPlan), 可使用默认计划模板 1 (该模板全天 24 小时有权限)。如不配置 CardRightPlan 字段, 则在所有时间段都没有权限。如需要控制在某一时间段内的权限, 可参考卡权限计划模板管理流程;

4.如返回的 NET\_DVR\_CARD\_STATUS 中 byStatus 为 0-失败, 则需要解析 dwErrorCode 字段 (如下所示) 来确认具体的错误信息, 并根据错误信息排查相关问题。

#define NET\_ERR\_ILLEGAL\_CARDNO 1904 //卡号错误 (卡号不符合门禁设备要求, 请检测卡号是否出错)

#define NET\_ERR\_DEVICE\_CARD\_FULL 1908 //卡已满 (门禁设备存储的卡号已达上限, 不能再下发其他卡号了)

---

#define NET_ERR_DOOR_OVER_LIMIT (门编号超出门禁设备实际支持的数目)	1917 //门超出设备最大能力
#define NET_ERR_NOT_SUPPORT_ONE_MORE_CARD (卡号下发的工号重复)	1920 //不支持一人多卡(不同卡号下发的工号重复)

## 2.3 指纹管理

### 2.3.1 功能介绍

指纹获取：从门禁设备中获取某张卡所关联的指纹信息；

指纹下发：下发某张卡所关联的指纹信息到门禁设备；

指纹删除：删除门禁设备中的指纹信息；

指纹采集：使用门禁设备实时采集设备前的指纹到平台，该指纹可用于指纹下发。

### 2.3.2 功能示例

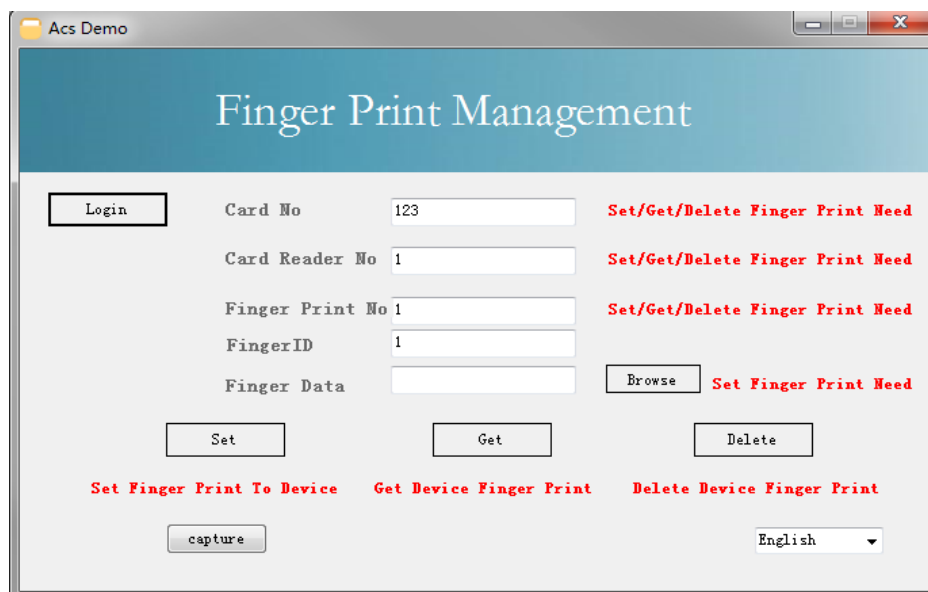
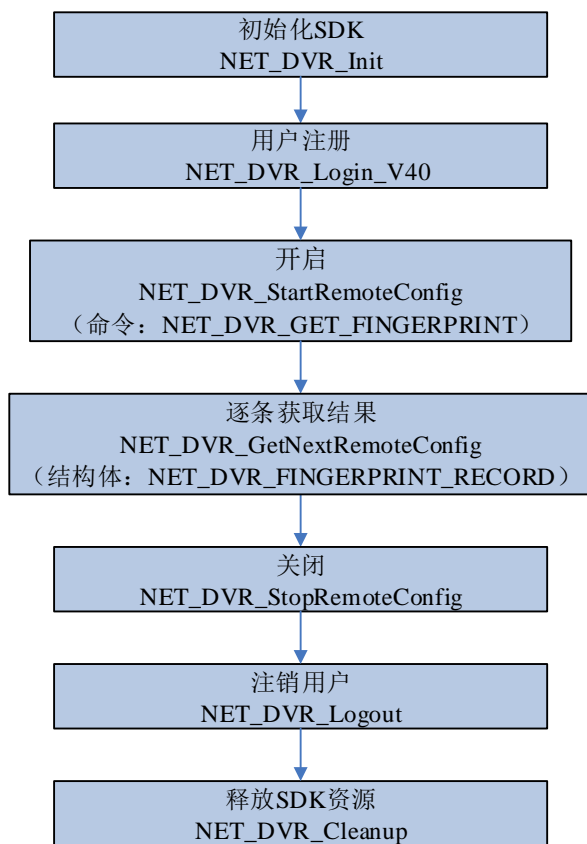


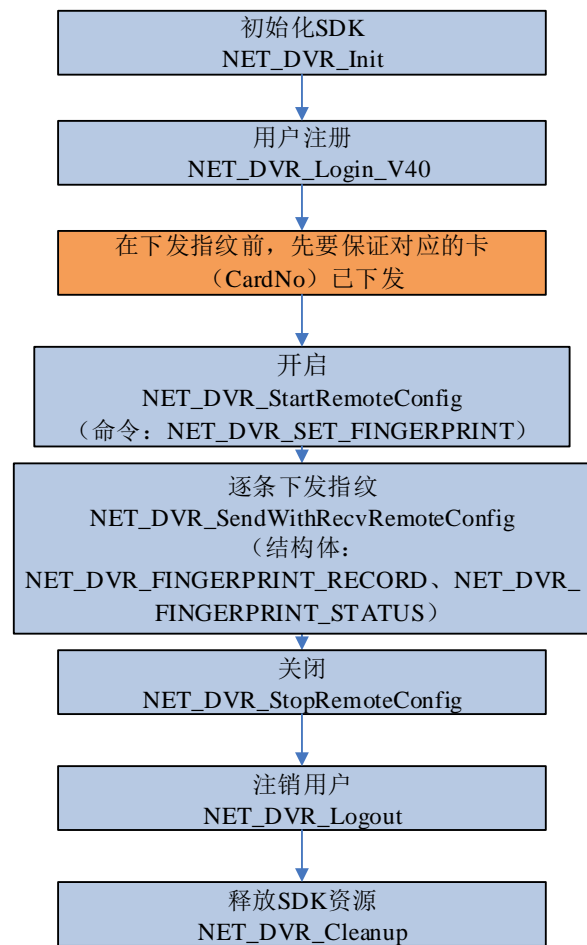
图 3 指纹管理示例

### 2.3.3 集成流程

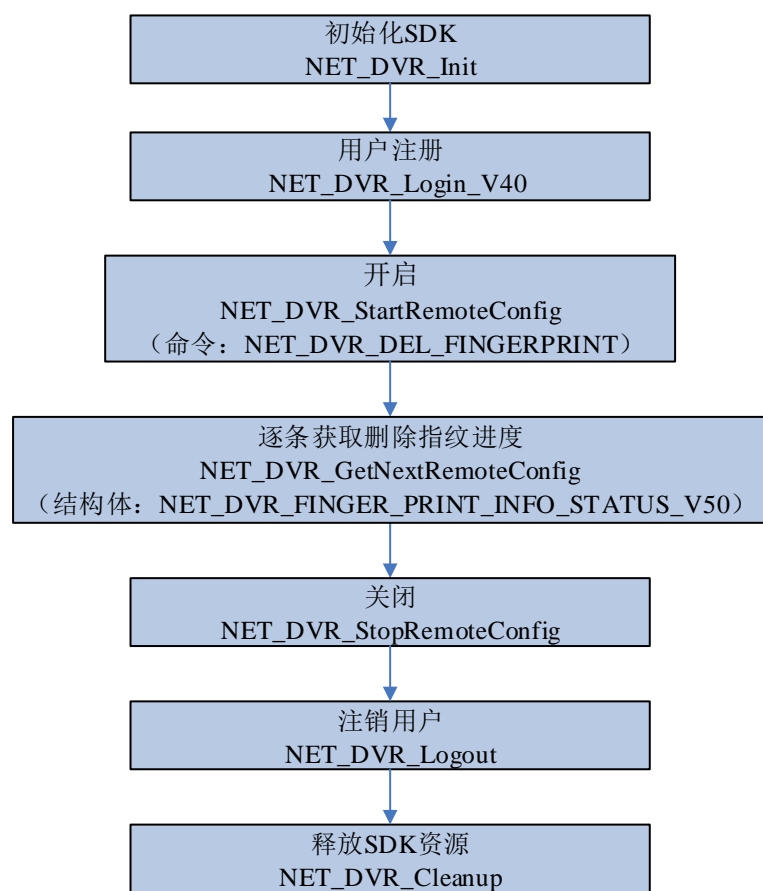




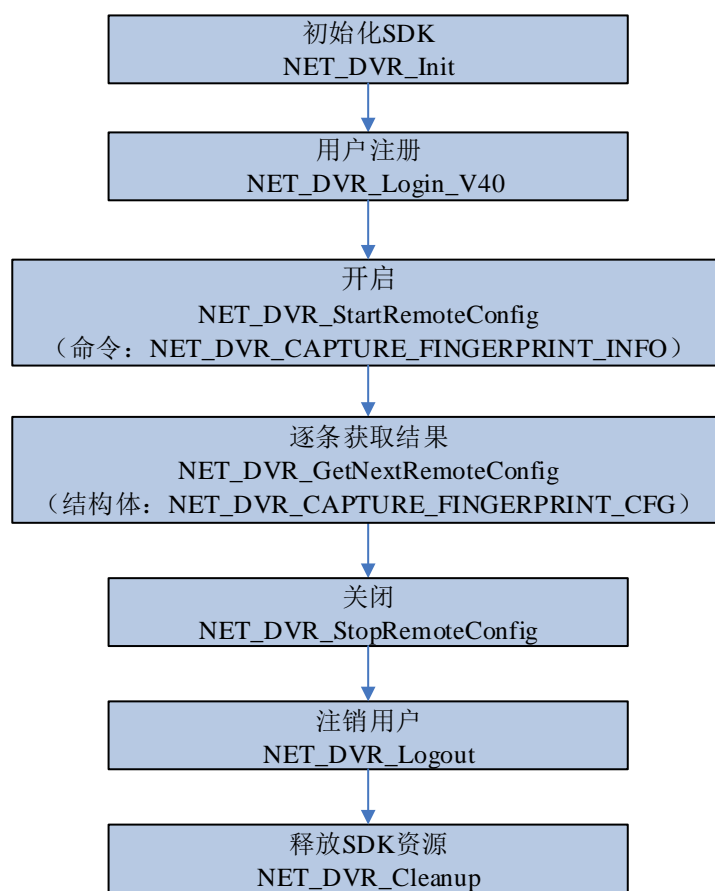
流程 5 指纹获取



流程 6 指纹下发



流程 7 指纹删除



流程 8 指纹采集

### 2.3.4 备注

- 1.由于卡管理中包含了人员相关信息，故：在下发指纹前，需要先下发卡；在删除卡前，需要先删除指纹；
- 2.人员和指纹之间通过卡号进行关联，并确定人员的唯一性；
- 3.在下发指纹时，每次调用指纹下发流程，可逐条下发多枚指纹（多次调用 NET\_DVR\_SendWithRecvRemoteConfig 接口），数目没有限制；
- 4.在获取指纹时，每次调用指纹获取流程最多只能获取某张卡所关联的指纹（1 张卡最多关联 10 枚指纹），如果想要再获取其他卡的指纹，则需要重新执行该流程（指的是从 NET\_DVR\_StartRemoteConfig、NET\_DVR\_GetNextRemoteConfig 到 NET\_DVR\_StopRemoteConfig 的流程）；
- 5.在下发指纹时：只有当 NET\_DVR\_FINGERPRINT\_STATUS 中 byRecvStatus 字段为 0-成功，同时 byCardReaderRecvStatus 字段为 1-成功时，才代表指纹下发成功。否则需要根据 byRecvStatus 字段（门禁设备检测到的错误）和 byCardReaderRecvStatus 字段（门禁设备内读卡器检测到的错误）来判断具体的出错原因，并重新下发该枚指纹。

