

明眸产品集成指导手册 (以人为中心)



1. 集成流程介绍

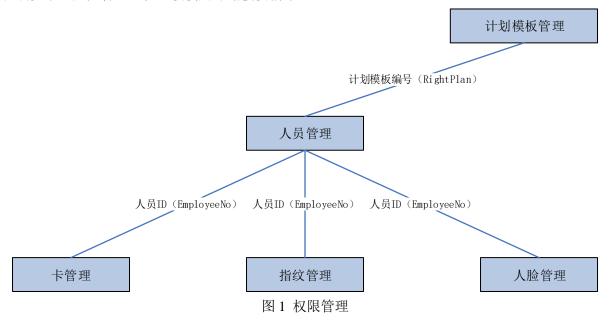
1.1 整体流程

1.1.1 权限管理

权限管理的整体流程如下图所示:

- 1.人员管理和计划模板管理之间通过计划模板编号(RightPlan)关联;
- 2.人员管理和卡管理、指纹管理、人脸管理之间通过人员 ID (employeeNo) 关联;
- 3.在下发卡权限/指纹权限/人脸权限前,首先要先下发人员权限;
- 4.在删除人员权限时,同时也会删除人员所关联的卡权限、指纹权限、人脸权限;也可以单独 对该人员的卡权限/指纹权限/人脸权限进行删除。

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,人员管理可以直接使用计划模板编号 1)。



1.1.2 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。

1.1.3 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。



1.2 人员管理

1.2.1 功能介绍

用于下发及获取设备人员相关信息:如人员 ID (employeeNo)、姓名、人员权限计划模板等相关信息。

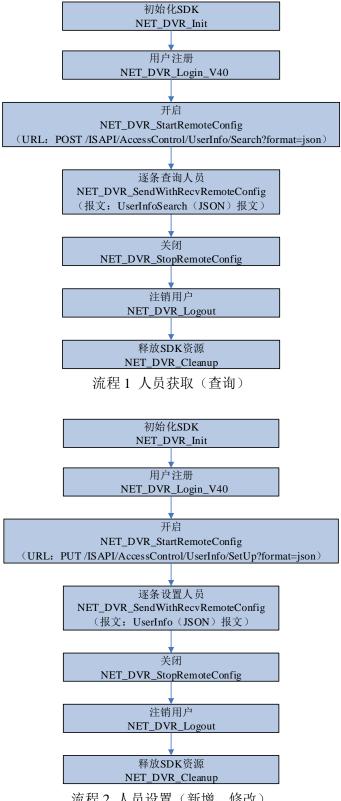
1.2.2 功能示例



图 2 人员管理示例

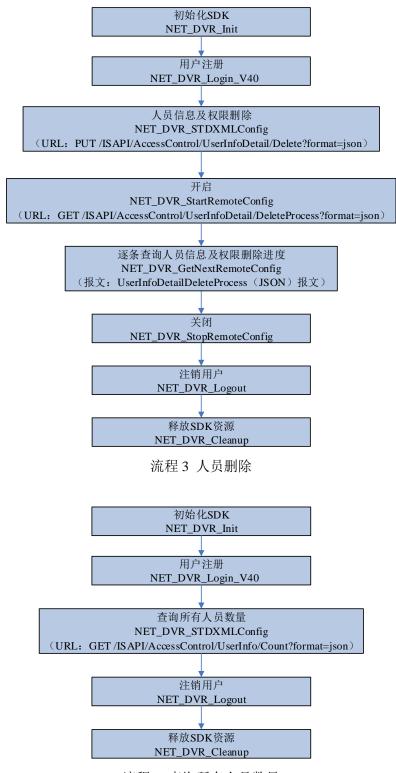
1.2.3 集成流程





流程 2 人员设置 (新增、修改)





流程 4 查询所有人员数量

1.2.4 备注

在下发人员时:

1.计划模板(RightPlan),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 RightPlan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,可参考人员权限计划模板



管理流程;

2.设备根据人员 ID (employeeNo) 判定人员不存在时,则添加该人员信息;设备根据人员 ID (employeeNo) 判定人员存在时,则修改该人员信息;

在删除人员时:

3.删除人员的同时,也会同时删除该人员所关联的卡、指纹、人脸信息。

1.3 卡管理

1.3.1 功能介绍

用于下发及获取设备卡相关信息:如工号、卡号、卡类型等相关信息。

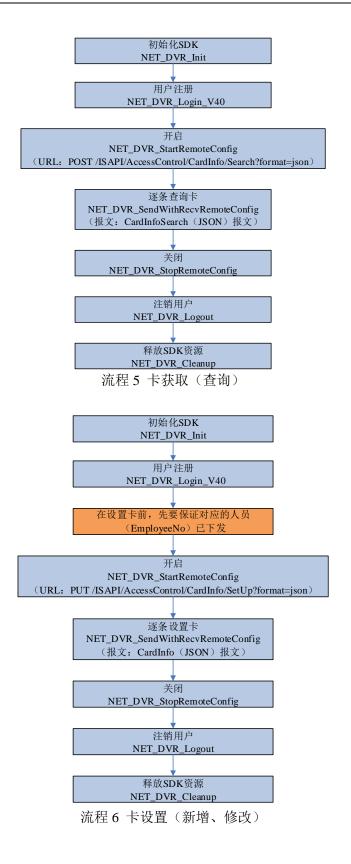
1.3.2 功能示例



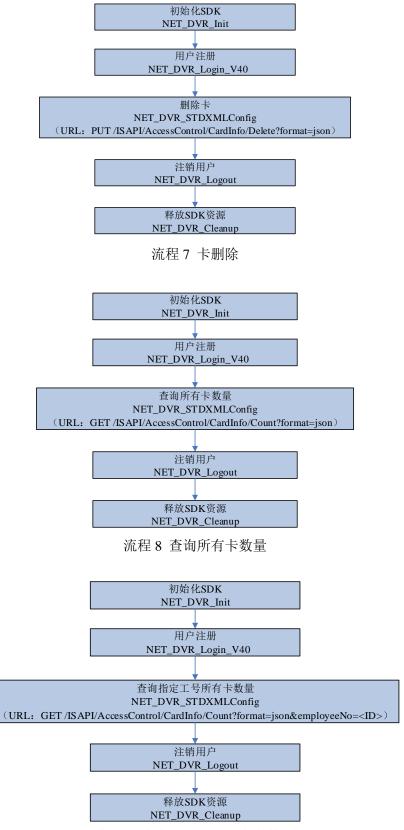
图 3 卡管理示例

1.3.3 集成流程









流程9 查询指定工号所有卡数量

1.3.4 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.设备根据卡号判定卡不存在时,则添加该卡信息;设备根据卡号判定卡存在时,则修改该卡信息。

1.4 指纹管理

1.4.1 功能介绍

指纹获取:从门禁设备中获取某个人员 ID (employeeNo) 所关联的指纹信息;

指纹下发: 下发某个人员 ID (employeeNo) 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。

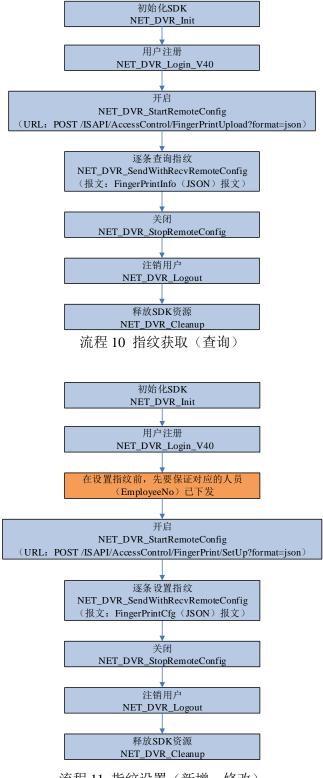
1.4.2 功能示例



图 4 指纹管理示例

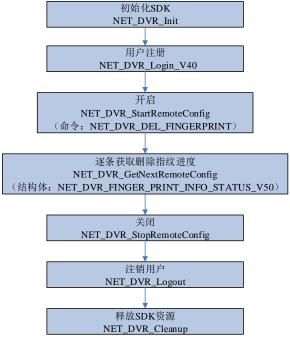
1.4.3 集成流程



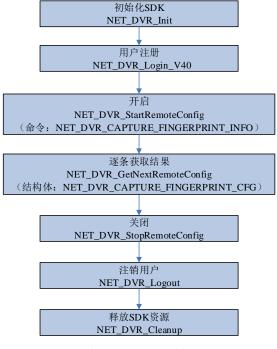


流程11指纹设置(新增、修改)





流程 12 指纹删除



流程 13 指纹采集

1.4.4 备注

- 1.在下发指纹前,需要先下发人员;
- 2.人员和指纹之间通过人员 ID (employeeNo) 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个人员所关联的指纹(1 个人最多关联 10 枚指纹),如果想要再获取其他人员的指纹,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)。

1.5 人脸管理

1.5.1 功能介绍

人脸获取:从门禁设备中获取某个人所关联的人脸信息;

人脸下发: 下发某个人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。

1.5.2 功能示例

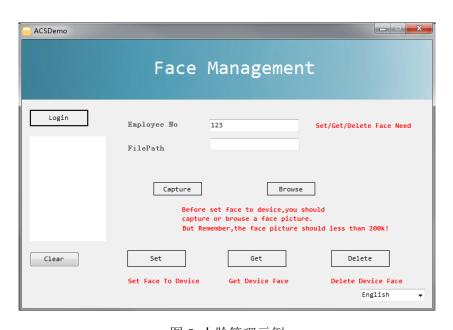
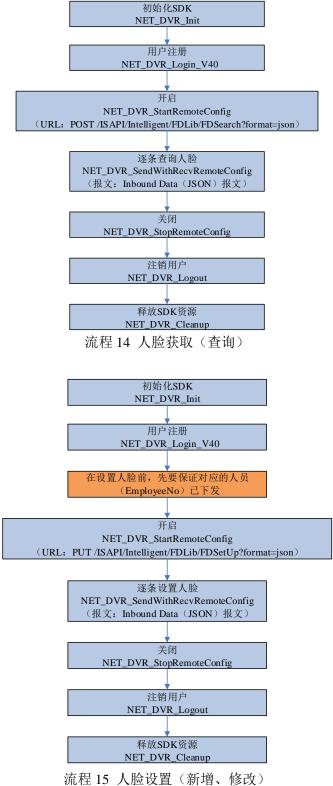


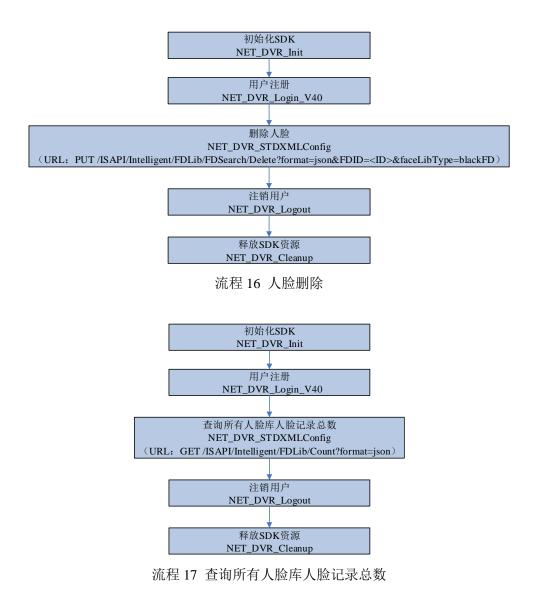
图 5 人脸管理示例

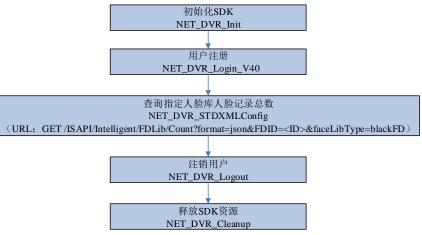
1.5.3 集成流程





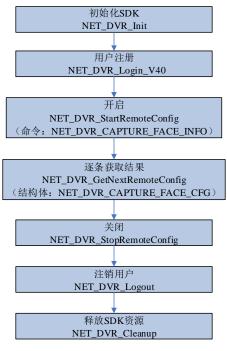






流程 18 查询指定人脸库人脸记录总数





流程 19 人脸采集

1.5.4 备注

- 1.在下发人脸前,需要先下发人员;
- 2.人员和人脸之间通过人员 ID (employeeNo) 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.人脸图片的大小最大不超过 200k;
- 5.门禁设备只有一个人脸库(默认,为 1,不需要单独创建。即**:** URL 中的 FDID=1,报文中的 "FDID": "1")。

1.6 远程控门

1.6.1 功能介绍

用于控制门禁设备门的开、关、常开、常关。

1.6.2 功能示例





图 6 远程控门示例

1.6.3 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 20 远程控门

1.6.4 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。

1.7 设备事件主动上传(布防)

1.7.1 功能介绍

用于实时获取门禁设备的人员刷卡、异常等信息。



1.7.2 功能示例

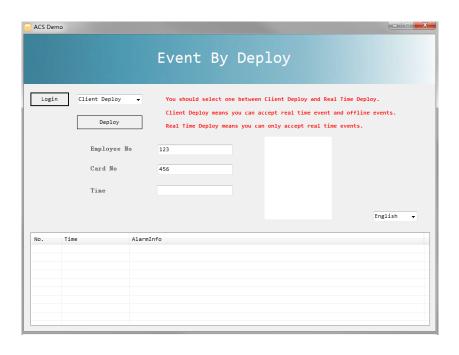
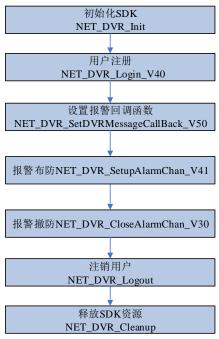


图 7 事件上传示例

1.7.3 集成流程



流程 21 设备事件主动上传 (布防)

1.7.4 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海康威视版权所有 17



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)。

1.8 主动获取设备事件

1.8.1 功能介绍

用于事后从门禁设备查询人员刷卡、异常等信息。

1.8.2 功能示例

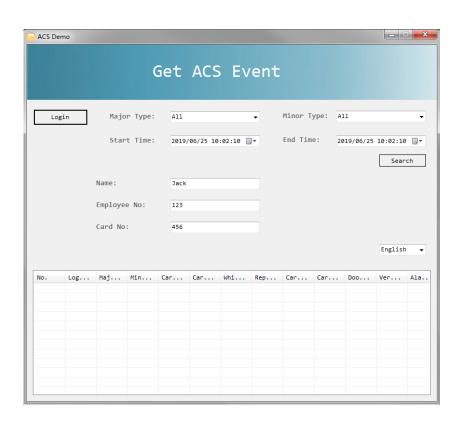


图 8 主动获取设备事件示例

1.8.3 集成流程





流程 22 主动获取设备事件流程

1.8.4 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。

1.9 人员权限计划模板管理

1.9.1 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)。

1.9.2 功能示例



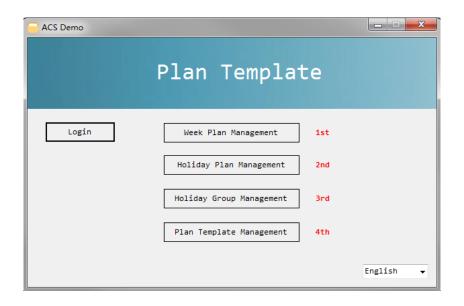
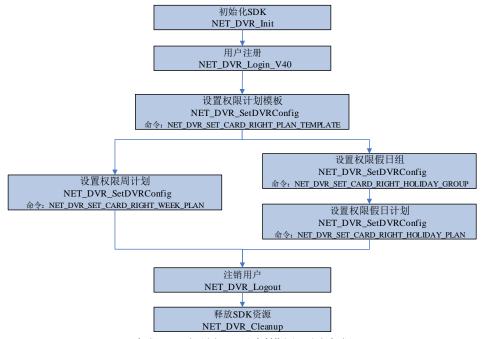


图 9 人员权限计划模板配置示例

1.9.3 集成流程



流程 23 人员权限计划模板配置流程

1.9.4 备注

1. 人 员 权 限 计 划 模 板 配 置 : 设 置 人 员 权 限 计 划 模 板 (NET_DVR_SET_CARD_RIGHT_PLAN_TEMPLATE) <—— 设置 人 员 权 限 周 计 划 (NET_DVR_SET_CARD_RIGHT_WEEK_PLAN) 、 设 置 人 员 权 限 假 日 组 (NET_DVR_SET_CARD_RIGHT_HOLIDAY_GROUP) <—— 设置 人 员 权 限 假 日 计 划 (NET_DVR_SET_CARD_RIGHT_HOLIDAY_PLAN)。



2.这里只是配置好了人员权限计划模板,下发人员时关联配置的模板后,该人员就拥有了配置的权限。

2. 接口附录

2.1人员管理

2.1.1 人员获取(查询)

2.1.1.1 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_JSON_CONFIG

lpInBuffer

[in] POST /ISAPI/AccessControl/UserInfo/Search?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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

lnInBuff

[in] 指向一个 UserInfoSearchCond (JSON) 报文

dwInBuffSize

[in] 一个 UserInfoSearchCond (JSON) 报文大小

lpOutBuff

[out] 指向一个 UserInfoSearch (JSON) 报文/ResponseStatus (JSON) 报文



dwOutBuffSize

[in] 一个 UserInfoSearch(JSON)报文/ResponseStatus(JSON)报文大小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

TELED TREGGERANCE SCHOOL STATE OF THE SECOND SCHOOL	THE CONTRACT WAS A SECTION OF THE WAY A SECTION OF		
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

2.1.1.2 命令码

#define NET_DVR_JSON_CONFIG 2550 //JSON 透传数据

2.1.1.3 JSON 报文

URL: POST/ISAPI/AccessControl/UserInfo/Search?format=json

UserInfoSearchCond(JSON)报文

```
{
    "UserInfoSearchCond": {
        "searchID": "", //必填, string, 搜索记录唯一标识, 用来确认上层客户端是否为
```



```
UserInfoSearch (JSON) 报文
   "UserInfoSearch": {
      "searchID": "",
                   //必填,搜索记录唯一标识,用来确认上层客户端是否为同一个(倘
若是同一个,设备记录内存,下次搜索加快速度),string 类型
      "responseStatusStrg": "OK", //必填,查询状态字符串描述:OK-查询结束,MORE-还
有数据等待查询,NO MATCH-没有匹配数据,string 类型
      "numOfMatches": 1, //必填,本次返回的记录条数,integer32 类型
      "totalMatches": 1, //必填,符合条件的记录总条数,integer32 类型
      "UserInfo":[ //可选, 人员信息
             "employeeNo": "", //必填, string, 工号(人员 ID)
             "name": "", //可选, string, 姓名
             "userType": "normal",
                             //必填, string, 人员类型, normal-普通人(主人),
visitor-来宾(访客),blackList-禁止名单人
             "closeDelayEnabled": true,  //可选, boolean,是否关门延迟, true-是,
false-否
                      //必填,有效期参数 (enable 不使能代表长期有效)
             "Valid" : {
                "enable": true, //必填, boolean, 使能有效期, false-不使能, true-
使能
                "beginTime": "", //必填,有效期起始时间(timeType 字段不存在
或为 local 时, beginTime 为设备本地时间,如: 2017-08-01T17:30:08; timeType 字段为 UTC
时, beginTime 为 UTC 时间,如: 2017-08-01T17:30:08+08:00)
                "endTime": "", //必填,有效期结束时间(timeType 字段不存在或
为 local 时, endTime 为设备本地时间,如: 2017-08-01T17:30:08; timeType 字段为 UTC 时,
endTime 为 UTC 时间,如: 2017-08-01T17:30:08+08:00)
                "timeType": "" //可选, string, 时间类型: local-设备本地时间,
UTC-UTC 时间
```



```
},
             "belongGroup": "1,3,5", //可选, string, 所属群组
             "password": "123456",
                               //可选, string, 密码
             "doorRight": "1,3", //可选, string, 门权限(代表对门 1、门 3 有权限)
(锁权限,此处为锁 ID,可填写多个,代表对锁 1、锁 3 有权限)
             "RightPlan": [ //可选,门权限计划(锁权限计划)
                    "doorNo": 1,
                               //可选, integer, 门编号(锁 ID)
                   "planTemplateNo": "1,3,5" //可选, string, 计划模板编号, 同
个门不同计划模板采用权限或的方式处理
             ],
             "maxOpenDoorTime": 0,  //可选, integer, 最大认证次数, 0 为无次数限
制
             "openDoorTime": 0,
                            //只读,可选, integer, 已认证次数
             "roomNumber": 123,
                             //可选,integer,房间号
             "floorNumber": 1,
                           //可选,integer,层号
             "doubleLockRight": true,  //可选, boolean, 反锁开门权限, true-有权限,
false-无权限
             "localUIRight": true,
                            //可选, boolean, 是否具有设备本地 UI 访问权限,
true-有权限, false-无权限
             "userVerifyMode": "card",
                                  //可选, string, 人员验证方式(人员验证方
式的优先级高于读卡器验证方式): cardAndPw-刷卡+密码, card-刷卡, cardOrPw-刷卡或密
码,
             //fp-指纹,fpAndPw-指纹+密码,fpOrCard-指纹或刷卡,fpAndCard-指纹+
刷卡,fpAndCardAndPw-指纹+刷卡+密码,faceOrFpOrCardOrPw-人脸或指纹或刷卡或密码,
             //faceAndFp-人脸+指纹,faceAndPw-人脸+密码,faceAndCard-人脸+刷卡,
face-人脸,employeeNoAndPw-工号+密码,fpOrPw-指纹或密码,employeeNoAndFp-工号+
指纹,
             //employeeNoAndFpAndPw-工号+指纹+密码,faceAndFpAndCard-人脸+指
纹+刷卡, faceAndPwAndFp-人脸+密码+指纹, employeeNoAndFace-工号+人脸,
faceOrfaceAndCard-人脸或人脸+刷卡,
             //fpOrface-指纹或人脸, cardOrfaceOrPw-刷卡或人脸或密码
             "gender": "male",
             /*可选, 人脸图片对应的人员性别: male-男, female-女, unknown-未知,
string 类型,*/
             "PersonInfoExtends": [{//可选
             "name":"", //可选,string,人员信息扩展名称
             "value":"" //可选,string,人员信息扩展内容
             }]
```