## 2.4 人脸管理

---

### 2.4.1 功能介绍

人脸获取：从门禁设备中获取某张卡所关联的人脸信息；

人脸下发：下发某张卡所关联的人脸信息到门禁设备；

人脸删除：删除门禁设备中的人脸信息；

人脸采集：使用门禁设备实时采集设备前的人脸图片到平台，该人脸图片可用于人脸下发。

### 2.4.2 功能示例

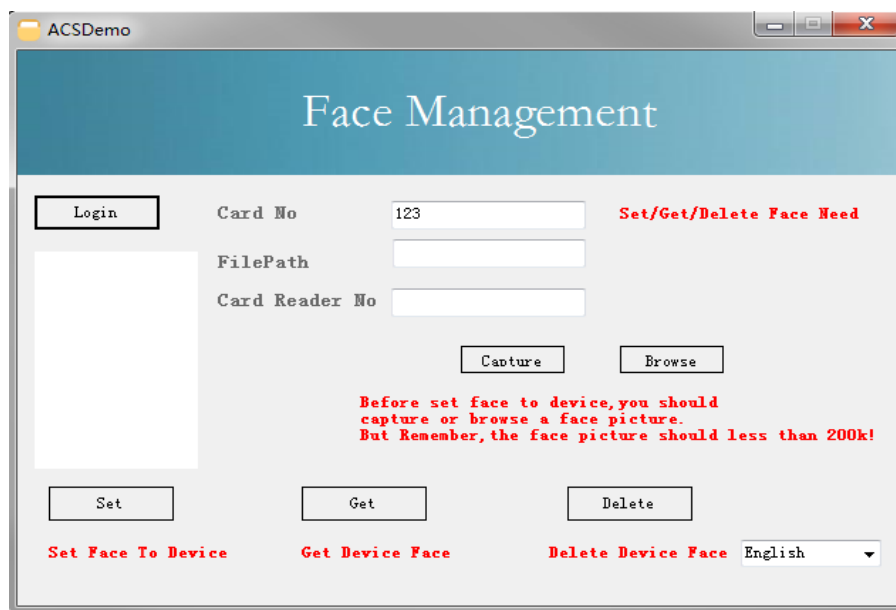
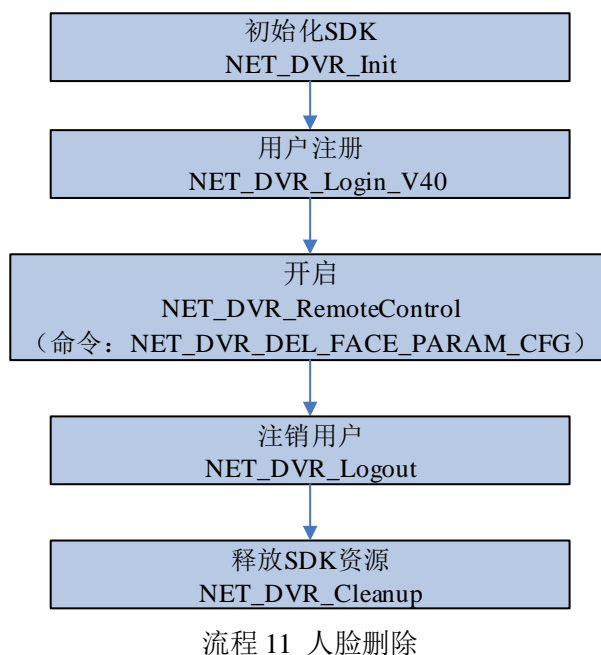
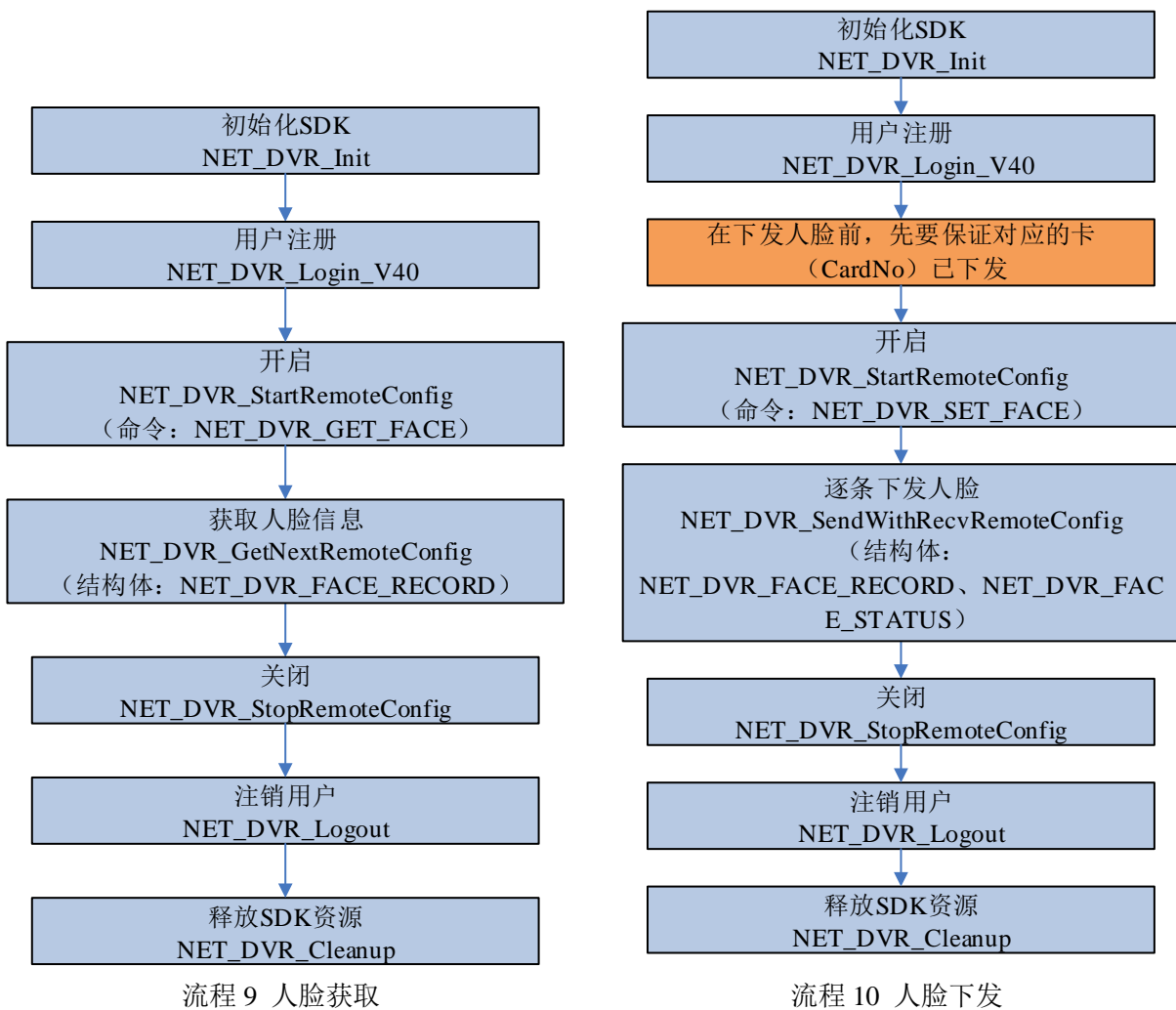
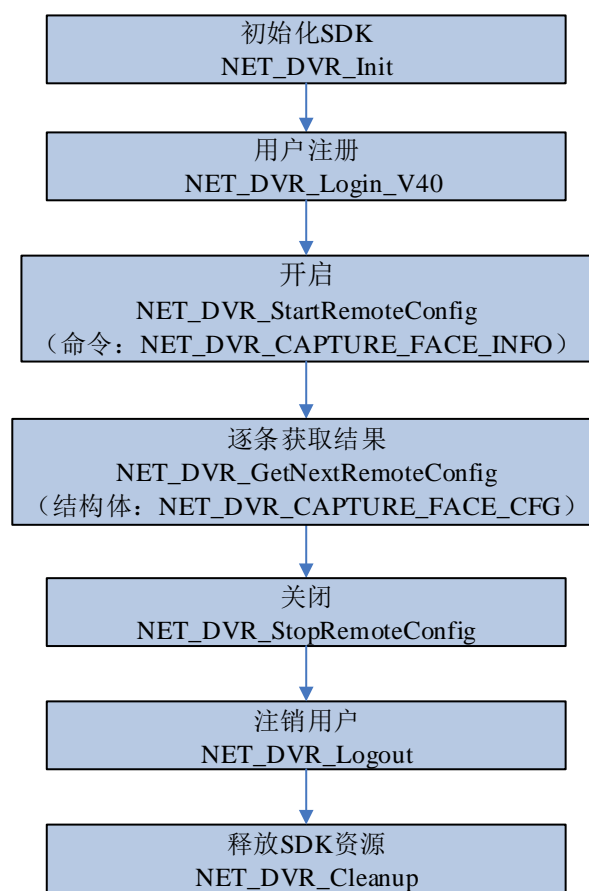


图 4 人脸管理示例

### 2.4.3 集成流程





流程 12 人脸采集

## 2.4.4 备注

- 1.由于卡管理中包含了人员相关信息，故：在下发人脸前，需要先下发卡；在删除卡前，需要先删除人脸；
- 2.人员和人脸之间通过卡号进行关联，并确定人员的唯一性；
- 3.在下发人脸时，每次调用人脸下发流程，可逐条下发多张人脸（多次调用NET\_DVR\_SendWithRecvRemoteConfig 接口），数目没有限制；
- 4.在获取人脸时，每次调用流程最多只能获取 1 张卡的 1 个人脸，如果想要再获取其他卡的人脸，则需要重新执行该流程（指的是从 NET\_DVR\_StartRemoteConfig 、NET\_DVR\_GetNextRemoteConfig 到 NET\_DVR\_StopRemoteConfig 的流程）；
- 5.人脸图片的大小最大不超过 200k；
- 6.在下发人脸图片时：只有当 NET\_DVR\_FACE\_STATUS 中 byRecvStatus 字段为 1-成功，才代表人脸图片下发成功。否则需要根据 byRecvStatus 字段来判断具体的出错原因，并重新下发该张人脸图片。

## 2.5 远程控门

### 2.5.1 功能介绍

用于控制门禁设备门的开、关、常开、常关。

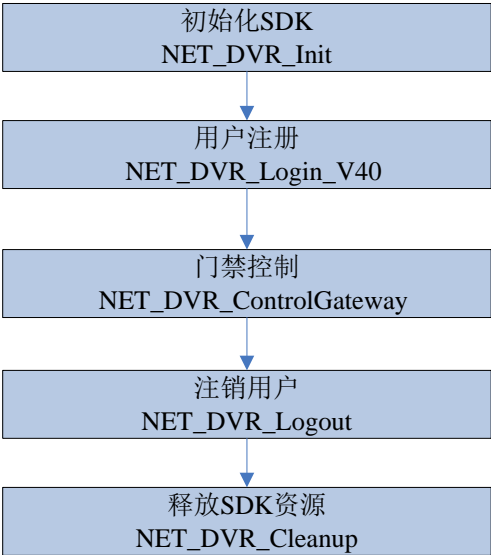
2.5.2 功能示例



图 5 远程控门示例

2.5.3 集成流程

远程控门对应的接口：NET\_DVR\_ControlGateway。接口的详细调用方式见附件。



流程 13 远程控门

2.5.4 备注

用户注册（NET\_DVR\_Login\_V40）成功之后即可通过门禁控制接口 NET\_DVR\_ControlGateway

来控制门禁的开启和关闭。

## 2.6 设备事件主动上传（布防）

### 2.6.1 功能介绍

用于实时获取门禁设备的人员刷卡、异常等信息。

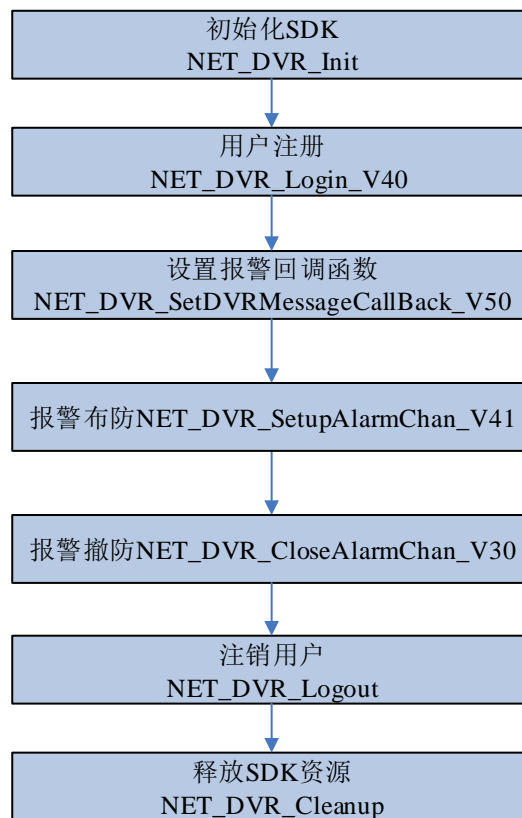
### 2.6.2 功能示例

No.	Time	AlarmInfo

图 6 事件上传示例

### 2.6.3 集成流程





流程 14 设备事件主动上传（布防）

## 2.6.4 备注

1. “布防”是指 SDK 主动连接设备，并发起报警上传命令，设备发生刷卡等事件时会立即上传给 SDK；

2. “布防”分为客户端布防和实时布防：客户端布防支持实时事件上传+离线事件上传；实时布防仅支持实时事件上传，不支持离线事件上传（**性能限制**：门禁设备只支持 1 路客户端布防，4 路实时布防；**调用区别**：通过 NET\_DVR\_SETUPALARM\_PARAM 中 byDeployType（布防类型：0-客户端布防，1-实时布防）字段进行区分）。

## 2.7 主动获取设备事件

### 2.7.1 功能介绍

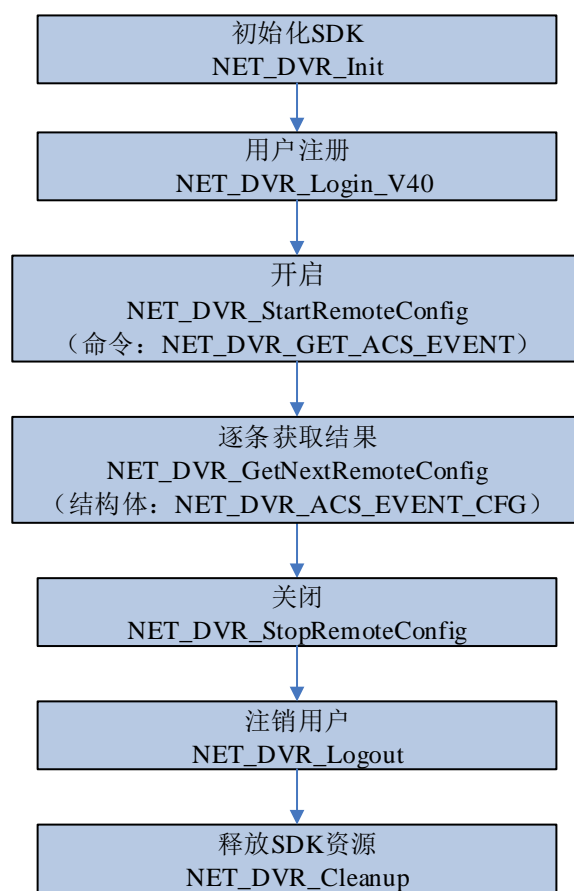
用于事后从门禁设备查询人员刷卡、异常等信息。

### 2.7.2 功能示例

[illegible]