2.1.1.4 备注

}

1.UserInfoSearchCond 中 EployeeNoList 字段不存在或为空时,代表查询所有用户。

2.1.2 人员设置(新增、修改)

2.1.2.1 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_JSON_CONFIG

lpInBuffer

[in] PUT /ISAPI/AccessControl/UserInfo/SetUp?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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 UserInfo(JSON)报文

dwInBuffSize

[in] 一个 UserInfo (JSON) 报文大小

lpOutBuff

[out] 指向一个 ResponseStatus (JSON) 报文

dwOutBuffSize



[in] 一个 ResponseStatus(JSON)报文大小

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

2.1.2.2 命令码

#define NET_DVR_JSON_CONFIG 2550 //JSON 透传数据

2.1.2.3 JSON 报文

URL: PUT/ISAPI/AccessControl/UserInfo/SetUp?format=json

UserInfo (JSON) 报文



```
//必填, string, 人员类型, normal-普通人(主人), visitor-
      "userType": "normal",
来宾(访客), blackList-禁止名单人
      "closeDelayEnabled": true,
                          //可选, boolean, 是否关门延迟, true-是, false-否
                //必填,有效期参数(enable 不使能代表长期有效)(有效时间跨度为
1970年1月1日0点0分0秒~2037年12月31日23点59分59秒)
          "enable": true.
                     //必填, boolean, 使能有效期, false-不使能, true-使能
         "beginTime": "",
                      //必填,有效期起始时间(timeType 字段不存在或为 local
时, beginTime 为设备本地时间,如: 2017-08-01T17:30:08; timeType 字段为 UTC 时, beginTime
为UTC时间,如: 2017-08-01T17:30:08+08:00)
                     //必填,有效期结束时间(timeType 字段不存在或为 local 时,
          "endTime": "",
endTime 为设备本地时间,如: 2017-08-01T17:30:08; timeType 字段为 UTC 时, endTime 为
UTC 时间,如:2017-08-01T17:30:08+08:00)
          "timeType": ""
                     //可选, string, 时间类型: local-设备本地时间, UTC-UTC
时间
      "belongGroup": "1,3,5", //可选, string, 所属群组
      "password": "123456", //可选, string, 密码
      "doorRight": "1,3", //可选, string, 门权限(代表对门1、门3有权限)(锁权限,
此处为锁 ID, 可填写多个, 代表对锁 1、锁 3 有权限)
      "RightPlan": [ //可选,门权限计划(锁权限计划)
             "doorNo": 1, //可选, integer, 门编号(锁 ID)
             "planTemplateNo": "1,3,5" //可选, string, 计划模板编号, 同个门不同
计划模板采用权限或的方式处理
      ],
      "maxOpenDoorTime": 0, //可选, integer, 最大认证次数, 0 为无次数限制
      "openDoorTime": 0, //只读,可选, integer, 已认证次数
                      //可选, integer, 房间号
      "roomNumber": 123,
      "floorNumber": 1, //可选, integer, 层号
      "doubleLockRight": true,  //可选,boolean,反锁开门权限,true-有权限,false-
无权限
      "localUIRight": true,  //可选,boolean,是否具有设备本地 UI 访问权限,true-有
权限, false-无权限
      "userVerifyMode": "card", //可选, string, 人员验证方式(人员验证方式的优先
级高于读卡器验证方式): cardAndPw-刷卡+密码, card-刷卡, cardOrPw-刷卡或密码, fp-指
纹, fpAndPw-指纹+密码,
      //fpOrCard-指纹或刷卡,fpAndCard-指纹+刷卡,fpAndCardAndPw-指纹+刷卡+密码,
faceOrFpOrCardOrPw-人脸或指纹或刷卡或密码,faceAndFp-人脸+指纹,faceAndPw-人脸+
密码, faceAndCard-人脸+刷卡,
      //face-人脸,employeeNoAndPw-工号+密码,fpOrPw-指纹或密码,
employeeNoAndFp-工号+指纹,employeeNoAndFpAndPw-工号+指纹+密码,
faceAndFpAndCard-人脸+指纹+刷卡,
```



```
//faceAndPwAndFp-人脸+密码+指纹,employeeNoAndFace-工号+人脸,
faceOrfaceAndCard-人脸或人脸+刷卡,fpOrface-指纹或人脸,cardOrfaceOrPw-刷卡或人脸或
密码,cardOrFace-刷卡或人脸,
      //cardOrFaceOrFp-刷卡或人脸或指纹, faceOrPw-人脸或密
码,employeeNoAndFaceAndPw-工号+人脸+密码,faceOrfaceAndCard-人脸或人脸+刷卡,
fpOrface-指纹或人脸, cardOrfaceOrPw-刷卡或人脸或密码
      "checkUser": true,
                     //可选, boolean, 设备是否进行人员重复添加校验, false-不
校验, true-校验(如果不配置该字段,则设备默认进行人员重复校验)(如果确认设备端不
存在任何人员信息,可将其置为 false,则设备不进行重复校验,这样会加快下发速度:如果
不确认,则不建议配置该字段)
      "numOfFace":0. //可选, 只读, 关联人脸数量, 不返回不支持
      "numOfFP":0, //可选, 只读, 关联指纹数量, 不返回不支持
      "numOfCard":0,//可选,只读,关联卡数量,不返回不支持
      "gender": "male",/*可选, 人脸图片对应的人员性别: male-男, female-女, unknown-未
知, string 类型,*/
      "PersonInfoExtends": [{//可选
         "name":"", //可选,string,人员信息扩展名称
         "value":"" //可选,string,人员信息扩展内容
      }],/*可选,自定义字段*/
      "operateType": "byTerminal",
      /*可选,操作类型,string:"byTerminal:按终端操作,byOrg:按组织操作,byTerminalOrg:
按终端组织操作*/
      "terminalNoList": [1, 2, 3, 4],
      /*可选,array,type 为 byTerminal,byTerminalOrg 时必填,终端 ID 列表*/
      "orgNoList": [1, 2, 3, 4],
      /*可选,array,type 为 byOrg,byTerminalOrg 时必填,组织 ID 列表*/
      "dynamicCode": "123456", //可选, wo, string, 动态权限码
      "callNumbers": ["","",""], //可选, string, 呼叫号码列表, 默认规则 X-X-X-X,
如 1-1-1-401, roomNumber 字段扩展, 支持列表时, 使用列表配置相关信息
      "floorNumbers": [1,2] //可选, integer, 层号列表, floorNumber 扩展, 支持列
表时,可使用该字段配置层号
```

2.1.2.4 备注

- 1.设备根据人员 ID (employeeNo) 判定人员不存在时,则添加该人员信息;
- 2.设备根据人员 ID (employeeNo) 判定人员存在时,则修改该人员信息;
- 3.当要删除某个人员时,deleteUser 为 true 即可,无论该人员是否存在,都会返回删除成功(删除只会删除该人员信息,不会删除其关联的卡、指纹、人脸信息)。

2.1.3 人员删除



2.1.3.1 接口函数

1.开始删除

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

lUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

[in] PUT /ISAPI/AccessControl/UserInfoDetail/Delete?format=json

lpInputParam->lpInBuffer

[in] 指向一个 UserInfoDetail (JSON) 报文

lpOutputParam->lpOutBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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.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_JSON_CONFIG

lpInBuffer

[in] GET /ISAPI/AccessControl/UserInfoDetail/DeleteProcess?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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

3.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 UserInfoDetailDeleteProcess(JSON)报文/ResponseStatus(JSON)报文 dwOutBuffSize

[in] 一个 UserInfoDetailDeleteProcess(JSON)报文/ResponseStatus(JSON)报文大小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

4.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

2.1.3.2 命令码

#define NET_DVR_JSON_CONFIG 2550 //JSON 透传数据

2.1.3.3 JSON 报文

URL: PUT /ISAPI/AccessControl/UserInfoDetail/Delete?format=json

URL: GET/ISAPI/AccessControl/UserInfoDetail/DeleteProcess?format=json

UserInfoDetail (JSON) 报文



```
{
    "UserInfoDetail": {
        "mode": "", //必填, string, 删除模式 (all-删除所有, byEmployeeNo-按人员 ID 删除)
    "EmployeeNoList": [ //可选, 人员 ID 列表
        {
             "employeeNo": "" //可选, string, 人员 ID (当 mode 为 byEmployeeNo 时, 该字段有效)
        }
        ]
    }
}
```