图 7 主动获取设备事件示例

### 2.7.3 集成流程



流程 15 主动获取设备事件流程

#### 2.7.4 备注

设备事件获取对应的命令码：NET\_DVR\_GET\_ACS\_EVENT，该接口可用于主动获取设备事件信息。接口的详细调用方式见附件。

## 2.8 卡权限计划模板管理

### 2.8.1 功能介绍

用于配置不同人员的开关门时间段（按照周、假日配置时间段）。

### 2.8.2 功能示例

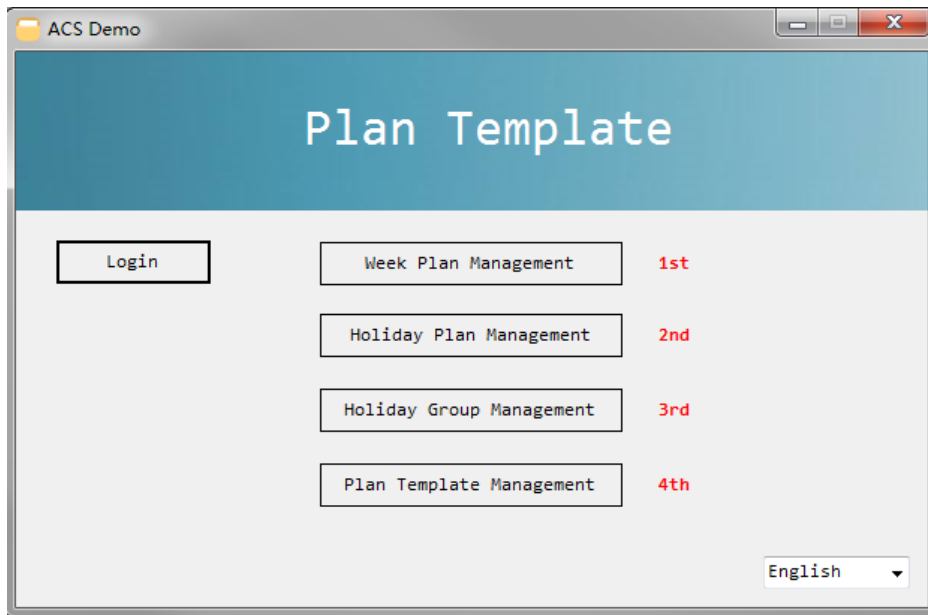
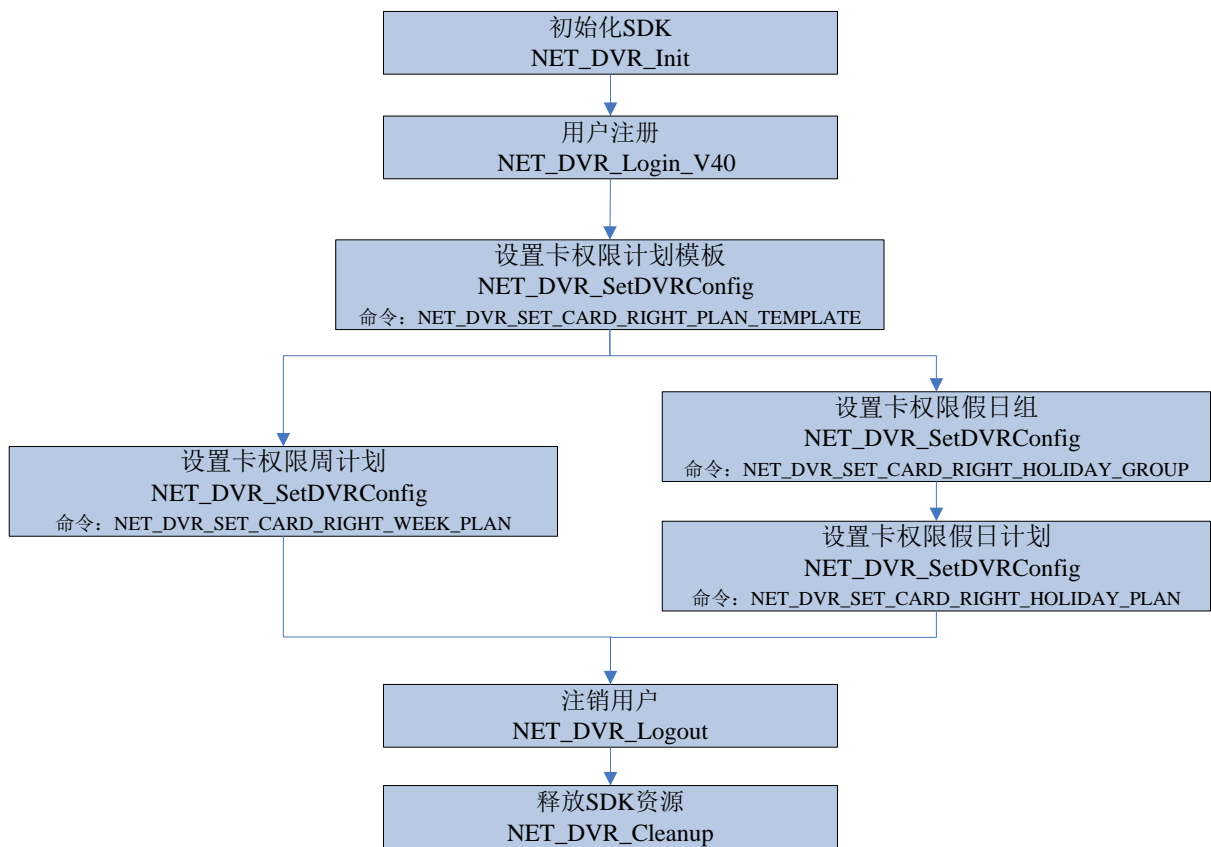


图 8 卡权限计划模板配置示例

### 2.8.3 集成流程



流程 16 卡权限计划模板配置流程

### 2.8.4 备注

---

1. 卡权限计划模板配置：设置卡权限计划模板（NET\_DVR\_SET\_CARD\_RIGHT\_PLAN\_TEMPLATE）<——设置卡权限周计划（NET\_DVR\_SET\_CARD\_RIGHT\_WEEK\_PLAN）、设置卡权限假日组（NET\_DVR\_SET\_CARD\_RIGHT\_HOLIDAY\_GROUP）<——设置卡权限假日计划（NET\_DVR\_SET\_CARD\_RIGHT\_HOLIDAY\_PLAN）。

2.这里只是配置好了卡权限计划模板，下发卡时关联配置的模板后，该卡就拥有了配置的权限。

## 3. 接口附录

### 3.1 卡管理

#### 3.1.1 卡获取（获取所有）

##### 3.1.1.1 接口函数

###### 1.开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG lUserID, DWORD dwCommand, LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL, LPVOID pUserData = NULL);
```

###### Parameters

lUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_CARD

lpInBuffer

[in] 指向一个 NET\_DVR\_CARD\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_CARD\_COND 结构体大小

###### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

###### 2.逐条获取结果

```
NET_DVR_API LONG __stdcall NET_DVR_GetNextRemoteConfig(LONG lHandle, void* lpOutBuff, DWORD dwOutBuffSize);
```

###### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpOutBuff

[out] 指向一个 NET\_DVR\_CARD\_RECORD 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_CARD\_RECORD 结构体大小

### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据，处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据，继续调用 NET_DVR_GetNextRemoteConfig
NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完，可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常，可调用 NET_DVR_StopRemoteConfig 结束长连接

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG IHandle);

### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.1.1.2 命令码

#define NET\_DVR\_GET\_CARD 2560 //获取卡

#### 3.1.1.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MAX_DOOR_NUM_256	门个数	256
MAX_GROUP_NUM_128	最大群组数	128
CARD_PASSWORD_LEN	卡密码长度	8
NAME_LEN	用户名长度	32

##### 结构体:

typedef struct \_NET\_DVR\_CARD\_COND

---

```

{
    DWORD dwSize;
    DWORD dwCardNum; //设置或获取卡数量，获取时置为 0xffffffff 表示获取所有卡信息
    BYTE  byRes[64];
}NET_DVR_CARD_COND, *LPNET_DVR_CARD_COND;

typedef struct _NET_DVR_CARD_RECORD
{
    DWORD                dwSize;
    BYTE                byCardNo[ACS_CARD_NO_LEN];
    BYTE                byCardType;
    BYTE                byLeaderCard;
    BYTE                byUserType;
    BYTE                byRes;
    BYTE                byDoorRight[MAX_DOOR_NUM_256];
    NET_DVR_VALID_PERIOD_CFG  struValid;
    BYTE                byBelongGroup[MAX_GROUP_NUM_128];
    BYTE                byCardPassword[CARD_PASSWORD_LEN];
    WORD                wCardRightPlan[MAX_DOOR_NUM_256];
    DWORD                dwMaxSwipeTimes;
    DWORD                dwSwipeTimes;
    DWORD                dwEmployeeNo;
    BYTE                byName[NAME_LEN];
    BYTE                byRes[260];
}NET_DVR_CARD_RECORD, *LPNET_DVR_CARD_RECORD;

```

### 3.1.2 卡获取（按条件获取）

#### 3.1.2.1 接口函数

##### 1.开启

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_StartRemoteConfig(LONG IUserID, DWORD dwCommand, LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL, LPVOID pUserData = NULL);

##### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_CARD

lpInBuffer

[in] 指向一个 NET\_DVR\_CARD\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_CARD\_COND 结构体大小

## Return Values

-1 表示失败，其他值作为 NET\_DVR\_SendWithRecvRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.逐条获取结果

```
NET_DVR_API LONG __stdcall NET_DVR_SendWithRecvRemoteConfig(LONG IHandle, void* lpInBuff, DWORD dwInBuffSize, void *lpOutBuff, DWORD dwOutBuffSize, DWORD *dwOutDataLen);
```

## Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpInBuff

[in] 指向一个 NET\_DVR\_CARD\_SEND\_DATA 结构体

dwInBuffSize

[in] 一个 NET\_DVR\_CARD\_SEND\_DATA 结构体大小

lpOutBuff

[out] 指向一个 NET\_DVR\_CARD\_RECORD 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_CARD\_RECORD 结构体大小

dwOutDataLen

[out] 实际收到的数据长度指针，不能为 NULL

## Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_CONFIG_STATUS_SUCCESS	1000	成功读取到数据，客户端处理完本次数据后需要再次调用 NET_DVR_SendWithRecvRemoteConfig 获取下一条数据
NET_SDK_CONFIG_STATUS_NEEDWAIT	1001	配置等待，客户端可重新 NET_DVR_SendWithRecvRemoteConfig
NET_SDK_CONFIG_STATUS_FINISH	1002	数据全部取完，此时客户端可调用 NET_DVR_StopRemoteConfig 结束
NET_SDK_CONFIG_STATUS_FAILED	1003	配置失败，客户端可重新 NET_DVR_SendWithRecvRemoteConfig 下发下一条
NET_SDK_CONFIG_STATUS_EXCEPTION	1004	配置异常，此时客户端可调用 NET_DVR_StopRemoteConfig 结束

## 3.关闭

```
NET_DVR_API BOOL __stdcall NET_DVR_StopRemoteConfig(LONG IHandle);
```



## Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

## Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 3.1.2.2 命令码

```
#define NET_DVR_GET_CARD    2560 //获取卡
```

### 3.1.2.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MAX_DOOR_NUM_256	门个数	256
MAX_GROUP_NUM_128	最大群组数	128
CARD_PASSWORD_LEN	卡密码长度	8
NAME_LEN	用户名长度	32

#### 结构体:

```
typedef struct _NET_DVR_CARD_COND
{
    DWORD dwSize;
    DWORD dwCardNum; //设置或获取卡数量，获取时置为 0xffffffff 表示获取所有卡信息
    BYTE  byRes[64];
}NET_DVR_CARD_COND, *LPNET_DVR_CARD_COND;
```

```
typedef struct _NET_DVR_CARD_SEND_DATA
{
    DWORD dwSize;
    BYTE  byCardNo[ACS_CARD_NO_LEN]; //卡号
    BYTE  byRes[16];
}NET_DVR_CARD_SEND_DATA, *LPNET_DVR_CARD_SEND_DATA;
```

```
typedef struct _NET_DVR_CARD_RECORD
{
    DWORD dwSize;
    BYTE  byCardNo[ACS_CARD_NO_LEN];
    BYTE  byCardType;
    BYTE  byLeaderCard;
    BYTE  byUserType;
    BYTE  byRes;
```

---

```

    BYTE                byDoorRight[MAX_DOOR_NUM_256];
    NET_DVR_VALID_PERIOD_CFG    struValid;
    BYTE                byBelongGroup[MAX_GROUP_NUM_128];
    BYTE                byCardPassword[CARD_PASSWORD_LEN];
    WORD                wCardRightPlan[MAX_DOOR_NUM_256];
    DWORD               dwMaxSwipeTimes;
    DWORD               dwSwipeTimes;
    DWORD               dwEmployeeNo;
    BYTE                byName[NAME_LEN];
    BYTE                byRes[260];
}NET_DVR_CARD_RECORD, *LPNET_DVR_CARD_RECORD;

```

### 3.1.3 卡下发

#### 3.1.3.1 接口函数

##### 1. 开启

```

NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG lUserID, DWORD dwCommand,
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,
LPVOID pUserData = NULL);

```

##### Parameters

lUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_SET\_CARD

lpInBuffer

[in] 指向一个 NET\_DVR\_CARD\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_CARD\_COND 结构体大小

##### Return Values

-1 表示失败，其他值作为 NET\_DVR\_SendWithRecvRemoteConfig 等接口的句柄。接口返回失败  
 请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

##### 2. 逐条获取结果

```

NET_DVR_API LONG __stdcall NET_DVR_SendWithRecvRemoteConfig(LONG lHandle, void*
lpInBuff, DWORD dwInBuffSize, void *lpOutBuff, DWORD dwOutBuffSize, DWORD
*dwOutDataLen);

```

##### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpInBuff

[in] 指向一个 NET\_DVR\_CARD\_RECORD 结构体

dwInBuffSize  
[in] 一个 NET\_DVR\_CARD\_RECORD 结构体大小

lpOutBuff  
[out] 指向一个 NET\_DVR\_CARD\_STATUS 结构体

dwOutBuffSize  
[in] 一个 NET\_DVR\_CARD\_STATUS 结构体大小

dwOutDataLen  
[out] 实际收到的数据长度指针，不能为 NULL

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_CONFIG_STATUS_SUCCESS	1000	成功读取到数据，客户端处理完本次数据后需要再次调用 NET_DVR_SendWithRecvRemoteConfig 获取下一条数据
NET_SDK_CONFIG_STATUS_NEEDWAIT	1001	配置等待，客户端可重新 NET_DVR_SendWithRecvRemoteConfig
NET_SDK_CONFIG_STATUS_FINISH	1002	数据全部取完，此时客户端可调用 NET_DVR_StopRemoteConfig 结束
NET_SDK_CONFIG_STATUS_FAILED	1003	配置失败，客户端可重新 NET_DVR_SendWithRecvRemoteConfig 下发下一条
NET_SDK_CONFIG_STATUS_EXCEPTION	1004	配置异常，此时客户端可调用 NET_DVR_StopRemoteConfig 结束

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

lHandle  
[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.1.3.2 命令码

#define NET\_DVR\_SET\_CARD 2561 //下发卡

#### 3.1.3.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值
-----	-------	------

---

ACS_CARD_NO_LEN	卡号长度	32
MAX_DOOR_NUM_256	门个数	256
MAX_GROUP_NUM_128	最大群组数	128
CARD_PASSWORD_LEN	卡密码长度	8
NAME_LEN	用户名长度	32

🌈 结构体:

```
typedef struct _NET_DVR_CARD_COND
{
    DWORD dwSize;
    DWORD dwCardNum; //设置或获取卡数量，获取时置为 0xffffffff 表示获取所有卡信息
    BYTE  byRes[64];
}NET_DVR_CARD_COND, *LPNET_DVR_CARD_COND;
```

```
typedef struct _NET_DVR_CARD_RECORD
{
    DWORD dwSize;
    BYTE  byCardNo[ACS_CARD_NO_LEN];
    BYTE  byCardType;
    BYTE  byLeaderCard;
    BYTE  byUserType;
    BYTE  byRes;
    BYTE  byDoorRight[MAX_DOOR_NUM_256];
    NET_DVR_VALID_PERIOD_CFG struValid;
    BYTE  byBelongGroup[MAX_GROUP_NUM_128];
    BYTE  byCardPassword[CARD_PASSWORD_LEN];
    WORD  wCardRightPlan[MAX_DOOR_NUM_256];
    DWORD dwMaxSwipeTimes;
    DWORD dwSwipeTimes;
    DWORD dwEmployeeNo;
    BYTE  byName[NAME_LEN];
    BYTE  byRes[260];
}NET_DVR_CARD_RECORD, *LPNET_DVR_CARD_RECORD;
```

```
typedef struct _NET_DVR_CARD_STATUS
{
    DWORD dwSize;
    BYTE  byCardNo[ACS_CARD_NO_LEN];
    DWORD dwErrorCode;
    BYTE  byStatus; // 状态: 0-失败, 1-成功
    BYTE  byRes[23];
}NET_DVR_CARD_STATUS, *LPNET_DVR_CARD_STATUS;
```

#### 3.1.3.4 备注

---

dwErrorCode 相关错误码:

#define NET_ERR_ILLEGAL_CARDNO	1904 //卡号错误（卡号不符合门禁设备要求，请检测卡号是否出错）
#define NET_ERR_DEVICE_CARD_FULL	1908 //卡已满（门禁设备存储的卡号已达上限，不能再下发其他卡号了）
#define NET_ERR_DOOR_OVER_LIMIT	1917 //门超出设备最大能力（门编号超出门禁设备实际支持的数目）
#define NET_ERR_NOT_SUPPORT_ONE_MORE_CARD	1920 //不支持一人多卡（不同卡号下发的工号重复）

### 3.1.4 卡删除

#### 3.1.4.1 接口函数

##### 1. 开启

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_StartRemoteConfig(LONG IUserID, DWORD dwCommand, LPVOID lpInBuffer, DWORD dwInBufferSize, fRemoteConfigCallback cbStateCallback = NULL, LPVOID pUserData = NULL);

##### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_DEL\_CARD

lpInBuffer

[in] 指向一个 NET\_DVR\_CARD\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_CARD\_COND 结构体大小

##### Return Values

-1 表示失败，其他值作为 NET\_DVR\_SendWithRecvRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

##### 2. 逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_SendWithRecvRemoteConfig(LONG IHandle, void\* lpInBuff, DWORD dwInBuffSize, void \*lpOutBuff, DWORD dwOutBuffSize, DWORD \*dwOutDataLen);

##### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpInBuff

[in] 指向一个 NET\_DVR\_CARD\_SEND\_DATA 结构体

dwInBuffSize

[in] 一个 NET\_DVR\_CARD\_SEND\_DATA 结构体大小

lpOutBuff

[out] 指向一个 NET\_DVR\_CARD\_STATUS 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_CARD\_STATUS 结构体大小

dwOutDataLen

[out] 实际收到的数据长度指针，不能为 NULL

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_CONFIG_STATUS_SUCCESS	1000	成功读取到数据，客户端处理完本次数据后需要再次调用 NET_DVR_SendWithRecvRemoteConfig 获取下一条数据
NET_SDK_CONFIG_STATUS_NEEDWAIT	1001	配置等待，客户端可重新 NET_DVR_SendWithRecvRemoteConfig
NET_SDK_CONFIG_STATUS_FINISH	1002	数据全部取完，此时客户端可调用 NET_DVR_StopRemoteConfig 结束
NET_SDK_CONFIG_STATUS_FAILED	1003	配置失败，客户端可重新 NET_DVR_SendWithRecvRemoteConfig 下发下一条
NET_SDK_CONFIG_STATUS_EXCEPTION	1004	配置异常，此时客户端可调用 NET_DVR_StopRemoteConfig 结束

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.1.4.2 命令码

#define NET\_DVR\_DEL\_CARD 2562 //删除卡

#### 3.1.4.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32

---

🌈 结构体:

```
typedef struct _NET_DVR_CARD_COND
{
    DWORD dwSize;
    DWORD dwCardNum; //设置或获取卡数量，获取时置为 0xffffffff 表示获取所有卡信息
    BYTE  byRes[64];
}NET_DVR_CARD_COND, *LPNET_DVR_CARD_COND;

typedef struct _NET_DVR_CARD_SEND_DATA
{
    DWORD dwSize;
    BYTE  byCardNo[ACS_CARD_NO_LEN]; //卡号
    BYTE  byRes[16];
}NET_DVR_CARD_SEND_DATA, *LPNET_DVR_CARD_SEND_DATA;

typedef struct _NET_DVR_CARD_STATUS
{
    DWORD  dwSize;
    BYTE   byCardNo[ACS_CARD_NO_LEN];
    DWORD  dwErrorCode;
    BYTE   byStatus; // 状态: 0-失败, 1-成功
    BYTE   byRes[23];
}NET_DVR_CARD_STATUS, *LPNET_DVR_CARD_STATUS;
```

## 3.2 指纹管理

### 3.2.1 指纹获取

#### 3.2.1.1 接口函数

##### 1.开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand,
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,
LPVOID pUserData = NULL);
```

##### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_FINGERPRINT

lpInBuffer

[in] 指向一个 NET\_DVR\_FINGERPRINT\_COND 结构体

dwInBufferSize:

---

[in] 一个 NET\_DVR\_FINGERPRINT\_COND 结构体大小

#### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 2.逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_GetNextRemoteConfig(LONG IHandle, void\* lpOutBuff, DWORD dwOutBuffSize);

#### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpOutBuff

[out] 指向一个 NET\_DVR\_FINGERPRINT\_RECORD 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_FINGERPRINT\_RECORD 结构体大小

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据,处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据,继续调用 NET_DVR_GetNextRemoteConfig
NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完,可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常,可调用 NET_DVR_StopRemoteConfig 结束长连接

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG IHandle);

#### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.2.1.2 命令码



---

```
#define NET_DVR_GET_FINGERPRINT    2563 //获取指纹
```

### 3.2.1.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
NET_SDK_EMPLOYEE_NO_LEN	工号长度	32
MAX_CARD_READER_NUM_512	最大读卡器数	512
MAX_FINGER_PRINT_LEN	最大指纹长度	768

#### 结构体:

```
typedef struct tagNET_DVR_FINGERPRINT_COND
```

```
{
```

```
    DWORD    dwSize;
```

```
    DWORD    dwFingerprintNum;    //指纹数量（指纹获取或下发时有效，指纹获取时为 0xffffffff  
表示获取所有指纹信息）
```

```
    BYTE     byCardNo[ACS_CARD_NO_LEN];    //指纹关联的卡号（指纹获取时有效）
```

```
    DWORD    dwEnableReaderNo;    //指纹读卡器编号（指纹获取时有效）
```

```
    BYTE     byFingerPrintID;    //指纹编号，有效值范围 1-10（指纹获取时有效，0xff 表示该卡  
所有指纹）
```

```
    BYTE     byRes[131];
```

```
}NET_DVR_FINGERPRINT_COND, *LPNET_DVR_FINGERPRINT_COND;
```

```
typedef struct _NET_DVR_FINGERPRINT_RECORD
```

```
{
```

```
    DWORD    dwSize;
```

```
    BYTE     byCardNo[ACS_CARD_NO_LEN]; //指纹关联的卡号
```

```
    DWORD    dwFingerPrintLen;    //指纹数据长度
```

```
    DWORD    dwEnableReaderNo;    //需要下发指纹的读卡器编号
```

```
    BYTE     byFingerPrintID;    //手指编号，有效值范围为 1-10
```

```
    BYTE     byFingerType;    //指纹类型 0-普通指纹，1-胁迫指纹
```

```
    BYTE     byRes1[30];
```

```
    BYTE     byFingerData[MAX_FINGER_PRINT_LEN];    //指纹数据内容
```

```
    BYTE     byRes[96];
```

```
}NET_DVR_FINGERPRINT_RECORD, *LPNET_DVR_FINGERPRINT_RECORD;
```

## 3.2.2 指纹下发

### 3.2.2.1 接口函数

#### 1.开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand,  
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,
```

---

LPVOID pUserData = NULL);

### Parameters

lUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_SET\_FINGERPRINT

lpInBuffer

[in] 指向一个 NET\_DVR\_FINGERPRINT\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_FINGERPRINT\_COND 结构体大小

### Return Values

-1 表示失败，其他值作为 NET\_DVR\_SendWithRecvRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_SendWithRecvRemoteConfig(LONG lHandle, void\* lpInBuff, DWORD dwInBuffSize, void \*lpOutBuff, DWORD dwOutBuffSize, DWORD \*dwOutDataLen);

### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpInBuff

[in] 指向一个 NET\_DVR\_FINGERPRINT\_RECORD 结构体

dwInBuffSize

[in] 一个 NET\_DVR\_FINGERPRINT\_RECORD 结构体大小

lpOutBuff

[out] 指向一个 NET\_DVR\_FINGERPRINT\_STATUS 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_FINGERPRINT\_STATUS 结构体大小

dwOutDataLen

[out] 实际收到的数据长度指针，不能为 NULL

### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_CONFIG_STATUS_SUCCESS	1000	成功读取到数据，客户端处理完本次数据后需要再次调用 NET_DVR_SendWithRecvRemoteConfig 获取下一条数据
NET_SDK_CONFIG_STATUS_NEEDWAIT	1001	配置等待，客户端可重新 NET_DVR_SendWithRecvRemoteConfig

NET_SDK_CONFIG_STATUS_FINISH	1002	数据全部取完，此时客户端可调用 NET_DVR_StopRemoteConfig 结束
NET_SDK_CONFIG_STATUS_FAILED	1003	配置失败，客户端可重新 NET_DVR_SendWithRecvRemoteConfig 下发下一条
NET_SDK_CONFIG_STATUS_EXCEPTION	1004	配置异常，此时客户端可调用 NET_DVR_StopRemoteConfig 结束

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.2.2.2 命令码

#define NET\_DVR\_SET\_FINGERPRINT 2564 //下发指纹

#### 3.2.2.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
NET_SDK_EMPLOYEE_NO_LEN	工号长度	32
MAX_CARD_READER_NUM_512	最大读卡器数	512
MAX_FINGER_PRINT_LEN	最大指纹长度	768
ERROR_MSG_LEN	下发错误信息	32

##### 结构体：

typedef struct tagNET\_DVR\_FINGERPRINT\_COND

{

    DWORD    dwSize;

    DWORD    dwFingerprintNum;    //指纹数量（指纹获取或下发时有效，指纹获取时为 0xffffffff

表示获取所有指纹信息）

    BYTE    byCardNo[ACS\_CARD\_NO\_LEN];    //指纹关联的卡号（指纹获取时有效）

    DWORD    dwEnableReaderNo;    //指纹读卡器编号（指纹获取时有效）

    BYTE    byFingerPrintID;    //指纹编号，有效值范围 1-10（指纹获取时有效，0xff 表示该卡所有指纹）

    BYTE    byRes[131];

}NET\_DVR\_FINGERPRINT\_COND, \*LPNET\_DVR\_FINGERPRINT\_COND;

---

```
typedef struct _NET_DVR_FINGERPRINT_RECORD
{
    DWORD    dwSize;
    BYTE    byCardNo[ACS_CARD_NO_LEN]; //指纹关联的卡号
    DWORD dwFingerPrintLen;    //指纹数据长度
    DWORD    dwEnableReaderNo;    //需要下发指纹的读卡器编号
    BYTE    byFingerPrintID;    //手指编号，有效值范围为 1-10
    BYTE    byFingerType;    //指纹类型 0-普通指纹，1-胁迫指纹
    BYTE    byRes1[30];
    BYTE    byFingerData[MAX_FINGER_PRINT_LEN];    //指纹数据内容
    BYTE    byRes[96];
}NET_DVR_FINGERPRINT_RECORD, *LPNET_DVR_FINGERPRINT_RECORD;
```

```
typedef struct _NET_DVR_FINGERPRINT_STATUS
{
    DWORD dwSize;
    BYTE    byCardNo[ACS_CARD_NO_LEN]; //指纹关联的卡号
    BYTE    byCardReaderRecvStatus; //指纹读卡器状态，按字节表示，0-失败，1-成功，2-该指纹模组不在线，3-重试或指纹质量差，4-内存已满，5-已存在该指纹，6-已存在该指纹 ID，7-非法指纹 ID，8-该指纹模组无需配置
    BYTE    byFingerPrintID;    //手指编号，有效值范围为 1-10
    BYTE    byFingerType;    //指纹类型 0-普通指纹，1-胁迫指纹
    BYTE    byRecvStatus;    //主机错误状态：0-成功，1-手指编号错误，2-指纹类型错误，3-卡号错误（卡号规格不符合设备要求），4-指纹未关联工号或卡号（工号或卡号字段为空），5-工号不存在，6-指纹数据长度为 0，7-读卡器编号错误，8-工号错误
    BYTE    byErrorMsg[ERROR_MSG_LEN]; //下发错误信息，当 byCardReaderRecvStatus 为 5 时，表示已存在指纹对应的卡号
    DWORD dwCardReaderNo;    //当 byCardReaderRecvStatus 为 5 时，表示已存在指纹对应的指纹读卡器编号，可用于下发错误返回。0 时表示无错误信息
    BYTE    byRes[20];
}NET_DVR_FINGERPRINT_STATUS, *LPNET_DVR_FINGERPRINT_STATUS;
```

### 3.2.3 指纹删除

#### 3.2.3.1 接口函数

##### 1. 开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand, LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL, LPVOID pUserData = NULL);
```

##### Parameters

IUserID

---

[in] NET\_DVR\_Login\_V40 等登录接口的返回值  
dwCommand  
[in] NET\_DVR\_DEL\_FINGERPRINT  
lpInBuffer  
[in] 指向一个 NET\_DVR\_FINGER\_PRINT\_INFO\_CTRL\_V50 结构体  
dwInBufferSize:  
[in] 一个 NET\_DVR\_FINGER\_PRINT\_INFO\_CTRL\_V50 结构体大小

#### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_GetNextRemoteConfig(LONG lHandle, void\* lpOutBuff, DWORD dwOutBuffSize);

#### Parameters

lHandle  
[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值  
lpOutBuff  
[out] 指向一个 NET\_DVR\_FINGER\_PRINT\_INFO\_STATUS\_V50 结构体  
dwOutBuffSize  
[in] 一个 NET\_DVR\_FINGER\_PRINT\_INFO\_STATUS\_V50 结构体大小

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据,处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据,继续调用 NET_DVR_GetNextRemoteConfig
NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完,可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常,可调用 NET_DVR_StopRemoteConfig 结束长连接

## 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 3.2.3.2 命令码

```
#define NET_DVR_DEL_FINGERPRINT    2565 //删除指纹
```

### 3.2.3.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MAX_CARD_READER_NUM	最大读卡器数	512
MAX_FINGER_PRINT_NUM	最大指纹个数	10
NET_SDK_EMPLOYEE_NO_LEN	工号长度	32

#### 结构体：

```
typedef struct tagNET_DVR_FINGER_PRINT_INFO_CTRL_V50
```

```
{  
    DWORD dwSize;  
    BYTE  byMode;           //删除方式，0-按卡号（人员 ID）方式删除，1-按读卡器删除  
    BYTE  byRes1[3];        //保留  
    NET_DVR_DEL_FINGER_PRINT_MODE_V50 struProcessMode; //处理方式  
    BYTE  byRes[64];        //保留  
}NET_DVR_FINGER_PRINT_INFO_CTRL_V50,  
*LPNET_DVR_FINGER_PRINT_INFO_CTRL_V50;
```

```
typedef union tagNET_DVR_DEL_FINGER_PRINT_MODE_V50
```

```
{  
    BYTE  uLen[588];        //联合体长度  
    NET_DVR_FINGER_PRINT_BYCARD_V50      struByCard;    //按卡号（人员 ID）的方式  
删除  
    NET_DVR_FINGER_PRINT_BYREADER_V50    struByReader;  //按读卡器的方式删除  
}NET_DVR_DEL_FINGER_PRINT_MODE_V50,  
*LPNET_DVR_DEL_FINGER_PRINT_MODE_V50;
```

```
typedef struct tagNET_DVR_FINGER_PRINT_BYCARD_V50
```

```
{  
    BYTE  byCardNo[ACS_CARD_NO_LEN]; //指纹关联的卡号  
    BYTE  byEnableCardReader[MAX_CARD_READER_NUM_512]; //指纹的读卡器信息，按位表示  
}
```