UserInfoDetailDeleteProcess (JSON) 报文

2.1.3.4 备注

1.删除人员信息的同时,删除其关联的卡信息、指纹信息、人脸信息(该接口调用返回后,不代表实际删除完成,要调用 GET /ISAPI/AccessControl/UserInfoDetail/DeleteProcess?format=json 来获取实际删除进度,当进度值 progressValue 为 success 时,代表已完成);

2.UserInfoDetail 中 EployeeNoList 字段不存在或为空时,代表删除所有人员信息及权限。

2.1.4 查询所有人员数量

2.1.4.1 接口函数

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

lUserID

```
[in] NET_DVR_Login_V40 等登录接口的返回值
```

lpInputParam->lpRequestUrl

[in] GET /ISAPI/AccessControl/UserInfo/Count?format=json

lpInputParam->lpInBuffer

[in] NULL

lpOutputParam->lpOutBuffer



[out] 指向一个 UserInfoCount (JSON) 报文 lpOutputParam->lpStatusBuffer [out] 指向一个 ResponseStatus (JSON) 报文

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

2.1.4.2 命令码

2.1.4.3 JSON 报文

URL: GET/ISAPI/AccessControl/UserInfo/Count?format=json

UserInfoCount (JSON) 报文

```
{
    "UserInfoCount" : {
        "userNumber": 100
    }
}
```

2.1.4.4 备注

2.2 卡管理

2.2.1 卡获取 (查询)

2.2.1.1 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_JSON_CONFIG

lpInBuffer

[in] POST /ISAPI/AccessControl/CardInfo/Search?format=json



dwInBufferSize:

[in] sizeof(lpInBuffer)

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 CardInfoSearchCond (JSON) 报文

dwInBuffSize

[in] 一个 CardInfoSearchCond (JSON) 报文大小

lpOutBuff

[out] 指向一个 CardInfoSearch(JSON)报文/ResponseStatus(JSON)报文 dwOutBuffSize

[in] 一个 CardInfoSearch(JSON)报文/ResponseStatus(JSON)报文大小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

2.2.1.2 命令码

#define NET_DVR_JSON_CONFIG 2550 //JSON 透传数据

2.2.1.3 JSON 报文

URL: POST /ISAPI/AccessControl/CardInfo/Search?format=json

CardInfoSearchCond (JSON) 报文

```
"CardInfoSearchCond": {
     "searchID": "".
                //必填, string, 搜索记录唯一标识, 用来确认上层客户端是否为
同一个(倘若是同一个,设备记录内存,下次搜索加快速度)
     "searchResultPosition": 0,
                       //必填, integer, 查询结果在结果列表中的起始位置。
当记录条数很多时,一次查询不能获取所有的记录,下一次查询时指定位置可以查询后面的
记录(若设备支持的最大 totalMatches 为 M 个,但是当前设备已存储的 totalMatches 为 N 个
(N<=M),则该字段的合法范围为 0~N-1)
     "maxResults": 30,
                   //必填, integer, 本次协议调用可获取的最大记录数(如
maxResults 值大于设备能力集返回的范围,则设备按照能力集最大值返回,设备不进行报错)
     "EmployeeNoList": [ //可选, 人员 ID 列表
           "employeeNo": "" //可选, string, 人员 ID
     ],
     "CardNoList": [ //可选,卡号列表(与人员 ID 列表互斥,两者选一)
           "cardNo": "1234567890"
                           //可选,string,卡号
  }
```

CardInfoSearch (JSON) 报文

```
{
    "CardInfoSearch" : {
```



```
//必填,搜索记录唯一标识, string, 用来确认上层客户端是否为同
      "searchID": "",
一个(倘若是同一个,设备记录内存,下次搜索加快速度)
      "responseStatusStrg": "OK", //必填, string, 查询状态字符串描述: OK-查询结束,
MORE-还有数据等待查询,NO MATCH-没有匹配数据
                      //必填, integer, 本次返回的记录条数
      "numOfMatches": 1,
                    //必填, integer, 符合条件的记录总条数
      "totalMatches": 1.
      "CardInfo":[//可选,人员信息
             "employeeNo": "", //必填, string, 人员 ID
             "cardNo": "", //必填, string, 卡号
             "cardType": "normalCard", //必填, string, 卡类型, normalCard-普通卡,
patrolCard-巡更卡,hijackCard-胁迫卡,superCard-超级卡,dismissingCard-解除卡
      ]
   }
```

2.2.1.4 备注

1.CardInfoSearchCond 中 EmployeeNoList 和 CardNoList 字段不存在或为空时, 代表查询所有卡。

2.2.2 卡设置 (新增、修改)

2.2.2.1 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_JSON_CONFIG

lpInBuffer

[in] PUT /ISAPI/AccessControl/CardInfo/SetUp?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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 CardInfo (JSON) 报文

dwInBuffSize

[in] 一个 CardInfo (JSON) 报文大小

lpOutBuff

[out] 指向一个 ResponseStatus (JSON) 报文

dwOutBuffSize

[in] 一个 ResponseStatus (JSON) 报文大小

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

2.2.2.2 命令码

#define NET DVR JSON CONFIG 2550 //JSON 透传数据

2.2.2.3 JSON 报文

URL: PUT/ISAPI/AccessControl/CardInfo/SetUp?format=json

CardInfo (JSON) 报文

2.2.2.4 备注

- 1.设备根据卡号判定卡不存在时,则添加该卡信息;
- 2.设备根据卡号判定卡存在时,则修改该卡信息;
- 3.当要删除某个人员的某张卡时,要添加 employeeNo、cardNo 字段和 deleteCard 为 true。无论该人员下该卡是否存在,都会返回删除成功(删除只会删除卡信息,不会删除其关联的人员信息);
- 4.当要删除某个人员的所有卡时,要添加 employeeNo 字段和 deleteCard 为 true。无论该人员是否存在(或该人员下是否有卡),都会返回删除成功(删除只会删除卡信息,不会删除其关联的人员信息)。

2.2.3 卡删除

2.2.3.1 接口函数

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

lUserID



[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

[in] PUT /ISAPI/AccessControl/CardInfo/Delete?format=json

lpInputParam->lpInBuffer

[in] 指向一个 CardInfoDelCond (JSON) 报文

lpOutputParam->lpOutBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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

2.2.3.2 命令码

2.2.3.3 JSON 报文

URL: PUT/ISAPI/AccessControl/CardInfo/Delete?format=json

CardInfoDelCond (JSON) 报文

2.2.3.4 备注

1.CardInfoDelCond 中 EmployeeNoList 和 CardNoList 字段不存在或为空时,代表删除所有卡; 2.删除所有人员的卡时,超时时间建议设置为 60s。

2.2.4 查询所有卡数量



2.2.4.1 接口函数

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

lUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

[in] GET /ISAPI/AccessControl/CardInfo/Count?format=json

lpInputParam->lpInBuffer

[in] NULL

lpOutputParam->lpOutBuffer

[out] 指向一个 CardInfoCount (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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

2.2.4.2 命令码

2.2.4.3 JSON 报文

URL: GET/ISAPI/AccessControl/CardInfo/Count?format=json

CardInfoCount (JSON) 报文

```
{
    "CardInfoCount" : {
        "cardNumber": 100
    }
}
```

2.2.4.4 备注

2.2.5 查询指定工号所有卡数量

2.2.5.1 接口函数



NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

IUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

[in] GET /ISAPI/AccessControl/CardInfo/Count?format=json&employeeNo=<ID>

lpInputParam->lpInBuffer

[in] NULL

lpOutputParam->lpOutBuffer

[out] 指向一个 CardInfoCount (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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

2.2.5.2 命令码

2.2.5.3 JSON 报文

URL: GET/ISAPI/AccessControl/CardInfo/Count?format=json&employeeNo=<ID>

CardInfoCount (JSON) 报文

```
{
    "CardInfoCount" : {
        "cardNumber": 100
    }
}
```

2.2.5.4 备注

1.URL 中的 <ID> 代表实际的人员ID (如:/ISAPI/AccessControl/CardInfo/Count?format=json&employeeNo=1 代表查询人员ID为1所拥有的卡的数量)

2.3指纹管理

2.3.1 指纹获取(查询)



2.3.1.1 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_JSON_CONFIG

lpInBuffer

[in] POST /ISAPI/AccessControl/FingerPrintUpload?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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 FingerPrintCond (JSON) 报文

dwInBuffSize

[in] 一个 FingerPrintCond (JSON) 报文大小

lpOutBuff

[out] 指向一个 FingerPrintInfo(JSON)报文/ResponseStatus(JSON)报文

dwOutBuffSize

[in] 一个 FingerPrintInfo(JSON)报文/ResponseStatus(JSON)报文大小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

2.3.1.2 命令码

#define NET_DVR_JSON_CONFIG 2550 //JSON 透传数据

2.3.1.3 JSON 报文

URL: POST /ISAPI/AccessControl/FingerPrintUpload?format=json

FingerPrintCond (JSON) 报文

```
{
    "FingerPrintCond": {
        "searchID": "", //必填,string,搜索记录唯一标识,用来确认上层客户端是否为
同一个(倘若是同一个,设备记录内存,下次搜索加快速度)
        "employeeNo": "", //必填,string,指纹关联的人员 ID
        "cardReaderNo": , //可选,integer,读卡器编号
        "fingerPrintID": , //可选,integer,手指编号: 1-10
      }
}
```

FingerPrintInfo (JSON) 报文



```
"FingerPrintInfo": {
      "searchID": "",
                  //必填, string, 搜索记录唯一标识,用来确认上层客户端是否为同
 -个(倘若是同一个,设备记录内存,下次搜索加快速度)
                //必填, string, 状态(OK-存在指纹, NoFP-不存在指纹)
      "FingerPrintList": [
             "cardReaderNo":, //必填, integer, 读卡器编号
             "fingerPrintID":,
                          //必填, integer, 手指编号: 1-10
             "fingerType": "",
                          //必填, string, 指纹类型: normalFP-普通指纹,
hijackFP-胁迫指纹,patrolFP-巡更指纹,superFP-超级指纹,dismissingFP-解除指纹
             "fingerData": "",  //必填, string, 指纹数据(需要 Base64 编码)
                           //可选, array, 是否有首次认证功能([1,3,5]代表指
             "leaderFP": [1,3,5]
纹对于门1、门3、门5有首次认证功能)
      ]
   }
```

2.3.1.4 备注

1.获取单个指纹: FingerPrintCond 中 searchID、employeeNo、cardReaderNo、fingerPrintID 均填写。若存在符合条件的指纹,则 FingerPrintInfo 中 status 返回 OK,并且 FingerPrintList 列表中返回相应的指纹信息;若不存在符合条件的指纹,则 FingerPrintInfo 中 status 返回 NoFP,并且 FingerPrintList 列表中返回内容为空;

2.获取某一人员 ID 的所有指纹: FingerPrintCond 中 searchID、employeeNo 填写, cardReaderNo、fingerPrintID 不需要填写。若存在符合条件的指纹,则 FingerPrintInfo 中 status 返回 OK,并且 FingerPrintList 列 表 中 返 回 相 应 的 指 纹 信 息 , 并 再 次 循 环 调 用 POST /ISAPI/AccessControl/FingerPrintUpload?format=json(FingerPrintCond 内的条件保持不变),多次获取符合条件的指纹信息,直到 FingerPrintInfo 中 status 返回 NoFP,说明已获取完所有符合条件的指纹;若不存在符合条件的指纹,则 FingerPrintInfo 中 status 返回 NoFP,并且 FingerPrintList 列表中返回内容为空,不需要再循环调用该接口了。

2.3.2 指纹设置(新增、修改)

2.3.2.1 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_JSON_CONFIG

lpInBuffer

[in] POST /ISAPI/AccessControl/FingerPrint/SetUp?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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 FingerPrintCfg(JSON)报文

dwInBuffSize

[in] 一个 FingerPrintCfg (JSON) 报文大小

lpOutBuff

[out] 指向一个 FingerPrintStatus(JSON)报文/ResponseStatus(JSON)报文 dwOutBuffSize

[in] 一个 FingerPrintStatus(JSON)报文/ResponseStatus(JSON)报文大小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

2.3.2.2 命令码

#define NET_DVR_JSON_CONFIG 2550 //JSON 透传数据

2.3.2.3 JSON 报文

URL: POST /ISAPI/AccessControl/FingerPrint/SetUp?format=jso

FingerPrintCfg (JSON) 报文

```
{
    "FingerPrintCfg": {
        "employeeNo": "", //必填, string, 指纹关联的人员 ID
        "enableCardReader": [1,3,5], //必填, array, 需要下发指纹的读卡器([1,3,5]代表需要下发给读卡器 1、读卡器 3、读卡器 5)
        "fingerPrintID": , //必填, integer, 手指编号: 1-10
        "deleteFingerPrint": true, //可选, boolean, 是否删除该指纹, true-是(只有删除该指纹时, 才填写该字段;新增或修改指纹时,不填写该字段)
        "fingerType": "", //必填, string, 指纹类型: normalFP-普通指纹, hijackFP-胁迫指纹, patrolFP-巡更指纹, superFP-超级指纹, dismissingFP-解除指纹
        "fingerData": "", //可选, string, 指纹数据(需要 Base64 编码)
    }
}
```

FingerPrintStatus (JSON) 报文

```
{
    "FingerPrintStatus": {
        "status": "", //可选, string, 状态: success-成功, failed-失败(对于实际下发指纹数据到读卡器时,不会返回该字段。如果仅是修改指纹参数(不涉及到指纹数据下发到读卡器)或删除指纹,才会返回该字段)
        "StatusList": [ //可选,状态列表(对于实际下发指纹数据到读卡器时,才会返
```



2.3.2.4 备注

- 1.如果 fingerData 字段没有下发:则代表修改相关参数信息,并不进行指纹数据下发;
- 2.如果 fingerData 字段有下发:对于某个读卡器上没有该指纹数据,则新创建;对于某个读卡器上有该指纹数据,则覆盖;
- 3.删除指纹方式:(1)employeeNo、enableCardReader、fingerPrintID 和 deleteFingerPrint,代表删除某一工号下指定读卡器中某一个手指编号的指纹;(2)employeeNo、fingerPrintID 和 deleteFingerPrint,代表删除某一工号下所有读卡器中某一个手指编号的指纹;(3)employeeNo、enableCardReader 和 deleteFingerPrint,代表删除某一工号下指定读卡器中所有手指编号的指纹;(4)employeeNo 和 deleteFingerPrint,代表删除某一工号下所有读卡器中所有手指编号的指纹。并且,无论该指纹数据是否存在,都会返回删除成功。

2.3.3 指纹删除

2.3.3.1 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_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

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



2.3.3.2 命令码

#define NET_DVR_DEL_FINGERPRINT 2565 //删除指纹

2.3.3.3 宏定义及结构体

ዹ 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_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;
                       //删除方式, 0-按卡号(人员 ID)方式删除, 1-按读卡器删除
   BYTE byMode;
   BYTE byRes1[3];
                      //保留
   NET_DVR_DEL_FINGER_PRINT_MODE_V50 struProcessMode; //处理方式
                        //保留
   BYTE byRes[64];
}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
                                                   //按卡号(人员 ID)的方式
   NET_DVR_FINGER_PRINT_BYCARD_V50
                                        struByCard;
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
   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]; //人员 ID
```

海康威视版权所有

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]; //人员 ID
                         //保留
   BYTE byRes[516];
}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
                        //保留
   BYTE byRes[63];
}NET_DVR_FINGER_PRINT_INFO_STATUS_V50,
*LPNET_DVR_FINGER_PRINT_INFO_STATUS_V50;
2.3.3.4 备注
```

2.3.4 指纹采集

2.3.4.1 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_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 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_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

TELESTIC OCCURRENCE OF THE CONTROL O			
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

2.3.4.2 命令码

#define NET_DVR_CAPTURE_FINGERPRINT_INFO 2504 //采集指纹



2.3.4.3 宏定义及结构体

ዹ 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

```
ዹ 结构体:
```

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

2.4 人脸管理

2.4.1 人脸获取(查询)

2.4.1.1 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_FACE_DATA_SEARCH



lpInBuffer

[in] POST /ISAPI/Intelligent/FDLib/FDSearch?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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 Inbound Data (JSON) 报文

dwInBuffSize

[in] 一个 Inbound Data (JSON) 报文大小

lpOutBuff

[out] 指向一个NET_DVR_JSON_DATA_CFG 结构体

dwOutBuffSize

[in] 一个 NET_DVR_JSON_DATA_CFG 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

2.4.1.2 命令码

#define NET_DVR_FACE_DATA_SEARCH 2552 //查询人脸库中的人脸数据

2.4.1.3 宏定义

```
typedef struct tagNET_DVR_JSON_DATA_CFG {
    DWORD dwSize; //结构体大小
    void *lpJsonData; //JSON 报文(Return(JSON)报文或 ResponseStatus(JSON)报文)
    DWORD dwJsonDataSize; //JSON 报文大小
    void *lpPicData; //图片内容(当 JSON 报文为当 ResponseStatus(JSON)报文时,该字段无意义; 当 Inbound Data(JSON)报文中没有 faceURL 时,该字段需要带上二进制图片内容)
    DWORD dwPicDataSize; //图片内容大小(大小限制为 200k 以内)
    BYTE byRes[256]; //保留
```

2.4.1.4 JSON 报文

URL: POST /ISAPI/Intelligent/FDLib/FDSearch?format=json

}NET_DVR_JSON_DATA_CFG,*LPNET_DVR_JSON_DATA_CFG;

Inbound Data (JSON) 报文(门禁设备在返回报文的同时,也返回图片,故 searchResultPosition 只能配置为 0, maxResults 只能配置为 1)



Return(JSON)报文(门禁设备在返回报文的同时,也返回图片,故 MatchList 列表只能返回一个人脸记录信息)

```
"requestURL": "",
                   //可选,string,请求 URL
                //必填, integer, 状态码
   "statusCode":,
   "statusString": "",
                 //必填, string, 状态描述
   "subStatusCode": "",
                   //必填, string, 子状态码
               //可选, integer, 当 statusCode 不为 1 时, 必填。错误码与 subStatusCode
   "errorCode": 1,
对应
                  //可选, string, 当 statusCode 不为 1 时,必填。错误详细信息,能
   "errorMsg": "ok",
具体到某一个参数的错误
   "responseStatusStrg": "OK",
                         //可选, string, 查询状态字符串描述: OK-查询结束,
MORE-还有数据等待查询,NO MATCH-没有匹配数据,最大长度为 32, {dep if errcode == 1
&& errMsg == ok}
   "numOfMatches": 1,
                    //可选,integer,本次返回的记录条数,{dep if errcode == 1 &&
errMsg == ok
   "totalMatches": 1,  //可选,integer,符合条件的记录总条数,{dep if errcode == 1 &&
errMsg == ok
   "MatchList": [{ //可选, array, 查询到的匹配数据信息
       "FPID": "",
                 //可选, string, 人脸记录 ID (与门禁人员 ID 字段一致), 最大长度
为 63 字节
   }]
```

2.4.1.5 备注

2.4.2 人脸设置(新增、修改)

2.4.2.1 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_FACE_DATA_SEARCH



lpInBuffer

[in] PUT /ISAPI/Intelligent/FDLib/FDSetUp?format=json

dwInBufferSize:

[in] sizeof(lpInBuffer)

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_JSON_DATA_CFG 结构体 dwInBuffSize

[in] 一个 NET_DVR_JSON_DATA_CFG 结构体大小

lpOutBuff

[out] 指向一个 ResponseStatus (JSON) 报文

dwOutBuffSize

[in] 一个 ResponseStatus(JSON)报文大小

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

2.4.2.2 命令码

#define NET_DVR_FACE_DATA_RECORD 2551 //添加人脸数据到人脸库

2.4.2.3 宏定义

```
typedef struct tagNET_DVR_JSON_DATA_CFG
{
```

DWORD dwSize; //结构体大小

void *lpJsonData; //JSON 报文(Return(JSON)报文或 ResponseStatus(JSON)报文)

DWORD dwJsonDataSize; //JSON 报文大小

void *lpPicData; //图片内容(当 JSON 报文为当 ResponseStatus(JSON)报文时,该字段无

意义; 当 Inbound Data (JSON) 报文中没有 faceURL 时,该字段需要带上二进制图片内容)

DWORD dwPicDataSize; //图片内容大小(大小限制为 200k 以内)

BYTE byRes[256]; //保留

}NET_DVR_JSON_DATA_CFG,*LPNET_DVR_JSON_DATA_CFG;