---

```
    BYTE  byFingerPrintID[MAX_FINGER_PRINT_NUM/*10*/];           //需要删除的手指编号，按
数组下标，值表示 0-不删除，1-删除该指纹
```

```
    BYTE  byRes1[2];
```

```
    BYTE  byEmployeeNo[NET_SDK_EMPLOYEE_NO_LEN]; //工号（人员 ID）
```

```
}NET_DVR_FINGER_PRINT_BYCARD_V50, *LPNET_DVR_FINGER_PRINT_BYCARD_V50;
```

```
typedef struct tagNET_DVR_FINGER_PRINT_BYREADER_V50
```

```
{
```

```
    DWORD dwCardReaderNo; //按值表示，指纹读卡器编号
```

```
    BYTE  byClearAllCard; //是否删除所有卡的指纹信息，0-按卡号（人员 ID）删除指纹信息，
1-删除所有卡（人员 ID）的指纹信息
```

```
    BYTE  byRes1[3];           //保留
```

```
    BYTE  byCardNo[ACS_CARD_NO_LEN]; //指纹关联的卡号
```

```
    BYTE  byEmployeeNo[NET_SDK_EMPLOYEE_NO_LEN]; //工号（人员 ID）
```

```
    BYTE  byRes[516];           //保留
```

```
}NET_DVR_FINGER_PRINT_BYREADER_V50,
```

```
*LPNET_DVR_FINGER_PRINT_BYREADER_V50;
```

```
typedef struct tagNET_DVR_FINGER_PRINT_INFO_STATUS_V50
```

```
{
```

```
    DWORD dwSize;
```

```
    DWORD dwCardReaderNo; //按值表示，指纹读卡器编号
```

```
    BYTE  byStatus;           //状态：0-无效，1-处理中，2-删除失败，3-成功
```

```
    BYTE  byRes[63];           //保留
```

```
}NET_DVR_FINGER_PRINT_INFO_STATUS_V50,
```

```
*LPNET_DVR_FINGER_PRINT_INFO_STATUS_V50;
```

## 3.2.4 指纹采集

### 3.2.4.1 接口函数

#### 1. 开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand,
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,
LPVOID pUserData = NULL);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_CAPTURE\_FINGERPRINT\_INFO

lpInBuffer

[in] 指向一个 NET\_DVR\_CAPTURE\_FINGERPRINT\_COND 结构体

dwInBufferSize:

---

[in] 一个 NET\_DVR\_CAPTURE\_FINGERPRINT\_COND 结构体大小

#### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 2.逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_GetNextRemoteConfig(LONG IHandle, void\* lpOutBuff, DWORD dwOutBuffSize);

#### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpOutBuff

[out] 指向一个 NET\_DVR\_CAPTURE\_FINGERPRINT\_CFG 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_CAPTURE\_FINGERPRINT\_CFG 结构体大小

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据,处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据,继续调用 NET_DVR_GetNextRemoteConfig
NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完,可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常,可调用 NET_DVR_StopRemoteConfig 结束长连接

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG IHandle);

#### Parameters

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.2.4.2 命令码



#define NET\_DVR\_CAPTURE\_FINGERPRINT\_INFO 2504 //采集指纹

### 3.2.4.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
MAX_FINGER_PRINT_LEN	最大指纹长度	768

#### 结构体:

```
typedef struct tagNET_DVR_CAPTURE_FINGERPRINT_COND
{
```

```
    DWORD dwSize;
    BYTE byFingerPrintPicType; //图片类型: 0-无意义
    BYTE byFingerNo; //手指编号, 范围 1-10
    BYTE byRes[126];
```

```
}NET_DVR_CAPTURE_FINGERPRINT_COND, *LPNET_DVR_CAPTURE_FINGERPRINT_COND;
```

```
typedef struct tagNET_DVR_CAPTURE_FINGERPRINT_CFG
{
```

```
    DWORD dwSize;
    DWORD dwFingerPrintDataSize; //指纹数据大小
    BYTE byFingerData[MAX_FINGER_PRINT_LEN]; //指纹数据内容
    DWORD dwFingerPrintPicSize; //指纹图片大小, 等于 0 时, 代表无指纹图片数据
    char* pFingerPrintPicBuffer; //指纹图片缓存
    BYTE byFingerNo; //手指编号, 范围 1-10
    BYTE byFingerPrintQuality; //指纹质量, 范围 1-100
    BYTE byRes[62];
```

```
}NET_DVR_CAPTURE_FINGERPRINT_CFG, *LPNET_DVR_CAPTURE_FINGERPRINT_CFG;
```

## 3.3 人脸管理

### 3.3.1 人脸获取

#### 3.3.1.1 接口函数

##### 1.开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand,
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,
LPVOID pUserData = NULL);
```

##### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

---

dwCommand

[in] NET\_DVR\_GET\_FACE

lpInBuffer

[in] 指向一个 NET\_DVR\_FACE\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_FACE\_COND 结构体大小

#### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_GetNextRemoteConfig(LONG lHandle, void\* lpOutBuff, DWORD dwOutBuffSize);

#### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpOutBuff

[out] 指向一个 NET\_DVR\_FACE\_RECORD 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_FACE\_RECORD 结构体大小

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据,处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据,继续调用 NET_DVR_GetNextRemoteConfig
NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完,可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常,可调用 NET_DVR_StopRemoteConfig 结束长连接

## 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 3.3.1.2 命令码

```
#define NET_DVR_GET_FACE    2566 //获取人脸
```

### 3.3.1.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MAX_CARD_READER_NUM_512	最大读卡器数	512

#### 结构体：

```
typedef struct _NET_DVR_FACE_COND
```

```
{  
    DWORD dwSize;  
    BYTE  byCardNo[ACS_CARD_NO_LEN];    //人脸关联的卡号（设置时该参数可不设置）  
    DWORD dwFaceNum;    // 设置或获取人脸数量，获取时置为 0xffffffff 表示获取所有人脸信息  
    DWORD dwEnableReaderNo;    // 人脸读卡器编号  
    BYTE  byRes[124];    // 保留  
}NET_DVR_FACE_COND, *LPNET_DVR_FACE_COND;
```

```
typedef struct _NET_DVR_FACE_RECORD
```

```
{  
    DWORD dwSize;  
    BYTE  byCardNo[ACS_CARD_NO_LEN];    //人脸关联的卡号  
    DWORD dwFaceLen;    //人脸数据长度  
    BYTE* pFaceBuffer;    //人脸数据指针  
    BYTE  byRes[128];  
}NET_DVR_FACE_RECORD, *LPNET_DVR_FACE_RECORD;
```

## 3.3.2 人脸下发

### 3.3.2.1 接口函数

#### 1.开启

```
NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand,  
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,  
LPVOID pUserData = NULL);
```

## Parameters

lUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_SET\_FACE

lpInBuffer

[in] 指向一个 NET\_DVR\_FACE\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_FACE\_COND 结构体大小

## Return Values

-1 表示失败，其他值作为 NET\_DVR\_SendWithRecvRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.逐条获取结果

```
NET_DVR_API LONG __stdcall NET_DVR_SendWithRecvRemoteConfig(LONG lHandle, void* lpInBuff, DWORD dwInBuffSize, void *lpOutBuff, DWORD dwOutBuffSize, DWORD *dwOutDataLen);
```

## Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpInBuff

[in] 指向一个 NET\_DVR\_FACE\_RECORD 结构体

dwInBuffSize

[in] 一个 NET\_DVR\_FACE\_RECORD 结构体大小

lpOutBuff

[out] 指向一个 NET\_DVR\_FACE\_STATUS 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_FACE\_STATUS 结构体大小

dwOutDataLen

[out] 实际收到的数据长度指针，不能为 NULL

## Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_CONFIG_STATUS_SUCCESS	1000	成功读取到数据，客户端处理完本次数据后需要再次调用 NET_DVR_SendWithRecvRemoteConfig 获取下一条数据
NET_SDK_CONFIG_STATUS_NEEDWAIT	1001	配置等待，客户端可重新 NET_DVR_SendWithRecvRemoteConfig
NET_SDK_CONFIG_STATUS_FINISH	1002	数据全部取完，此时客户端可调用 NET_DVR_StopRemoteConfig 结束

NET_SDK_CONFIG_STATUS_FAILED	1003	配置失败，客户端可重新NET_DVR_SendWithRecvRemoteConfig下发下一条
NET_SDK_CONFIG_STATUS_EXCEPTION	1004	配置异常，此时客户端可调用NET_DVR_StopRemoteConfig结束

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.3.2.2 命令码

```
#define NET_DVR_SET_FACE    2567 //下发人脸
```

#### 3.3.2.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MAX_CARD_READER_NUM_512	最大读卡器数	512
ERROR_MSG_LEN	下发错误信息	32

##### 结构体:

```
typedef struct _NET_DVR_FACE_COND
```

```
{
    DWORD dwSize;
    BYTE byCardNo[ACS_CARD_NO_LEN];    //人脸关联的卡号（设置时该参数可不设置）
    DWORD dwFaceNum;    // 设置或获取人脸数量，获取时置为 0xffffffff 表示获取所有人脸信息
    DWORD dwEnableReaderNo;    // 人脸读卡器编号
    BYTE byRes[124];    // 保留
}NET_DVR_FACE_COND, *LPNET_DVR_FACE_COND;
```

```
typedef struct _NET_DVR_FACE_RECORD
```

```
{
    DWORD dwSize;
    BYTE byCardNo[ACS_CARD_NO_LEN];    //人脸关联的卡号
    DWORD dwFaceLen;    //人脸数据长度
    BYTE* pFaceBuffer;    //人脸数据指针
}
```

---

```

    BYTE  byRes[128];
}NET_DVR_FACE_RECORD, *LPNET_DVR_FACE_RECORD;

typedef struct _NET_DVR_FACE_STATUS
{
    DWORD dwSize;
    BYTE  byCardNo[ACS_CARD_NO_LEN]; //人脸关联的卡号
    BYTE  byErrorMsg[ERROR_MSG_LEN]; //下发错误信息，当 byCardReaderRecvStatus 为 4 时，
    表示已存在人脸对应的卡号
    DWORD dwReaderNo; //人脸读卡器编号，可用于下发错误返回
    BYTE  byRecvStatus; //人脸读卡器状态，按字节表示，0-失败，1-成功，2-重试或人脸质量差，
    3-内存已满(人脸数据满)，4-已存在该人脸，5-非法人脸 ID，6-算法建模失败，7-未下发卡权限，8-
    未定义（保留），9-人眼间距小距小，10-图片数据长度小于 1KB，11-图片格式不符（png/jpg/bmp），
    12-图片像素数量超过上限，13-图片像素数量低于下限，14-图片信息校验失败，15-图片解码失败，
    16-人脸检测失败，17-人脸评分失败
    BYTE  byRes[131];
}NET_DVR_FACE_STATUS, *LPNET_DVR_FACE_STATUS;

```

### 3.3.3 人脸删除

#### 3.3.3.1 接口函数

```

NET_DVR_API BOOL __stdcall NET_DVR_RemoteControl(LONG IUserID, DWORD dwCommand,
LPVOID lpInBuffer, DWORD dwInBufferSize);

```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_DEL\_FACE\_PARAM\_CFG

lpInBuffer

[in] 指向一个 NET\_DVR\_FACE\_PARAM\_CTRL 结构体

dwInBufferSize

[in] 一个 NET\_DVR\_FACE\_PARAM\_CTRL 结构体大小

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.3.3.2 命令码

```

#define NET_DVR_DEL_FACE_PARAM_CFG    2509 //删除人脸

```

#### 3.3.3.3 宏定义及结构体

## 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MAX_CARD_READER_NUM_512	最大读卡器数	512
MAX_FACE_NUM	最大人脸数	2

## 结构体:

```
typedef struct tagNET_DVR_FACE_PARAM_CTRL
```

```
{  
    DWORD          dwSize;  
    BYTE   byMode;           //删除方式, 0-按卡号方式删除, 1-按读卡器删除  
    BYTE   byRes1[3];        //保留  
    NET_DVR_DEL_FACE_PARAM_MODE struProcessMode; //处理方式  
    BYTE   byRes[64];        //保留  
}NET_DVR_FACE_PARAM_CTRL, *LPNET_DVR_FACE_PARAM_CTRL;
```

```
typedef union tagNET_DVR_DEL_FACE_PARAM_MODE
```

```
{  
    BYTE   uLen[588];        //联合体长度  
    NET_DVR_FACE_PARAM_BYCARD      struByCard;    //按卡号的方式删除  
    NET_DVR_FACE_PARAM_BYREADER    struByReader;  //按读卡器的方式删除  
}NET_DVR_DEL_FACE_PARAM_MODE, *LPNET_DVR_DEL_FACE_PARAM_MODE;
```

```
typedef struct tagNET_DVR_FACE_PARAM_BYCARD
```

```
{  
    BYTE   byCardNo[ACS_CARD_NO_LEN]; //人脸关联的卡号  
    BYTE   byEnableCardReader[MAX_CARD_READER_NUM_512]; //人脸的读卡器信息, 按数组  
    表示  
    BYTE   byFaceID[MAX_FACE_NUM];      //需要删除的人脸编号, 按数组下标, 值表示 0-  
    不删除, 1-删除该人脸  
    BYTE   byRes1[42];                  //保留  
}NET_DVR_FACE_PARAM_BYCARD, *LPNET_DVR_FACE_PARAM_BYCARD;
```

```
typedef struct tagNET_DVR_FACE_PARAM_BYREADER
```

```
{  
    DWORD dwCardReaderNo; //按值表示, 人脸读卡器编号  
    BYTE   byClearAllCard; //是否删除所有卡的人脸信息, 0-按卡号删除人脸信息, 1-删除所有卡  
    的人脸信息  
    BYTE   byRes1[3];      //保留  
    BYTE   byCardNo[ACS_CARD_NO_LEN]; //人脸关联的卡号  
    BYTE   byRes[548];     //保留  
}NET_DVR_FACE_PARAM_BYREADER, *LPNET_DVR_FACE_PARAM_BYREADER;
```

### 3.3.4 人脸采集

#### 3.3.4.1 接口函数

##### 1. 开启

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_StartRemoteConfig(LONG IUserID, DWORD dwCommand, LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL, LPVOID pUserData = NULL);

##### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_CAPTURE\_FACE\_INFO

lpInBuffer

[in] 指向一个 NET\_DVR\_CAPTURE\_FACE\_COND 结构体

dwInBufferSize:

[in] 一个 NET\_DVR\_CAPTURE\_FACE\_COND 结构体大小

##### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

##### 2. 逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_GetNextRemoteConfig(LONG lHandle, void\* lpOutBuff, DWORD dwOutBuffSize);

##### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

lpOutBuff

[out] 指向一个 NET\_DVR\_CAPTURE\_FACE\_CFG 结构体

dwOutBuffSize

[in] 一个 NET\_DVR\_CAPTURE\_FACE\_CFG 结构体大小

##### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据,处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据,继续调用 NET_DVR_GetNextRemoteConfig



NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完，可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常，可调用 NET_DVR_StopRemoteConfig 结束长连接

### 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG lHandle);

#### Parameters

lHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.3.4.2 命令码

#define NET\_DVR\_CAPTURE\_FACE\_INFO 2510 //采集人脸

#### 3.3.4.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值

##### 结构体:

```
typedef struct tagNET_DVR_CAPTURE_FACE_COND
{
```

DWORD dwSize;

BYTE byRes[128];

```
}NET_DVR_CAPTURE_FACE_COND, *LPNET_DVR_CAPTURE_FACE_COND;
```

```
typedef struct tagNET_DVR_CAPTURE_FACE_CFG
{
```

DWORD dwSize;

DWORD dwFaceTemplate1Size; //人脸模板 1 数据大小，等于 0 时，代表无人脸模板 1 数据

char\* pFaceTemplate1Buffer; //人脸模板 1 数据缓存（不大于 2.5k）

DWORD dwFaceTemplate2Size; //人脸模板 2 数据大小，等于 0 时，代表无人脸模板 2 数据

char\* pFaceTemplate2Buffer; //人脸模板 2 数据缓存（不大于 2.5K）

DWORD dwFacePicSize; //人脸图片数据大小，等于 0 时，代表无人脸图片数据

char\* pFacePicBuffer; //人脸图片数据缓存

BYTE byFaceQuality1; //人脸质量，范围 1-100

---