2.4.2.4 JSON 报文

URL: PUT /ISAPI/Intelligent/FDLib/FDSetUp?format=json

Inbound Data (JSON) 报文

```
【 "faceLibType": "blackFD", //必填, string, 人脸库类型: blackFD-名单库 "FDID": "1", //必填, string, 人脸库 ID 最大长度为 63 字节(门禁设备默认为 1) "FPID": "", //可选, string, 人脸记录 ID, 如果外部传入, 最长 63 字节, 字母数字组合, 需要保证唯一性。如果外部不传则由设备自动生成(与门禁人员 ID 字段一致) "deleteFP": true, //可选, boolean, 是否删除该人脸, true-是(只有删除该人脸时, 才填写该字段; 新增或修改人脸时, 不填写该字段)
```

2.4.2.5 备注



设置人脸数据(包括人脸图片、人员信息等)到人脸库:

- 1.如果该人员 ID 中人脸图片不存在时,则新建人脸数据;
- 2.如果该人员 ID 中人脸图片存在时,则覆盖人脸数据;
- 3.删除人脸时,需要填写 faceLibType、FDID、FPID 和 deleteFP 字段,人脸数据不需要下发(并且无论是否存在该人脸,都会删除成功)。

2.4.3 人脸删除

2.4.3.1 接口函数

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

IUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

[in] PUT

/ISAPI/Intelligent/FDLib/FDSearch/Delete?format=json&FDID=<ID>&faceLibType=blackFDlpInputParam->lpInBuffer

[in] 指向一个 Inbound Data (JSON) 报文

lpOutputParam->lpOutBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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

2.4.3.2 命令码

2.4.3.3 JSON 报文

URL: PUT /ISAPI/Intelligent/FDLib/FDSearch/Delete?format=json&FDID=<ID>&faceLibType=blackFD

Inbound Data (JSON) 报文



2.4.3.4 备注

}

1.URL 中的 FDID=1。

2.4.4 查询所有人脸库人脸记录总数

2.4.4.1 接口函数

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

IUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

[in] GET /ISAPI/Intelligent/FDLib/Count?format=json

lpInputParam->lpInBuffer

[in] NULL

lpOutputParam->lpOutBuffer

[out] 指向一个 Return (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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

2.4.4.2 命令码

2.4.4.3 JSON 报文

 $URL: \ GET\ / ISAPI/Intelligent/FDLib/Count? format=json$

Return (JSON) 报文

```
{
    "requestURL": "", //可选,string,请求 URL
    "statusCode": , //必填,integer,状态码
    "statusString": "", //必填,string,状态描述
    "subStatusCode": "", //必填,string,子状态码
    "errorCode": 1, //可选,integer,当 statusCode 不为 1 时,必填。错误码,与 subStatusCode
```



```
对应
"errorMsg": "ok", //可选, string, 当 statusCode 不为 1 时,必填。错误详细信息,能具体到某一个参数的错误
"FDRecordDataInfo": [{ //可选,人脸库记录信息,{dep if errcode==1&&errMsg==ok}"FDID": "", //可选, string,人脸库 ID,最大长度为 63 字节(门禁设备默认为 1)"faceLibType": "blackFD", //可选, string,人脸库类型: blackFD-名单库,最大长度为 32
"recordDataNumber": 123 //可选, integer,记录数据条数
}]
}
```

2.4.4.4 备注

2.4.5 查询指定人脸库人脸记录总数

2.4.5.1 接口函数

NET_DVR_API BOOL __stdcall NET_DVR_STDXMLConfig(LONG lUserID, NET_DVR_XML_CONFIG_INPUT* lpInputParam, NET_DVR_XML_CONFIG_OUTPUT* lpOutputParam);

Parameters

lUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lpInputParam->lpRequestUrl

 $[in] \ GET\ /ISAPI/Intelligent/FDLib/Count?format=json\&FDID=<ID>\&faceLibType=blackFDlpInputParam->lpInBuffer$

[in] NULL

lpOutputParam->lpOutBuffer

[out] 指向一个 Return (JSON) 报文

lpOutputParam->lpStatusBuffer

[out] 指向一个 ResponseStatus (JSON) 报文

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

2.4.5.2 命令码

2.4.5.3 JSON 报文



URL: GET/ISAPI/Intelligent/FDLib/Count?format=json&FDID=<ID>&faceLibType=blackFD

Return (JSON) 报文

```
"requestURL": "",
                  //可选,string,请求 URL
               //必填,integer,状态码
   "statusCode":,
   "statusString": "",
                //必填, string, 状态描述
   "subStatusCode": "", //必填, string, 子状态码
   "errorCode": 1, //可选, integer, 当 statusCode 不为 1 时, 必填。错误码, 与 subStatusCode
对应
                 //可选, string, 当 statusCode 不为 1 时,必填。错误详细信息,能
   "errorMsg": "ok",
具体到某一个参数的错误
              //可选, string, 人脸库 ID, 最大长度为 63 字节(门禁设备默认为 1)
   "FDID": "",
   "faceLibType": "blackFD",  //可选,string,人脸库类型: blackFD-名单库,最大长度
为32
   "recordDataNumber": 123 //可选, integer, 记录数据条数
```

2.4.5.4 备注

1.URL 中的 FDID=1。

2.4.6 人脸采集

2.4.6.1 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_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

2.4.6.2 命令码

#define NET_DVR_CAPTURE_FACE_INFO 2510 //采集人脸

2.4.6.3 宏定义及结构体



ዹ 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)
                          //人脸图片数据大小,等于0时,代表无人脸图片数据
   DWORD dwFacePicSize;
                       //人脸图片数据缓存
   char* pFacePicBuffer;
                         //人脸质量,范围 1-100
   BYTE byFaceQuality1;
   BYTE byFaceQuality2;
                         //人脸质量,范围 1-100
                        //采集进度,目前只有两种进度值:0-未采集到人脸,100-采集
   BYTE byCaptureProgress;
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
```

//红外人脸图片数据缓存 char* pInfraredFacePicBuffer;

byRes[116]; BYTE

}NET_DVR_CAPTURE_FACE_CFG, *LPNET_DVR_CAPTURE_FACE_CFG;

2.5 远程控门

2.5.1 接口函数

远程控门

NET_DVR_API BOOL __stdcall NET_DVR_ControlGateway(LONG lUserID, LONG lGatewayIndex, DWORD dwStaic);

Parameters

IUserID

[in] NET_DVR_Login_V40 等登录接口的返回值

lGatewayIndex

[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

2.6设备事件主动上传(布防)

2.6.1 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 lUserID,



LPNET_DVR_SETUPALARM_PARAM lpSetupParam);

Parameters

1UserID

[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

2.6.2 命令码

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

2.6.3 宏定义及结构体

ዹ 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

ዹ 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, 3-禁止名单卡, 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 byAllowListNo; //允许名单单号,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(当接的设备为人员通道设备时,门 1 为进方向,门 2 为出方向)

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 byMask; //是否带口罩: 0-保留, 1-未知, 2-不戴口罩, 3-戴口罩

DWORD dwSerialNo; //事件流水号,为0无效

BYTE byChannelControllerID; //通道控制器 ID, 为 0 无效, 1-主通道控制器, 2-从通道控制器

BYTE byChannelControllerLampID; //通道控制器灯板 ID, 为 0 无效(有效范围 1-255)

BYTE byChannelControllerIRAdaptorID; //通道控制器红外转接板 ID, 为 0 无效(有效范围 1-255)

BYTE byChannelControllerIREmitterID; //通道控制器红外对射 ID, 为 0 无效(有效范围 1-255)

BYTE byHelmet;//可选,是否戴安全帽: 0-保留,1-未知,2-不戴安全,3-戴安全帽

BYTE byRes[3];

}NET DVR ACS EVENT INFO, *LPNET DVR ACS EVENT INFO;

typedef struct tagNET_DVR_ACS_EVENT_INFO_EXTEND

DWORD dwFrontSerialNo; //事件流水号,为 0 无效(若该字段为 0,平台根据 dwSerialNo 判断是否丢失事件;若该字段不为 0,平台根据该字段和 dwSerialNo 字段共同判断是否丢失事件)(主要用于解决报警订阅后导致 dwSerialNo 不连续的情况)

BYTE byUserType; //人员类型: 0-无效, 1-普通人(主人), 2-来宾(访客), 3-禁止名单人, 4-管理员

BYTE byCurrentVerifyMode; //读卡器当前验证方式: 0-无效, 1-休眠, 2-刷卡+密码, 3-刷卡, 4-刷卡或密码, 5-指纹, 6-指纹+密码, 7-指纹或刷卡, 8-指纹+刷卡, 9-指纹+刷卡+密码, 10-人脸或指纹或刷卡或密码, 11-人脸+指纹, 12-人脸+密码, 13-人脸+刷卡, 14-人脸, 15-工号+密码, 16-指纹或密码, 17-工号+指纹, 18-工号+指纹+密码, 19-人脸+指纹+刷卡, 20-人脸+密码+指纹, 21-工号



+人脸, 22-人脸或人脸+刷卡, 23-指纹或人脸, 24-刷卡或人脸或密码, 25-刷卡或人脸, 26-刷卡或 人脸或指纹,27-刷卡或指纹或密码 BYTE byCurrentEvent; //是否为实时事件: 0-无效, 1-是(实时事件), 2-否(离线事件) BYTE byPurePwdVerifyEnable; //设备是否支持纯密码认证, 0-不支持, 1-支持 BYTE byEmployeeNo[NET_SDK_EMPLOYEE_NO_LEN]; //工号(人员ID)(对于设备来说, 如果使用了工号(人员 ID)字段, byEmployeeNo 一定要传递, 如果 byEmployeeNo 可转换为 dwEmployeeNo,那么该字段也要传递;对于上层平台或客户端来说,优先解析 byEmployeeNo 字段, 如该字段为空,再考虑解析 dwEmployeeNo 字段) BYTE byAttendanceStatus; //考勤状态: 0-未定义,1-上班, 2-下班, 3-开始休息, 4-结束休息, 5-开始加班, 6-结束加班 BYTE byStatusValue; //考勤状态值 BYTE byRes2[2]; BYTE byUUID[NET SDK UUID LEN/*36*/]; //UUID(该字段仅在对接萤石平台过程中才会 使用) BYTE byDeviceName[NET_DEV_NAME_LEN]; //设备序列号 BYTE byRes[24]; }NET_DVR_ACS_EVENT_INFO_EXTEND, *LPNET_DVR_ACS_EVENT_INFO_EXTEND; //扩展结构体信息 V20 typedef struct tagNET_DVR_ACS_EVENT_INFO_EXTEND_V20 { BYTE byRemoteCheck; //是否需要远程核验(0-无效, 1-不需要(默认), 2-需要) BYTE byThermometryUnit; //测温单位(0-摄氏度(默认),1-华氏度,2-开尔文) BYTE byIsAbnomalTemperature; //人脸抓拍测温是否温度异常: 1-是, 0-否 BYTE byRes2; float fCurrTemperature; //人脸温度(精确到小数点后一位) NET VCA POINT struRegionCoordinates; //人脸温度坐标 DWORD dwQRCodeInfoLen; //二维码信息长度,不为 0 是表示后面带数据 DWORD dwVisibleLightDataLen; //热成像相机可见光图片长度,不为 0 是表示后面带数据 DWORD dwThermalDataLen; //热成像图片长度,不为 0 是表示后面带数据 char *pQRCodeInfo; //二维码信息指针 char *pVisibleLightData; //热成像相机可见光图片指针 char *pThermalData; //热成像图片指针 BYTE byAttendanceLabel[64]; //考勤自定义名称 BYTE byRes[960]; NET_DVR_ACS_EVENT_INFO_EXTEND_V20, *LPNET_DVR_ACS_EVENT_INFO_EXTEND_V20; typedef struct tagNET_DVR_ACS_ALARM_INFO

{

DWORD dwSize:

DWORD dwMajor; //报警主类型,参考宏定义 DWORD dwMinor; //报警次类型,参考宏定义

NET DVR TIME struTime; //时间



```
sNetUser[MAX_NAMELEN];//网络操作的用户名
   BYTE
   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
         *pPicData;
   char
   WORD wInductiveEventType; //归纳事件类型, 0-无效, 客户端判断该值为非 0 值后, 报警类
型通过归纳事件类型区分,否则通过原有报警主次类型(dwMajor、dwMinor)区分
   BYTE
           byPicTransType;
                             //图片数据传输方式: 0-二进制; 1-url
   BYTE
           byRes1;
                           //保留字节
   DWORD dwIOTChannelNo;
                          //IOT 通道号
                                //byAcsEventInfoExtend 为 1 时 ,表示指向一个
           *pAcsEventInfoExtend;
NET_DVR_ACS_EVENT_INFO_EXTEND 结构体
                              //pAcsEventInfoExtend 是否有效: 0-无效, 1-有效
   BYTE
           byAcsEventInfoExtend;
           byTimeType; //时间类型: 0-设备本地时间, 1-UTC 时间(struTime 的时间)
   BYTE
   BYTE
           byRes2;
                           //保留字节
                                 //pAcsEventInfoExtendV20 是否有效: 0-无效, 1-有效
   BYTE
           byAcsEventInfoExtendV20;
                                 //byAcsEventInfoExtendV20 为 1 时,表示指向一个
   char
          *pAcsEventInfoExtendV20;
NET DVR ACS EVENT INFO EXTEND V20 结构体
   BYTE byRes[4];
}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;
```

2.6.4 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)

2.7 主动获取设备事件

2.7.1 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_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 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_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 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

2.7.2 命令码

#define NET_DVR_GET_ACS_EVENT 2514 //设备事件获取

2.7.3 宏定义及结构体

ዹ 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

ዹ 结构体:

typedef struct tagNET_DVR_ACS_EVENT_COND

DWORD dwSize;

DWORD dwMajor; //报警主类型,参考事件上传宏定义,0-全部 DWORD dwMinor; //报警次类型,参考事件上传宏定义,0-全部

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 byTimeType; //时间类型:0-设备本地时间(默认),1-UTC时间(struStartTime和struEndTime的时间)

BYTE byRes2[2]; //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)

DWORD dwIOTChannelNo; //IOT 通道号, 0-无效

WORD wInductiveEventType; //归纳事件类型, 0-无效, 其他值参见 2.2 章节, 客户端判断该值为非 0 值后, 报警类型通过归纳事件类型区分, 否则通过原有报警主次类型(dwMajor、dwMinor)区分

BYTE bySearchType; //搜索方式: 0-保留, 1-按事件源搜索(此时通道号为非视频通道号), 2-按监控点 ID 搜索

BYTE byEventAttribute; //事件属性: 0-未定义, 1-合法事件, 2-其它



char szMonitorID[NET_SDK_MONITOR_ID_LEN/*64*/]; //监控点 ID(由设备序列号、通道类型、编号组成,例如门禁点:设备序列号+"DOOR"+门编号)

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

BYTE byRes[140]; //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, 3-禁止名单卡, 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 byAllowListNo; //允许名单单号,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 byEventAttribute; //事件属性: 0-未定义, 1-合法认证, 2-其它

DWORD dwSerialNo; //事件流水号,为0无效

BYTE byChannelControllerID; //通道控制器 ID, 为 0 无效, 1-主通道控制器, 2-从通道控制器

BYTE byChannelControllerLampID; //通道控制器灯板 ID, 为 0 无效(有效范围 1-255)

BYTE byChannelControllerIRAdaptorID; //通道控制器红外转接板 ID, 为 0 无效(有效范围 1-255)

BYTE byChannelControllerIREmitterID; //通道控制器红外对射 ID, 为 0 无效(有效范围 1-255) DWORD dwRecordChannelNum; //录像通道数目



char *pRecordChannelData;//录像通道,大小为 sizeof(DWORD)* dwRecordChannelNum BYTE byUserType; //人员类型: 0-无效,1-普通人(主人),2-来宾(访客),3-禁止名单人,4-管理员

BYTE byCurrentVerifyMode; //读卡器当前验证方式: 0-无效, 1-休眠, 2-刷卡+密码, 3-刷卡, 4-刷卡或密码, 5-指纹, 6-指纹+密码, 7-指纹或刷卡, 8-指纹+刷卡, 9-指纹+刷卡+密码, 10-人脸或指纹或刷卡或密码, 11-人脸+指纹, 12-人脸+密码,

//13-人脸+刷卡,14-人脸,15-工号+密码,16-指纹或密码,17-工号+指纹,18-工号+指纹+密码,19-人脸+指纹+刷卡,20-人脸+密码+指纹,21-工号+人脸,22-人脸或人脸+刷卡,23-指纹或人脸,24-刷卡或人脸或密码,25-刷卡或人脸,26-刷卡或人脸或指纹,27-刷卡或指纹或密码

BYTE byAttendanceStatus; //考勤状态: 0-未定义,1-上班, 2-下班, 3-开始休息, 4-结束休息, 5-开始加班, 6-结束加班

BYTE byStatusValue; //考勤状态值

BYTE byEmployeeNo[NET_SDK_EMPLOYEE_NO_LEN]; //工号(人员 ID)(对于设备来说,如果使用了工号(人员 ID)字段,byEmployeeNo 一定要传递,如果 byEmployeeNo 可转换为dwEmployeeNo,那么该字段也要传递;对于上层平台或客户端来说,优先解析 byEmployeeNo 字段,如该字段为空,再考虑解析 dwEmployeeNo 字段)

BYTE byRes1; //保留

BYTE byMask; //是否带口罩: 0-保留, 1-未知, 2-不戴口罩, 3-戴口罩

BYTE byThermometryUnit; //测温单位(0-摄氏度(默认),1-华氏度,2-开尔文)

BYTE byIsAbnomalTemperature; //人脸抓拍测温是否温度异常: 1-是, 0-否

float fCurrTemperature; //人脸温度(精确到小数点后一位)

NET_VCA_POINT struRegionCoordinates; //人脸温度坐标

BYTE byRes[48];

}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,参考宏定义 DWORD dwMinor; //报警次类型,参考宏定义

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;

WORD wInductiveEventType; //归纳事件类型, 0-无效, 其他值参见 2.2 章节, 客户端判断该值为非 0 值后, 报警类型通过归纳事件类型区分, 否则通过原有报警主次类型(dwMajor、dwMinor)区分

BYTE byTimeType; //时间类型: 0-设备本地时间(默认),1-UTC 时间(struTime 的时间)BYTE byRes1;

DWORD dwQRCodeInfoLen; //二维码信息长度,不为 0 是表示后面带数据

DWORD dwVisibleLightDataLen; //热成像相机可见光图片长度,不为 0 是表示后面带数据

DWORD dwThermalDataLen; //热成像图片长度,不为 0 是表示后面带数据



char *pQRCodeInfo; //二维码信息指针

char *pVisibleLightData; //热成像相机可见光图片指针

char *pThermalData; //热成像图片指针

BYTE byRes[36];

}NET_DVR_ACS_EVENT_CFG, *LPNET_DVR_ACS_EVENT_CFG;

2.8人员权限计划模板管理

2.8.1 人员权限周计划配置

2.8.1.1 接口函数

获取卡权限周计划

NET_DVR_API BOOL __stdcall NET_DVR_GetDVRConfig(LONG lUserID, DWORD dwCommand,LONG lChannel, 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 lUserID, DWORD dwCommand,LONG lChannel, 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_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

2.8.1.2 命令码

#define NET_DVR_GET_CARD_RIGHT_WEEK_PLAN 2126 //获取权限周计划 #define NET_DVR_SET_CARD_RIGHT_WEEK_PLAN 2127 //设置权限周计划

2.8.1.3 宏定义及结构体

ዹ 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

ዹ 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;
```

2.8.2 人员权限假日计划配置

2.8.2.1 接口函数

获取卡权限假日计划

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

Parameters

lUserID

[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 lUserID, DWORD dwCommand,LONG lChannel, 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_PLAN

IChannel

[in] 假目计划编号(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

2.8.2.2 命令码

#define NET_DVR_GET_CARD_RIGHT_HOLIDAY_PLAN 2130 // 获取权限假日计划 #define NET_DVR_SET_CARD_RIGHT_HOLIDAY_PLAN 2131 // 设置权限假日计划

2.8.2.3 宏定义及结构体

ዹ 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
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;
2.8.3 人员权限假日组配置
2.8.3.1 接口函数
获取卡权限假日组
NET_DVR_API
              BOOL
                      __stdcall NET_DVR_GetDVRConfig(LONG
                                                           lUserID,
                                                                    DWORD
dwCommand,LONG lChannel, LPVOID lpOutBuffer, DWORD dwOutBufferSize, LPDWORD
lpBytesReturned);
```

Parameters