```
    BYTE    byFaceQuality2;          //人脸质量，范围 1-100
    BYTE    byCaptureProgress;      //采集进度，目前只有两种进度值：0-未采集到人脸，100-采集
    到人脸（只有在进度为 100 时，才解析人脸信息）
    BYTE    byRes1;
    DWORD    dwInfraredFacePicSize;  //红外人脸图片数据大小，等于 0 时，代表无人脸图片数据
    char*    pInfraredFacePicBuffer; //红外人脸图片数据缓存
    BYTE    byRes[116];
}NET_DVR_CAPTURE_FACE_CFG, *LPNET_DVR_CAPTURE_FACE_CFG;
```

## 3.4 远程控门

### 3.4.1 接口函数

#### 远程控门

```
NET_DVR_API BOOL __stdcall NET_DVR_ControlGateway(LONG IUserID, LONG IGatewayIndex,
DWORD dwStatic);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

IGatewayIndex

[in] 门编号（-1 表示对所有门进行操作；明眸仅支持一个门，故门编号填 1 即可）

dwStatic

[in] 命令值：0-关门，1-开门，2-常开，3-常关，4-恢复（恢复为普通状态）

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 3.5 设备事件主动上传（布防）

### 3.5.1 接口函数

报警接口调用流程参考集成中设备事件主动上传：

#### 1.设置回调函数

```
NET_DVR_API BOOL __stdcall NET_DVR_SetDVRMessageCallBack_V50(int iIndex, MSGCallBack
fMessageCallBack, void* pUser);
```

#### Parameters

iIndex

[in] 回调函数索引，取值范围：[0,15]

fMessageCallBack

[in] 回调函数

---

pUser

[in] 用户数据

### Callback Function

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

### Callback Function Parameters

lCommand

[out] COMM\_ALARM\_ACS

pAlarmer

[out] 报警设备信息，包括设备序列号、IP 地址、登录 IUserID 句柄等

pAlarmInfo

[out] 指向一个 NET\_DVR\_ACS\_ALARM\_INFO 结构体

dwBufLen

[out] 一个 NET\_DVR\_ACS\_ALARM\_INFO 结构体大小

pUser

[out] 用户数据

### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.报警布防

```
NET_DVR_API LONG __stdcall NET_DVR_SetupAlarmChan_V41(LONG IUserID,  
LPNET_DVR_SETUPALARM_PARAM lpSetupParam);
```

### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

lpSetupParam

[in] 报警布防参数

### Return Values

-1 表示失败，其他值作为 NET\_DVR\_CloseAlarmChan\_V30 函数的句柄参数。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 3.报警撤防

```
NET_DVR_API BOOL __stdcall NET_DVR_CloseAlarmChan_V30(LONG lAlarmHandle);
```

### Parameters

lAlarmHandle

[in] NET\_DVR\_SetupAlarmChan\_V41 的返回值

### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 3.5.2 命令码

```
#define COMM_ALARM_ACS    0x5002 //门禁主机报警
```

### 3.5.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MACADDR_LEN	mac 地址长度	6
MAX_NAMELEN	DVR 本地登陆名	16

#### 结构体:

```
typedef struct tagNET_DVR_ACS_EVENT_INFO
{
    DWORD dwSize;
    BYTE byCardNo[ACS_CARD_NO_LEN]; //卡号，为 0 无效
    BYTE byCardType; //卡类型，1-普通卡，2-残疾人卡，3-黑名单卡，4-巡更卡，5-胁迫卡，6-超级卡，7-来宾卡，为 0 无效
    BYTE byWhiteListNo; //白名单单号,1-8，为 0 无效
    BYTE byReportChannel; //报告上传通道，1-布防上传，2-中心组 1 上传，3-中心组 2 上传，为 0 无效
    BYTE byCardReaderKind; //读卡器属于哪一类，0-无效，1-IC 读卡器，2-身份证读卡器，3-二维码读卡器,4-指纹头
    DWORD dwCardReaderNo; //读卡器编号，为 0 无效
    DWORD dwDoorNo; //门编号(楼层编号)，为 0 无效
    DWORD dwVerifyNo; //多重卡认证序号，为 0 无效
    DWORD dwAlarmInNo; //报警输入号，为 0 无效
    DWORD dwAlarmOutNo; //报警输出号，为 0 无效
    DWORD dwCaseSensorNo; //事件触发器编号
    DWORD dwRs485No; //RS485 通道号，为 0 无效
    DWORD dwMultiCardGroupNo; //群组编号
    WORD wAccessChannel; //人员通道号
    BYTE byDeviceNo; //设备编号，为 0 无效
    BYTE byDistractControlNo; //分控器编号，为 0 无效
    DWORD dwEmployeeNo; //工号，为 0 无效
    WORD wLocalControllerID; //就地控制器编号，0-门禁主机，1-64 代表就地控制器
    BYTE byInternetAccess; //网口 ID: (1-上行网口 1,2-上行网口 2,3-下行网口 1)
    BYTE byType; //防区类型，0:即时防区,1-24 小时防区,2-延时防区 ,3-内部防区，4-钥匙防区 5-火警防区 6-周界防区 7-24 小时无声防区 8-24 小时辅助防区，9-24 小时震动防区,10-门禁紧急开门防区，11-门禁紧急关门防区 0xff-无
    BYTE byMACAddr[MACADDR_LEN]; //物理地址，为 0 无效
    BYTE bySwipeCardType; //刷卡类型，0-无效，1-二维码
```

---

```

    BYTE  byRes2;
    DWORD dwSerialNo; //事件流水号，为 0 无效
    BYTE  byRes[8];
}NET_DVR_ACS_EVENT_INFO, *LPNET_DVR_ACS_EVENT_INFO;

typedef struct tagNET_DVR_ACS_ALARM_INFO
{
    DWORD dwSize;
    DWORD dwMajor; //报警主类型（见门禁事件主次类型章节）
    DWORD dwMinor; //报警次类型（见门禁事件主次类型章节）
    NET_DVR_TIME struTime; //时间
    BYTE  sNetUser[MAX_NAMELEN]; //网络操作的用户名
    NET_DVR_IPADDR struRemoteHostAddr; //远程主机地址
    NET_DVR_ACS_EVENT_INFO struAcsEventInfo; //详细参数
    DWORD dwPicDataLen; //图片数据大小，不为 0 是表示后面带数据
    char  *pPicData;
    BYTE  byRes[24];
}NET_DVR_ACS_ALARM_INFO, *LPNET_DVR_ACS_ALARM_INFO;

typedef struct tagNET_DVR_SETUPALARM_PARAM
{
    DWORD dwSize;
    BYTE  byRes2[10];
    BYTE  byDeployType; //布防类型：0-客户端布防，1-实时布防
    BYTE  byRes1[5];
}NET_DVR_SETUPALARM_PARAM, *LPNET_DVR_SETUPALARM_PARAM;

```

### 3.5.4 备注

布防类型：0-客户端布防（支持实时事件+离线事件上传）；1-实时布防（仅支持实时事件上传，不支持离线事件上传）

## 3.6 主动获取设备事件

### 3.6.1 接口函数

#### 1. 开启

```

NET_DVR_API LONG __stdcall NET_DVR_StartRemoteConfig(LONG IUserID, DWORD dwCommand,
LPVOID lpInBuffer, DWORD dwInBufferLen, fRemoteConfigCallback cbStateCallback = NULL,
LPVOID pUserData = NULL);

```

#### Parameters

IUserID

---

[in] NET\_DVR\_Login\_V40 等登录接口的返回值  
dwCommand  
[in] NET\_DVR\_GET\_ACS\_EVENT  
lpInBuffer  
[in] 指向一个 NET\_DVR\_ACS\_EVENT\_COND 结构体  
dwInBufferSize:  
[in] 一个 NET\_DVR\_ACS\_EVENT\_COND 结构体大小

#### Return Values

-1 表示失败，其他值作为 NET\_DVR\_GetNextRemoteConfig 等接口的句柄。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

## 2.逐条获取结果

NET\_DVR\_API LONG \_\_stdcall NET\_DVR\_GetNextRemoteConfig(LONG IHandle, void\* lpOutBuff, DWORD dwOutBuffSize);

#### Parameters

IHandle  
[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值  
lpOutBuff  
[out] 指向一个 NET\_DVR\_ACS\_EVENT\_CFG 结构体  
dwOutBuffSize  
[in] 一个 NET\_DVR\_ACS\_EVENT\_CFG 结构体大小

#### Return Values

-1 表示失败，其他值表示当前的获取状态等信息，详见下表。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

宏定义	宏定义值	含义
NET_SDK_GET_NEXT_STATUS_SUCCESS	1000	成功读取到数据,处理完本次数据后需要再次调用 NET_DVR_GetNextRemoteConfig 获取下一条数据
NET_SDK_GET_NEXT_STATUS_NEED_WAIT	1001	需等待设备发送数据,继续调用 NET_DVR_GetNextRemoteConfig
NET_SDK_GET_NEXT_STATUS_FINISH	1002	数据全部取完,可调用 NET_DVR_StopRemoteConfig 结束长连接
NET_SDK_GET_NEXT_STATUS_FAILED	1003	出现异常,可调用 NET_DVR_StopRemoteConfig 结束长连接

## 3.关闭

NET\_DVR\_API BOOL \_\_stdcall NET\_DVR\_StopRemoteConfig(LONG IHandle);

#### Parameters

---

IHandle

[in] 句柄，NET\_DVR\_StartRemoteConfig 的返回值

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 3.6.2 命令码

```
#define NET_DVR_GET_ACS_EVENT    2514 //设备事件获取
```

### 3.6.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
ACS_CARD_NO_LEN	卡号长度	32
MACADDR_LEN	mac 地址长度	6
MAX_NAMELEN	DVR 本地登陆名	16

#### 结构体:

```
typedef struct tagNET_DVR_ACS_EVENT_COND
```

```
{  
    DWORD          dwSize;  
    DWORD dwMajor; //报警主类型，0-全部（见门禁事件主次类型章节）  
    DWORD dwMinor; //报警次类型，0-全部（见门禁事件主次类型章节）  
    NET_DVR_TIME    struStartTime; //开始时间  
    NET_DVR_TIME    struEndTime; //结束时间  
    BYTE byCardNo[ACS_CARD_NO_LEN]; //卡号  
    BYTE byName[NAME_LEN]; //持卡人姓名  
    BYTE byPicEnable; //是否带图片，0-不带图片，1-带图片  
    BYTE byRes2[3]; //保留  
    DWORD dwBeginSerialNo; //起始流水号（为0时默认全部）  
    DWORD dwEndSerialNo; //结束流水号（为0时默认全部）  
    BYTE byRes[244]; //保留  
}NET_DVR_ACS_EVENT_COND,*LPNET_DVR_ACS_EVENT_COND;
```

```
typedef struct tagNET_DVR_ACS_EVENT_DETAIL
```

```
{  
    DWORD dwSize;  
    BYTE byCardNo[ACS_CARD_NO_LEN]; //卡号（mac 地址），为0无效  
    BYTE byCardType; //卡类型，1-普通卡，2-残疾人卡，3-黑名单卡，4-巡更卡，5-胁迫卡，6-超级卡，7-来宾卡，8-解除卡，为0无效  
    BYTE byWhiteListNo; //白名单单号,1-8，为0无效  
    BYTE byReportChannel; //报告上传通道，1-布防上传，2-中心组1上传，3-中心组2上传，为0
```

无效

BYTE byCardReaderKind; //读卡器属于哪一类, 0-无效, 1-IC 读卡器, 2-身份证读卡器, 3-二维码读卡器, 4-指纹头

DWORD dwCardReaderNo; //读卡器编号, 为 0 无效

DWORD dwDoorNo; //门编号 (楼层编号), 为 0 无效

DWORD dwVerifyNo; //多重卡认证序号, 为 0 无效

DWORD dwAlarmInNo; //报警输入号, 为 0 无效

DWORD dwAlarmOutNo; //报警输出号, 为 0 无效

DWORD dwCaseSensorNo; //事件触发器编号

DWORD dwRs485No; //RS485 通道号, 为 0 无效

DWORD dwMultiCardGroupNo; //群组编号

WORD wAccessChannel; //人员通道号

BYTE byDeviceNo; //设备编号, 为 0 无效 (有效范围 1-255)

BYTE byDistractControlNo; //分控器编号, 为 0 无效

DWORD dwEmployeeNo; //工号, 为 0 无效

WORD wLocalControllerID; //就地控制器编号, 0-门禁主机, 1-64 代表就地控制器

BYTE byInternetAccess; //网口 ID: (1-上行网口 1, 2-上行网口 2, 3-下行网口 1)

BYTE byType; //防区类型, 0:即时防区, 1-24 小时防区, 2-延时防区, 3-内部防区, 4-钥匙防区 5-火警防区 6-周界防区 7-24 小时无声防区 8-24 小时辅助防区, 9-24 小时震动防区, 10-门禁紧急开门防区, 11-门禁紧急关门防区 0xff-无

BYTE byMACAddr[MACADDR\_LEN]; //物理地址, 为 0 无效

BYTE bySwipeCardType; //刷卡类型, 0-无效, 1-二维码

BYTE byRes2;

DWORD dwSerialNo; //事件流水号, 为 0 无效

BYTE byRes[112];

}NET\_DVR\_ACS\_EVENT\_DETAIL, \*LPNET\_DVR\_ACS\_EVENT\_DETAIL;

typedef struct tagNET\_DVR\_ACS\_EVENT\_CFG

{

DWORD dwSize;

DWORD dwMajor; //报警主类型 (见门禁事件主次类型章节)

DWORD dwMinor; //报警次类型 (见门禁事件主次类型章节)

NET\_DVR\_TIME struTime; //时间

BYTE sNetUser[MAX\_NAMELEN]; //网络操作的用户名

NET\_DVR\_IPADDR struRemoteHostAddr; //远程主机地址

NET\_DVR\_ACS\_EVENT\_DETAIL struAcsEventInfo; //详细参数

DWORD dwPicDataLen; //图片数据大小, 不为 0 是表示后面带数据

char \*pPicData;

BYTE byRes[64];

}NET\_DVR\_ACS\_EVENT\_CFG, \*LPNET\_DVR\_ACS\_EVENT\_CFG;

### 3.7 卡权限计划模板管理



---

## 3.7.1 卡权限周计划配置

### 3.7.1.1 接口函数

#### 获取卡权限周计划

```
NET_DVR_API  BOOL  __stdcall  NET_DVR_GetDVRConfig(LONG  IUserID,  DWORD  
dwCommand,LONG  IChannel,  LPVOID  lpOutBuffer,  DWORD  dwOutBufferSize,  LPDWORD  
lpBytesReturned);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_CARD\_RIGHT\_WEEK\_PLAN

IChannel

[in] 周计划编号（从 1 开始）

lpOutBuffer

[out] 指向一个 NET\_DVR\_WEEK\_PLAN\_CFG 结构体

dwOutBufferSize

[in] 一个 NET\_DVR\_WEEK\_PLAN\_CFG 结构体大小

lpBytesReturned

[out] 实际收到的数据长度指针，不能为 NULL

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 设置卡权限周计划

```
NET_DVR_API  BOOL  __stdcall  NET_DVR_SetDVRConfig(LONG  IUserID,  DWORD  
dwCommand,LONG  IChannel,  LPVOID  lpInBuffer,  DWORD  dwInBufferSize);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_SET\_CARD\_RIGHT\_WEEK\_PLAN

IChannel

[in] 周计划编号（从 1 开始）

lpInBuffer

[in] 指向一个 NET\_DVR\_WEEK\_PLAN\_CFG 结构体

dwInBufferSize

[in] 一个 NET\_DVR\_WEEK\_PLAN\_CFG 结构体大小

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，

---

通过错误码判断出错原因

### 3.7.1.2 命令码

```
#define NET_DVR_GET_CARD_RIGHT_WEEK_PLAN    2126 //获取卡权限周计划
#define NET_DVR_SET_CARD_RIGHT_WEEK_PLAN    2127 //设置卡权限周计划
```

### 3.7.1.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
MAX_DAYS	每周天数	7
MAX_TIMESEGMENT_V30	设备最大时间段数	8

#### 结构体

```
typedef struct tagNET_DVR_WEEK_PLAN_CFG
{
    DWORD dwSize;
    BYTE byEnable; //是否使能, 1-使能, 0-不使能
    BYTE byRes1[3];
    NET_DVR_SINGLE_PLAN_SEGMENT struPlanCfg[MAX_DAYS][MAX_TIMESEGMENT_V30];
//周计划参数
    BYTE byRes2[16];
}NET_DVR_WEEK_PLAN_CFG, *LPNET_DVR_WEEK_PLAN_CFG;

typedef struct tagNET_DVR_SINGLE_PLAN_SEGMENT
{
    BYTE byEnable; //是否使能, 1-使能, 0-不使能
    BYTE byRes[7];
    NET_DVR_TIME_SEGMENT struTimeSegment; //时间段参数 (要保证时间段不能重叠, 否则下
发会失败)
}NET_DVR_SINGLE_PLAN_SEGMENT, *LPNET_DVR_SINGLE_PLAN_SEGMENT;

typedef struct tagNET_DVR_TIME_SEGMENT
{
    NET_DVR_SIMPLE_DAYTIME struBeginTime; //开始时间点
    NET_DVR_SIMPLE_DAYTIME struEndTime; //结束时间点
}NET_DVR_TIME_SEGMENT, *LPNET_DVR_TIME_SEGMENT;

typedef struct tagNET_DVR_SIMPLE_DAYTIME
{
    BYTE byHour; //时
    BYTE byMinute; //分
    BYTE bySecond; //秒
```

---

```
    BYTE byRes;  
}NET_DVR_SIMPLE_DAYTIME, *LPNET_DVR_SIMPLE_DAYTIME;
```

## 3.7.2 卡权限假日计划配置

### 3.7.2.1 接口函数

#### 获取卡权限假日计划

```
NET_DVR_API  BOOL  __stdcall  NET_DVR_GetDVRConfig(LONG  IUserID,  DWORD  
dwCommand,LONG  IChannel,  LPVOID  lpOutBuffer,  DWORD  dwOutBufferSize,  LPDWORD  
lpBytesReturned);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_CARD\_RIGHT\_HOLIDAY\_PLAN

IChannel

[in] 假日计划编号（从 1 开始）

lpOutBuffer

[out] 指向一个 NET\_DVR\_HOLIDAY\_PLAN\_CFG 结构体

dwOutBufferSize

[in] 一个 NET\_DVR\_HOLIDAY\_PLAN\_CFG 结构体大小

lpBytesReturned

[out] 实际收到的数据长度指针，不能为 NULL

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 设置卡权限假日计划

```
NET_DVR_API  BOOL  __stdcall  NET_DVR_SetDVRConfig(LONG  IUserID,  DWORD  
dwCommand,LONG IChannel, LPVOID lpInBuffer, DWORD dwInBufferSize);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_SET\_CARD\_RIGHT\_HOLIDAY\_PLAN

IChannel

[in] 假日计划编号（从 1 开始）

lpInBuffer

[in] 指向一个 NET\_DVR\_HOLIDAY\_PLAN\_CFG 结构体

dwInBufferSize

---

[in] 一个 NET\_DVR\_HOLIDAY\_PLAN\_CFG 结构体大小

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

#### 3.7.2.2 命令码

```
#define NET_DVR_GET_CARD_RIGHT_HOLIDAY_PLAN    2130 //获取卡权限假日计划
#define NET_DVR_SET_CARD_RIGHT_HOLIDAY_PLAN    2131 //设置卡权限假日计划
```

#### 3.7.2.3 宏定义及结构体

##### 宏定义

宏定义	宏定义含义	宏定义值
MAX_TIMESEGMENT_V30	设备最大时间段数	8

##### 结构体

```
typedef struct tagNET_DVR_HOLIDAY_PLAN_CFG
```

```
{
    DWORD dwSize;
    BYTE byEnable; //是否使能， 1-使能， 0-不使能
    BYTE byRes1[3];
    NET_DVR_DATE struBeginDate; //假日开始日期
    NET_DVR_DATE struEndDate; //假日结束日期
    NET_DVR_SINGLE_PLAN_SEGMENT struPlanCfg[MAX_TIMESEGMENT_V30]; //时间段参数
    BYTE byRes2[16];
}NET_DVR_HOLIDAY_PLAN_CFG, *LPNET_DVR_HOLIDAY_PLAN_CFG;
```

```
typedef struct tagNET_DVR_DATE
```

```
{
    WORD wYear;
    BYTE byMonth;
    BYTE byDay;
}NET_DVR_DATE, *LPNET_DVR_DATE;
```

```
typedef struct tagNET_DVR_SINGLE_PLAN_SEGMENT
```

```
{
    BYTE byEnable; //是否使能， 1-使能， 0-不使能
    BYTE byRes[7];
    NET_DVR_TIME_SEGMENT struTimeSegment; //时间段参数（要保证时间段不能重叠，否则下
发会失败）
}NET_DVR_SINGLE_PLAN_SEGMENT, *LPNET_DVR_SINGLE_PLAN_SEGMENT;
```

---

```

typedef struct tagNET_DVR_TIME_SEGMENT
{
    NET_DVR_SIMPLE_DAYTIME struBeginTime; //开始时间点
    NET_DVR_SIMPLE_DAYTIME struEndTime;   //结束时间点
}NET_DVR_TIME_SEGMENT, *LPNET_DVR_TIME_SEGMENT;

typedef struct tagNET_DVR_SIMPLE_DAYTIME
{
    BYTE byHour; //时
    BYTE byMinute; //分
    BYTE bySecond; //秒
    BYTE byRes;
}NET_DVR_SIMPLE_DAYTIME, *LPNET_DVR_SIMPLE_DAYTIME;

```

### 3.7.3 卡权限假日组配置

#### 3.7.3.1 接口函数

##### 获取卡权限假日组

```

NET_DVR_API  BOOL  __stdcall  NET_DVR_GetDVRConfig(LONG  IUserID,  DWORD
dwCommand, LONG  IChannel, LPVOID  lpOutBuffer,  DWORD  dwOutBufferSize, LPDWORD
lpBytesReturned);

```

##### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_CARD\_RIGHT\_HOLIDAY\_GROUP

IChannel

[in] 假日组编号（从 1 开始）

lpOutBuffer

[out] 指向一个 NET\_DVR\_HOLIDAY\_GROUP\_CFG 结构体

dwOutBufferSize

[in] 一个 NET\_DVR\_HOLIDAY\_GROUP\_CFG 结构体大小

lpBytesReturned

[out] 实际收到的数据长度指针，不能为 NULL

##### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

##### 设置卡权限假日组

```

NET_DVR_API  BOOL  __stdcall  NET_DVR_SetDVRConfig(LONG  IUserID,  DWORD
dwCommand, LONG IChannel, LPVOID lpInBuffer, DWORD dwInBufferSize);

```

Parameters

lUserID  
[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand  
[in] NET\_DVR\_SET\_CARD\_RIGHT\_HOLIDAY\_GROUP

lChannel  
[in] 假日组编号（从 1 开始）

lpInBuffer  
[in] 指向一个 NET\_DVR\_HOLIDAY\_GROUP\_CFG 结构体

dwInBufferSize  
[in] 一个 NET\_DVR\_HOLIDAY\_GROUP\_CFG 结构体大小

Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

3.7.3.2 命令码

#define NET\_DVR\_GET\_CARD\_RIGHT\_HOLIDAY\_GROUP 2134 //获取卡权限假日组

#define NET\_DVR\_SET\_CARD\_RIGHT\_HOLIDAY\_GROUP 2135 //设置卡权限假日组

3.7.3.3 宏定义及结构体

宏定义

宏定义	宏定义含义	宏定义值
HOLIDAY_GROUP_NAME_LEN	假日组名称长度	32
MAX_HOLIDAY_PLAN_NUM	假日组最大假日计划数	16

结构体

```
typedef struct tagNET_DVR_HOLIDAY_GROUP_CFG
{
    DWORD dwSize;
    BYTE byEnable; //是否启用，1-启用，0-不启用
    BYTE byRes1[3];
    BYTE byGroupName[HOLIDAY_GROUP_NAME_LEN]; //假日组名称
    DWORD dwHolidayPlanNo[MAX_HOLIDAY_PLAN_NUM]; //假日计划编号，就前填充，遇 0 无效
    BYTE byRes2[32];
}NET_DVR_HOLIDAY_GROUP_CFG, *LPNET_DVR_HOLIDAY_GROUP_CFG;
```

3.7.4 卡权限计划模板配置

3.7.4.1 接口函数

---

### 获取卡权限计划模板

```
NET_DVR_API  BOOL  __stdcall  NET_DVR_GetDVRConfig(LONG  IUserID,  DWORD
dwCommand, LONG  IChannel,  LPVOID  lpOutBuffer,  DWORD  dwOutBufferSize,  LPDWORD
lpBytesReturned);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_GET\_CARD\_RIGHT\_PLAN\_TEMPLATE

IChannel

[in] 计划模板编号（从 1 开始）

lpOutBuffer

[out] 指向一个 NET\_DVR\_PLAN\_TEMPLATE 结构体

dwOutBufferSize

[in] 一个 NET\_DVR\_PLAN\_TEMPLATE 结构体大小

lpBytesReturned

[out] 实际收到的数据长度指针，不能为 NULL

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 设置卡权限计划模板

```
NET_DVR_API  BOOL  __stdcall  NET_DVR_SetDVRConfig(LONG  IUserID,  DWORD
dwCommand, LONG  IChannel, LPVOID  lpInBuffer, DWORD  dwInBufferSize);
```

#### Parameters

IUserID

[in] NET\_DVR\_Login\_V40 等登录接口的返回值

dwCommand

[in] NET\_DVR\_SET\_CARD\_RIGHT\_PLAN\_TEMPLATE

IChannel

[in] 计划模板编号（从 1 开始）

lpInBuffer

[in] 指向一个 NET\_DVR\_PLAN\_TEMPLATE 结构体

dwInBufferSize

[in] 一个 NET\_DVR\_PLAN\_TEMPLATE 结构体大小

#### Return Values

TRUE 表示成功，FALSE 表示失败。接口返回失败请调用 NET\_DVR\_GetLastError 获取错误码，通过错误码判断出错原因

### 3.7.4.2 命令码

```
#define NET_DVR_GET_CARD_RIGHT_PLAN_TEMPLATE  2138 //获取卡权限计划模板
```

---

```
#define NET_DVR_SET_CARD_RIGHT_PLAN_TEMPLATE    2139 //设置卡权限计划模板
```

### 3.7.4.3 宏定义及结构体

#### 宏定义

宏定义	宏定义含义	宏定义值
TEMPLATE_NAME_LEN	计划模板名称长度	32
MAX_HOLIDAY_GROUP_NUM	计划模板最大假日组数	16

#### 结构体

```
typedef struct tagNET_DVR_PLAN_TEMPLATE
{
    DWORD dwSize;
    BYTE byEnable; //是否启用, 1-启用, 0-不启用
    BYTE byRes1[3];
    BYTE byTemplateName[TEMPLATE_NAME_LEN]; //模板名称
    DWORD dwWeekPlanNo; //周计划编号, 0 为无效
    DWORD dwHolidayGroupNo[MAX_HOLIDAY_GROUP_NUM]; //假日组编号, 就前填充, 遇 0 无效
    BYTE byRes2[32];
}NET_DVR_PLAN_TEMPLATE, *LPNET_DVR_PLAN_TEMPLATE;
```

## 3.8 门禁事件主次类型

```
/* 报警 */
//主类型
#define MAJOR_ALARM    0x1
//次类型
#define MINOR_HOST_DESMANTLE_ALARM    0x404 //设备防拆报警
#define MINOR_HOST_DESMANTLE_RESUME    0x405 //设备防拆恢复
#define MINOR_CARD_READER_DESMANTLE_ALARM    0x406 //读卡器防拆报警
#define MINOR_CARD_READER_DESMANTLE_RESUME    0x407 //读卡器防拆恢复
#define MINOR_CASE_SENSOR_ALARM    0x408 //事件输入报警
#define MINOR_CASE_SENSOR_RESUME    0x409 //事件输入恢复
#define MINOR_STRESS_ALARM    0x40a //胁迫报警
#define MINOR_OFFLINE_ECENET_NEARLY_FULL    0x40b //离线事件满 90%报警
#define MINOR_CARD_MAX_AUTHENTICATE_FAIL    0x40c //卡号认证失败超次报警
#define MINOR_SD_CARD_FULL    0x40d //SD 卡存储满报警
#define MINOR_SECURITY_MODULE_DESMANTLE_ALARM    0x40f //门控安全模块防拆报警
#define MINOR_SECURITY_MODULE_DESMANTLE_RESUME    0x410 //门控安全模块防拆恢复

/* 异常 */
```



---

```

//主类型
#define MAJOR_EXCEPTION      0x2
//次类型
#define MINOR_NET_BROKEN      0x27 //网络断开
#define MINOR_DEV_POWER_ON    0x400 //设备上电启动
#define MINOR_NET_RESUME      0x407 //网络恢复
#define MINOR_FLASH_ABNORMAL  0x408 //FLASH 读写异常
#define MINOR_CARD_READER_OFFLINE 0x409 //读卡器掉线
#define MINOR_CARD_READER_RESUME 0x40a //读卡器掉线恢复
#define MINOR_SECURITY_MODULE_OFF 0x40f //门控安全模块掉线
#define MINOR_SECURITY_MODULE_RESUME 0x410 //门控安全模块在线
#define MINOR_ID_CARD_READER_NOT_CONNECT 0x41d //身份证阅读器未连接
#define MINOR_ID_CARD_READER_RESUME 0x41e //身份证阅读器连接恢复
#define MINOR_COM_NOT_CONNECT 0x423 //COM 口未连接
#define MINOR_COM_RESUME      0x424 //COM 口连接恢复
#define MINOR_PEOPLE_AND_ID_CARD_DEVICE_ONLINE 0x426 //人证设备在线
#define MINOR_PEOPLE_AND_ID_CARD_DEVICE_OFFLINE 0x427 //人证设备离线
#define MINOR_LOCAL_LOGIN_LOCK 0x428 //本地登录锁定
#define MINOR_LOCAL_LOGIN_UNLOCK 0x429 //本地登录解锁