```
lUserID
```

```
[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

海康威视版权所有



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 lUserID, DWORD dwCommand,LONG lChannel, 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

IChannel

[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

2.8.3.2 命令码

#define NET_DVR_GET_CARD_RIGHT_HOLIDAY_GROUP 2134 // 获取权限假日组 #define NET_DVR_SET_CARD_RIGHT_HOLIDAY_GROUP 2135 // 设置权限假日组

2.8.3.3 宏定义及结构体

ዹ 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

ዹ 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;

2.8.4 人员权限计划模板配置

2.8.4.1 接口函数

获取卡权限计划模板

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

Parameters

1UserID

[in] NET_DVR_Login_V40 等登录接口的返回值

dwCommand

[in] NET_DVR_GET_CARD_RIGHT_PLAN_TEMPLATE

1Channel

[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 lUserID, DWORD dwCommand,LONG lChannel, 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_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

2.8.4.2 命令码

#define NET_DVR_GET_CARD_RIGHT_PLAN_TEMPLATE 2138 // 获取权限计划模板 #define NET_DVR_SET_CARD_RIGHT_PLAN_TEMPLATE 2139 // 设置权限计划模板

2.8.4.3 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

ዹ 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;

2.9 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_ECENT_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_BROKEN0x27 //网络断开#define MINOR_DEV_POWER_ON0x400 //设备上电启动#define MINOR_NET_RESUME0x407 //网络恢复

#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 0x71 //远程注销登陆 #define MINOR_REMOTE_LOGOUT 0x79 //远程布防 #define MINOR_REMOTE_ARM #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_INTPUT 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 0x407 //远程恢复默认参数 #define MINOR REMOTE RESTORE CFG #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 0x40e //修改中心组参数配置 #define MINOR_MOD_REPORT_GROUP_PARAM #define MINOR UNLOCK PASSWORD OPEN DOOR 0x40f //解除码输入 #define MINOR_REMOTE_ACTUAL_GUARD 0x419 //远程实时布防 0x41a //远程实时撤防 #define MINOR REMOTE ACTUAL UNGUARD /*事件*/ //主类型 #define MAJOR EVENT 0x5//次类型 0x01 //合法卡认证通过 #define MINOR_LEGAL_CARD_PASS #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 0x09 //无此卡号 #define MINOR INVALID CARD 0x0a //反潜回认证失败 #define MINOR_ANTI_SNEAK_FAIL 0x0c //卡不属于多重认证群组 #define MINOR_NOT_BELONG_MULTI_GROUP #define MINOR_INVALID_MULTI_VERIFY_PERIOD 0x0d //卡不在多重认证时间段内 0x0e //多重认证模式超级权限认证失败 #define MINOR_MULTI_VERIFY_SUPER_RIGHT_FAIL 0x0f //多重认证模式远程认证失败 #define MINOR_MULTI_VERIFY_REMOTE_RIGHT_FAIL #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 0x12 //首卡开门结束 #define MINOR_LEADER_CARD_OPEN_END 0x13 //常开状态开始 #define MINOR_ALWAYS_OPEN_BEGIN 0x14 //常开状态结束 #define MINOR_ALWAYS_OPEN_END #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 0x18 //开门按钮放开 #define MINOR_DOOR_BUTTON_RELEASE #define MINOR_DOOR_OPEN_NORMAL 0x19 //正常开门(门磁) #define MINOR_DOOR_CLOSE_NORMAL 0x1a //正常关门(门磁) 0x1b //门异常打开(门磁)

#define MINOR_DOOR_OPEN_ABNORMAL



#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_BEGIN #define MINOR_ALWAYS_CLOSE_END	0x10 //帝关状态/f 妇 0x20 //常关状态结束
#define MINOR_MULTI_VERIFY_NEED_REMOTE_OPEN	V 0X21 // 岁里夕里以证而安远柱月
	FY SUCCESS 0x22 //多重认证超级密
#define MINOR_MULTI_VERIFY_SUPERPASSWD_VERI 码认证成功事件	FY_SUCCESS 0X22 //多里认证起级铝
#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_TIMEOU	JT 0x2a //刷卡加指纹认证超时
#define MINOR_CARD_FINGERPRINT_PASSWD_VERIF	Y_PASS 0x2b //刷卡加指纹加密码认
证通过	
#define MINOR_CARD_FINGERPRINT_PASSWD_VERIF	Y_FAIL 0x2c //刷卡加指纹加密码认
证失败	
#define MINOR_CARD_FINGERPRINT_PASSWD_VERIF	Y_TIMEOUT 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_TIME	OUT 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	S 0x3f //人脸加密码加指纹认证通过
#define MINOR_FACE_AND_PW_AND_FP_VERIFY_FAIL	L 0x40 //人脸加密码加指纹认证失败
#define MINOR_FACE_AND_PW_AND_FP_VERIFY_TIM	IEOUT 0x41 //人脸加密码加指
纹认证超时	
#define MINOR_FACE_CARD_AND_FP_VERIFY_PASS	0x42 //人脸加刷卡加指纹
认证通过	
#define MINOR_FACE_CARD_AND_FP_VERIFY_FAIL	0x43 //人脸加刷卡加指纹



认证失败

#define MINOR FACE CARD AND FP VERIFY TIMEOUT 0x44 //人脸加刷卡加指 纹认证超时 #define MINOR EMPLOYEENO AND FP VERIFY PASS 0x45 //工号加指纹认证通过 #define MINOR_EMPLOYEENO_AND_FP_VERIFY_FAIL //工号加指纹认证失败 0x46 //工号加指纹认证超时 #define MINOR EMPLOYEENO AND FP VERIFY TIMEOUT 0x47 #define MINOR_EMPLOYEENO_AND_FP_AND_PW_VERIFY_PASS 0x48 //工号加指纹加 密码认证通过 #define MINOR_EMPLOYEENO_AND_FP_AND_PW_VERIFY_FAIL 0x49 //工号加指纹加密 码认证失败 #define MINOR EMPLOYEENO AND FP AND PW VERIFY TIMEOUT 0x4a //工号加指纹加 密码认证超时 0x4b //人脸认证通过 #define MINOR FACE VERIFY PASS #define MINOR_FACE_VERIFY_FAIL 0x4c //人脸认证失败 #define MINOR_EMPLOYEENO_AND_FACE_VERIFY_PASS 0x4d //工号加人脸认证通过 0x4e //工号加人脸认证失败 #define MINOR_EMPLOYEENO_AND_FACE_VERIFY_FAIL #define MINOR_EMPLOYEENO_AND_FACE_VERIFY_TIMEOUT 0x4f //工号加人脸认证超时 0x50 //人脸识别失败 #define MINOR FACE RECOGNIZE FAIL 0x51 //首卡授权开始 #define MINOR_FIRSTCARD_AUTHORIZE_BEGIN

#define MINOR FIRSTCARD AUTHORIZE END #define MINOR_EMPLOYEENO_AND_PW_PASS

#define MINOR_EMPLOYEENO_AND_PW_FAIL #define MINOR EMPLOYEENO AND PW TIMEOUT

#define MINOR_HUMAN_DETECT_FAIL

#define MINOR_PEOPLE_AND_ID_CARD_COMPARE_PASS #define MINOR_PEOPLE_AND_ID_CARD_COMPARE_FAIL

#define MINOR CERTIFICATE BLACK LIST

0x52 //首卡授权结束

0x65 //工号加密码认证通过

0x66 //工号加密码认证失败

0x67 //工号加密码认证超时

0x68 //真人检测失败

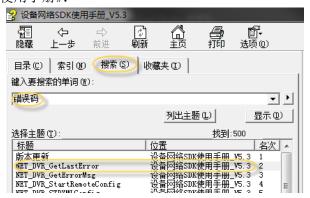
0x69 //人证比对通过

0x70 //人证比对失败

0x71 //禁止名单事件

2.10 错误码表

详情见《设备网络 SDK 使用手册》:





DVR_GetLastError		
B最后操作的 <mark>造模型</mark> 。		
NORD NET_DVR_GetLastError(
Values		
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	讲库等 范德丽 ,详见下表。	
	网络	通讯库错误码
# 袭类型	错误位	機義信息
NET_DVR_NOERROR	0	沒有错误.
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,一般和通道指关,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 资通道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。 设备设有对应的通道号。
NET_DVR_OVER_MAXLINK	5	设备总的连接数超过最大。
IET_DVR_VERSIONNOMATCH	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	传送的数据有误。 反送给设备或者从设备接收到的数据错误,如选程部数据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	设备命令执行程时。
NET_DVR_ERRORSERIALPORT	15	事口号错误。指定的设备事口号不存在。
IET_DVR_ERRORALARMPORT	16	报警端口错误。指定的设备报警输入载者输出端口不存在。
HET_DVR_PARAMETER_ERROR	17	参数错误。SDK接口中给入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
ET_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该错误。
NET_DVR_DISK_FULL	21	设备硬盘溝。
NET_DVR_DISK_ERROR	22	设备硬盘出错
NET_DVR_NOSUPPORT	23	设备不支持。

2.11 ResponseStatus(JSON 格式)

"requestURL": "", //可选, string, URI

"statusCode":, //可选, integer, 状态码, 无法用 1 表示时(1 表示成功且无特殊状态), 否则必须返回

"statusString": "", //可选, string, 状态描述, 无法用 ok 表示时(ok 表示成功且无特殊状态), 否则必须返回

"subStatusCode": "", //可选, string, 子状态码, 无法用 ok 表示(ok 表示成功且无特殊状态), 否则必须返回

"errorCode": 1, //必填,integer,错误码,当 statusCode 不为 1 时,与 subStatusCode 对应

"errorMsg": "ok" //必填,string,错误信息,对 errorCode 的解释,当 statusCode 不为 1 时,解释信息在协议约束中,允许设备在后续的版本迭代中,进行优化丰富提升(不限制死)

3. 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.人员、卡、指纹、人脸之间通过人员 ID (employeeNo) 进行关联,设备内的人员 ID 要保证唯一性:
- 4.人员、人员权限计划模板之前通过计划模板编号(RightPlan)进行关联;
- 5.下发卡、指纹、人脸前,首先要保证人员已经下发;