/* 操作 */
//主类型
#define MAJOR_OPERATION      0x3
//次类型
#define MINOR_LOCAL_LOGIN    0x50 //本地登陆
#define MINOR_LOCAL_UPGRADE  0x5a //本地升级
#define MINOR_REMOTE_LOGIN    0x70 //远程登录
#define MINOR_REMOTE_LOGOUT   0x71 //远程注销登陆
#define MINOR_REMOTE_ARM      0x79 //远程布防
#define MINOR_REMOTE_DISARM   0x7a //远程撤防
#define MINOR_REMOTE_REBOOT    0x7b //远程重启
#define MINOR_REMOTE_UPGRADE  0x7e //远程升级
#define MINOR_REMOTE_CFGFILE_OUTPUT 0x86 //远程导出配置文件
#define MINOR_REMOTE_CFGFILE_INPUT 0x87 //远程导入配置文件
#define MINOR_REMOTE_OPEN_DOOR 0x400 //远程开门
#define MINOR_REMOTE_CLOSE_DOOR 0x401 //远程关门(受控)
#define MINOR_REMOTE_ALWAYS_OPEN 0x402 //远程常开(自由)
#define MINOR_REMOTE_ALWAYS_CLOSE 0x403 //远程常关(禁用)
#define MINOR_REMOTE_CHECK_TIME 0x404 //远程手动校时
#define MINOR_NTP_CHECK_TIME 0x405 //NTP 自动校时
#define MINOR_REMOTE_CLEAR_CARD 0x406 //远程清空卡号
#define MINOR_REMOTE_RESTORE_CFG 0x407 //远程恢复默认参数
#define MINOR_LOCAL_RESTORE_CFG 0x40a //本地恢复默认参数
#define MINOR_REMOTE_CAPTURE_PIC 0x40b //远程抓拍

```

---

```

#define MINOR_MOD_NET_REPORT_CFG          0x40c //修改网络中心参数配置
#define MINOR_MOD_REPORT_GROUP_PARAM      0x40e //修改中心组参数配置
#define MINOR_UNLOCK_PASSWORD_OPEN_DOOR   0x40f //解除码输入
#define MINOR_REMOTE_ACTUAL_GUARD          0x419 //远程实时布防
#define MINOR_REMOTE_ACTUAL_UNGUARD        0x41a //远程实时撤防

/*事件*/
//主类型
#define MAJOR_EVENT      0x5
//次类型
#define MINOR_LEGAL_CARD_PASS          0x01 //合法卡认证通过
#define MINOR_CARD_AND_PSW_PASS        0x02 //刷卡加密码认证通过
#define MINOR_CARD_AND_PSW_FAIL        0x03 //刷卡加密码认证失败
#define MINOR_CARD_AND_PSW_TIMEOUT     0x04 //数卡加密码认证超时
#define MINOR_CARD_AND_PSW_OVER_TIME   0x05 //刷卡加密码超次
#define MINOR_CARD_NO_RIGHT             0x06 //未分配权限
#define MINOR_CARD_INVALID_PERIOD       0x07 //无效时段
#define MINOR_CARD_OUT_OF_DATE          0x08 //卡号过期
#define MINOR_INVALID_CARD              0x09 //无此卡号
#define MINOR_ANTI_SNEAK_FAIL           0x0a //反潜回认证失败
#define MINOR_NOT_BELONG_MULTI_GROUP    0x0c //卡不属于多重认证群组
#define MINOR_INVALID_MULTI_VERIFY_PERIOD 0x0d //卡不在多重认证时间段内
#define MINOR_MULTI_VERIFY_SUPER_RIGHT_FAIL 0x0e //多重认证模式超级权限认证失败
#define MINOR_MULTI_VERIFY_REMOTE_RIGHT_FAIL 0x0f //多重认证模式远程认证失败
#define MINOR_MULTI_VERIFY_SUCCESS      0x10 //多重认证成功
#define MINOR_LEADER_CARD_OPEN_BEGIN    0x11 //首卡开门开始
#define MINOR_LEADER_CARD_OPEN_END      0x12 //首卡开门结束
#define MINOR_ALWAYS_OPEN_BEGIN         0x13 //常开状态开始
#define MINOR_ALWAYS_OPEN_END           0x14 //常开状态结束
#define MINOR_LOCK_OPEN                 0x15 //门锁打开
#define MINOR_LOCK_CLOSE                0x16 //门锁关闭
#define MINOR_DOOR_BUTTON_PRESS         0x17 //开门按钮打开
#define MINOR_DOOR_BUTTON_RELEASE       0x18 //开门按钮放开
#define MINOR_DOOR_OPEN_NORMAL          0x19 //正常开门（门磁）
#define MINOR_DOOR_CLOSE_NORMAL         0x1a //正常关门（门磁）
#define MINOR_DOOR_OPEN_ABNORMAL        0x1b //门异常打开（门磁）
#define MINOR_DOOR_OPEN_TIMEOUT         0x1c //门打开超时（门磁）
#define MINOR_ALARMOUT_ON               0x1d //报警输出打开
#define MINOR_ALARMOUT_OFF              0x1e //报警输出关闭
#define MINOR_ALWAYS_CLOSE_BEGIN        0x1f //常关状态开始
#define MINOR_ALWAYS_CLOSE_END          0x20 //常关状态结束
#define MINOR_MULTI_VERIFY_NEED_REMOTE_OPEN 0x21 //多重多重认证需要远程开门
#define MINOR_MULTI_VERIFY_SUPERPASSWD_VERIFY_SUCCESS 0x22 //多重认证超级密

```

---

码认证成功事件

#define MINOR_MULTI_VERIFY_REPEAT_VERIFY	0x23	//多重认证重复认证事件
#define MINOR_MULTI_VERIFY_TIMEOUT	0x24	//多重认证重复认证事件
#define MINOR_DOORBELL_RINGING	0x25	//门铃响
#define MINOR_FINGERPRINT_COMPARE_PASS	0x26	//指纹比对通过
#define MINOR_FINGERPRINT_COMPARE_FAIL	0x27	//指纹比对失败
#define MINOR_CARD_FINGERPRINT_VERIFY_PASS	0x28	//刷卡加指纹认证通过
#define MINOR_CARD_FINGERPRINT_VERIFY_FAIL	0x29	//刷卡加指纹认证失败
#define MINOR_CARD_FINGERPRINT_VERIFY_TIMEOUT	0x2a	//刷卡加指纹认证超时
#define MINOR_CARD_FINGERPRINT_PASSWD_VERIFY_PASS 认证成功	0x2b	//刷卡加指纹加密码认 证通过
#define MINOR_CARD_FINGERPRINT_PASSWD_VERIFY_FAIL 认证失败	0x2c	//刷卡加指纹加密码认 证失败
#define MINOR_CARD_FINGERPRINT_PASSWD_VERIFY_TIMEOUT 码认证超时	0x2d	//刷卡加指纹加密 码认证超时
#define MINOR_FINGERPRINT_PASSWD_VERIFY_PASS	0x2e	//指纹加密码认证通过
#define MINOR_FINGERPRINT_PASSWD_VERIFY_FAIL	0x2f	//指纹加密码认证失败
#define MINOR_FINGERPRINT_PASSWD_VERIFY_TIMEOUT	0x30	//指纹加密码认证超时
#define MINOR_FINGERPRINT_INEXISTENCE	0x31	//指纹不存在
#define MINOR_CALL_CENTER	0x33	//呼叫中心事件
#define MINOR_FACE_AND_FP_VERIFY_PASS	0x36	//人脸加指纹认证通过
#define MINOR_FACE_AND_FP_VERIFY_FAIL	0x37	//人脸加指纹认证失败
#define MINOR_FACE_AND_FP_VERIFY_TIMEOUT	0x38	//人脸加指纹认证超时
#define MINOR_FACE_AND_PW_VERIFY_PASS	0x39	//人脸加密码认证通过
#define MINOR_FACE_AND_PW_VERIFY_FAIL	0x3a	//人脸加密码认证失败
#define MINOR_FACE_AND_PW_VERIFY_TIMEOUT	0x3b	//人脸加密码认证超时
#define MINOR_FACE_AND_CARD_VERIFY_PASS	0x3c	//人脸加刷卡认证通过
#define MINOR_FACE_AND_CARD_VERIFY_FAIL	0x3d	//人脸加刷卡认证失败
#define MINOR_FACE_AND_CARD_VERIFY_TIMEOUT	0x3e	//人脸加刷卡认证超时
#define MINOR_FACE_AND_PW_AND_FP_VERIFY_PASS	0x3f	//人脸加密码加指纹认证通过
#define MINOR_FACE_AND_PW_AND_FP_VERIFY_FAIL	0x40	//人脸加密码加指纹认证失败
#define MINOR_FACE_AND_PW_AND_FP_VERIFY_TIMEOUT 纹认证超时	0x41	//人脸加密码加指 纹认证超时
#define MINOR_FACE_CARD_AND_FP_VERIFY_PASS 认证成功	0x42	//人脸加刷卡加指纹 认证成功
#define MINOR_FACE_CARD_AND_FP_VERIFY_FAIL 认证失败	0x43	//人脸加刷卡加指纹 认证失败
#define MINOR_FACE_CARD_AND_FP_VERIFY_TIMEOUT 纹认证超时	0x44	//人脸加刷卡加指 纹认证超时
#define MINOR_EMPLOYEEENO_AND_FP_VERIFY_PASS	0x45	//工号加指纹认证通过
#define MINOR_EMPLOYEEENO_AND_FP_VERIFY_FAIL	0x46	//工号加指纹认证失败
#define MINOR_EMPLOYEEENO_AND_FP_VERIFY_TIMEOUT	0x47	//工号加指纹认证超时
#define MINOR_EMPLOYEEENO_AND_FP_AND_PW_VERIFY_PASS 密码认证通过	0x48	//工号加指纹加 密码认证通过

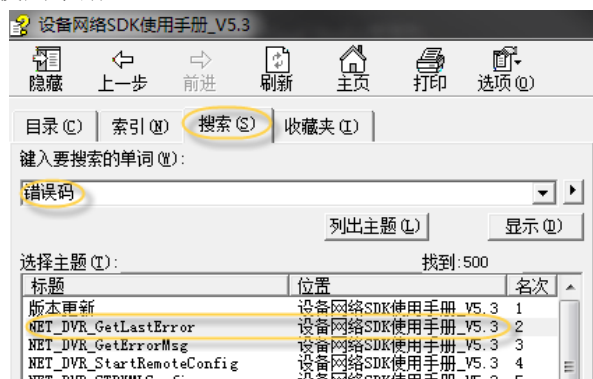
```

#define MINOR_EMPLOYEEENO_AND_FP_AND_PW_VERIFY_FAIL      0x49 //工号加指纹加密
密码认证失败
#define MINOR_EMPLOYEEENO_AND_FP_AND_PW_VERIFY_TIMEOUT  0x4a //工号加指纹加
密码认证超时
#define MINOR_FACE_VERIFY_PASS                            0x4b //人脸认证通过
#define MINOR_FACE_VERIFY_FAIL                            0x4c //人脸认证失败
#define MINOR_EMPLOYEEENO_AND_FACE_VERIFY_PASS          0x4d //工号加人脸认证通过
#define MINOR_EMPLOYEEENO_AND_FACE_VERIFY_FAIL          0x4e //工号加人脸认证失败
#define MINOR_EMPLOYEEENO_AND_FACE_VERIFY_TIMEOUT       0x4f //工号加人脸认证超时
#define MINOR_FACE_RECOGNIZE_FAIL                        0x50 //人脸识别失败
#define MINOR_FIRSTCARD_AUTHORIZE_BEGIN                  0x51 //首卡授权开始
#define MINOR_FIRSTCARD_AUTHORIZE_END                    0x52 //首卡授权结束
#define MINOR_EMPLOYEEENO_AND_PW_PASS                   0x65 //工号加密码认证通过
#define MINOR_EMPLOYEEENO_AND_PW_FAIL                   0x66 //工号加密码认证失败
#define MINOR_EMPLOYEEENO_AND_PW_TIMEOUT                 0x67 //工号加密码认证超时
#define MINOR_HUMAN_DETECT_FAIL                          0x68 //真人检测失败
#define MINOR_PEOPLE_AND_ID_CARD_COMPARE_PASS           0x69 //人证比对通过
#define MINOR_PEOPLE_AND_ID_CARD_COMPARE_FAIL           0x70 //人证比对失败
#define MINOR_CERTIFICATE_BLACK_LIST                    0x71 //黑名单事件

```

### 3.9 错误码表

详情见《设备网络 SDK 使用手册》：



## NET\_DVR\_GetLastError

返回最后操作的错误码。

```
DWORD NET_DVR_GetLastError(  
);
```

### Return Values

返回值为错误码，错误码主要分为网络通讯库、RTSP通讯库、软解码库、设备对接库等，详见下表。

### 网络通讯库错误码

错误类型	错误值	错误信息
NET_DVR_NOERROR	0	没有错误。
NET_DVR_PASSWORD_ERROR	1	用户名密码错误，注册时输入的用户名或者密码错误。
NET_DVR_NOENOUGHPRI	2	权限不足，一般和通道相关，例如有预览通道1权限，无预览通道2权限，即有预览权限但不完全，预览通道2返回此错误。
NET_DVR_NOINIT	3	SDK未初始化。
NET_DVR_CHANNEL_ERROR	4	通道号错误，设备没有对应的通道号。
NET_DVR_OVER_MAXLINK	5	设备号的连接数超过最大。
NET_DVR_VERSIONMISMATCH	6	版本不匹配，SDK和设备的版本不匹配。
NET_DVR_NETWORK_FAIL_CONNECT	7	连接设备失败，设备不在线或网络原因引起的连接超时等。
NET_DVR_NETWORK_SEND_ERROR	8	向设备发送失败。
NET_DVR_NETWORK_RECV_ERROR	9	从设备接收数据失败。
NET_DVR_NETWORK_RECV_TIMEOUT	10	从设备接收数据超时。
NET_DVR_NETWORK_ERRORDATA	11	传递的数据有误，发送给设备或者从设备接收到的数据错误，如远程参数配置时输入设备不支持的值。
NET_DVR_ORDER_ERROR	12	调用次序错误。
NET_DVR_OPERNOPERMIT	13	无此权限，用户对某个功能模块的权限，例如无预览权限用户预览返回此错误。
NET_DVR_COMMANDTIMEOUT	14	设备命令执行超时。
NET_DVR_ERRORSERIALPORT	15	串口号错误，指定的设备串口号不存在。
NET_DVR_ERRORALARMPORT	16	报警端口错误，指定的设备报警输入或者输出端口不存在。
NET_DVR_PARAMETER_ERROR	17	参数错误，SDK接口中给定的输入或输出参数为空，或者参数格式或值不符合要求。
NET_DVR_CHAN_EXCEPTION	18	设备通道处于错误状态。
NET_DVR_NODISK	19	设备无硬盘，当设备无硬盘时，对设备的录像文件、硬盘配置等操作失败。
NET_DVR_ERRORDISKNUM	20	硬盘号错误，当对设备进行硬盘管理操作时，指定的硬盘号不存在时返回此错误。
NET_DVR_DISK_FULL	21	设备硬盘满。
NET_DVR_DISK_ERROR	22	设备硬盘出错。
NET_DVR_NOSUPPORT	23	设备不支持。

## 4. FAQ

- 1.下发人脸图片大小限制 200k 以内；
- 2.默认卡权限计划模板编号为 1，该计划模板全天 24 小时有效；
- 3.卡、指纹、人脸之间通过卡号（CardNo）进行关联，设备内的卡号要保证唯一性；
- 4.卡、卡权限计划模板之前通过计划模板编号（CardRightPlan）进行关联；
- 5.下发指纹、人脸前，首先要保证卡（包含卡号、人员信息及权限信息；如实际场景中没有真实卡，CardNo 可填写虚拟卡号，但要保证唯一性）已经下发；
- 6.删除卡时，首先要保证卡所关联的指纹、人脸已经删除；