# Access Control Device Alternate SDK Port

**Manual**

1. **SDK Intro**

Network SDK is kit developed for NVR, NVS, network camera, speed dome and etc. This manual describes function, port and call among functions in each development kit with example.

This manual mainly introduces alarm host related function, support the following models：

|  |  |
| --- | --- |
| Model | Device Name |
| BSC12XXA | **A&C Host** |
| BSC12XXB | **Two-door Controller** |
| BSR11XXA | A&C Card Reader |

Device SDK under Windows

|  |  |  |
| --- | --- | --- |
| Functional Module | File | **Note** |
| Function Library | avglobal.h | **head file** |
| dhassistant.h | **head file** |
| dhnetsdk.h | **head file** |
| dhnetsdk.lib | **Lib file** |
| dhnetsdk.dll | **Dll file** |
| avnetsdk.dll | **Dll file** |
| Config Library | dhconfigsdk.h | **Config head file** |
| dhconfigsdk.lib | **Config Lib file** |
| dhconfigsdk.dll | **Config Dll file** |
| avnetsdk.dll Aux Library | **Infra.dll** | **function aux library** |
| **Json.dll** | **function aux library** |
| **Stream.dll** | **function aux library** |
| **NetFramework.dll** | **function aux library** |
| **StreamSvr.dll** | **function aux library** |

Device SDK under Linux

|  |  |  |
| --- | --- | --- |
| Functional Module | File | **Note** |
| Function Library 1 | avglobal.h | **head file** |
| dhassistant.h | **head file** |
| dhnetsdk.h | **head file** |
| libdhdvr.so | basic network library |
| libdhnetsdk.so | function library 1 |
| Function Library 2 | libavnetsdk.so | function library 2 |
| Config Library | dhconfigsdk.h | **head file** |
| libdhconfigsdk.so | Config library |
| libavnetsdk.so Aux Library | **libInfra.so** | **function aux library** |
| **libJson.so** | **function aux library** |
| **libStream.so** | **function aux library** |
| **libNetFramework.so** | **function aux library** |

This SDK Function Library and config library are mandatory. If contain avnetsdk.dll，aux library 2 is mandatory too. If no other encoding tool, take play library and its aux library.

**Function Library** is main body of device network SDK, used mainly in communication among network client and products and is responsible to control, search, config, get and process stream data.

**Config library** is to pack and analyze structure of config function.

**Aux library** is mandatory library of avnetsdk. Without these files will influence usage of avnetsdk.

1. **Terminology**

**Forced Card:**

One of A&C card, used to unlock door when cardholder is threatened by someone. A&C system will identify forced card and alarm, activate other operations.

Table of Contents

[Access Control Device Alternate SDK Port 1](#_Toc386635607)

[**I．** **SDK Intro** 2](#_Toc386635608)

[**II．** **Terminology** 4](#_Toc386635609)

[**1．** **Function Call Sequence** 6](#_Toc386635610)

[**2．** **Universal Port** 6](#_Toc386635611)

[**3．** **Device Basic Information** 7](#_Toc386635612)

[**4．** **Listen to Event** 10](#_Toc386635613)

[**5．** **Config Port** 11](#_Toc386635614)

[**6．** **Control Port** 13](#_Toc386635615)

[**7．** **Record Operation** 13](#_Toc386635616)

[**8．** **Record Set Search** 14](#_Toc386635617)

[9． **Alarm Record(Log Search)** 17](#_Toc386635618)

[**10．** **A&C Status** 19](#_Toc386635619)

1. **Function Call Sequence**
2. SDK initialization(**CLIENT\_Init()**), login(**CLIENT\_Login()**)and etc.
3. Get device capacity information(**CLIENT\_QueryNewSystemInfo()**)
4. Configure device(**CLIENT\_GetNewDevConfig()**, **CLIENT\_SetNewDevConfig()** and etc. )
5. Control device, subscription, record search, login search(**CLIENT\_ControlDevice()**, **CLIENT\_SetDVRMessCallBack()**,**CLIENT\_StartListenEx()**, **CLIENT\_StopListen(), CLIENT\_FindRecord(),CLIENT\_FindNextRecord(),CLIENT\_QueryRecordCount(), CLIENT\_FindRecordClose() , CLIENT\_QueryDeviceLog(),CLIENT\_QueryDevLogCount()**)
6. Logout, SDK anti-initialization(**CLIENT\_Logout()**, **CLIENT\_Cleanup()**)
7. **Universal Port**

1). Set login network port

void **CLIENT\_SetNetworkParam**(NET\_PARAM\* *pNetParam*);

Port note：Set login network port. Call port before login is recommended. If not call, SDK uses default value.

Parameter：

[in] *pNetParam* network parameter structure indicator, please refer to NET\_PARAM

Return value：N/A

NET\_PARAM structure:

// Set login parameter

typedef struct

{

int nWaittime; // and etc. overtime waiting(unit is ms)，as 0 default 5000ms

int nConnectTime; // overtime connection(unit is ms)，as 0 default 1500ms

int nConnectTryNum; // connection attempts，as 0 default 1 time

int nSubConnectSpaceTime; // sum of sub connection waiting time(unit is ms)，as 0 default 10ms

int nGetDevInfoTime; // get device information overtime，as 0 default 1000ms

int nConnectBufSize; // each connection receiving data buffer size(unit is byte)，as 0 default 250\*1024

int nGetConnInfoTime; // get sub connection information overtime(unit is ms)，as 0 default 1000ms

int nSearchRecordTime; // by time search record file overtime(unit is ms),as 0 default is 3000ms

int nsubDisconnetTime; // detect sub connection offline waiting time sum(unit is ms)，as 0 default is 60000ms

BYTE byNetType; // network type, 0-LAN, 1-WAN

BYTE byPlaybackBufSize; // playback data receiving buffer size（unit is M），as 0 default is 4M

BYTE byReserved1[2]; // Align

Int nPicBufSize; // real time picture receiving buffer size（unit is byte），as 0 default 2\*1024\*1024

BYTE bReserved[4]; // reserved text

} NET\_PARAM;

1. **Device Basic Information**

**3.1 Device Type**

Port：LLONG **CLIENT\_Login**(char \**pchDVRIP*, WORD *wDVRPort*, char \**pchUserName*, char \**pchPassword*, LPNET\_DEVICEINFO *lpDeviceInfo*, int \**error* = 0);

Parameter：

|  |  |
| --- | --- |
| [in] pchDVRIP | Device IP |
| [in] wDVRPort | Device Port |
| [in] pchUserName | Username |
| [in] pchPassword | Password |
| [out] lpDeviceInfo | Device Info |
| [out] error | If port failed, return error code |

// Device Info

typedef struct

{

BYTE sSerialNumber[DH\_SERIALNO\_LEN]; // SN

BYTE byAlarmInPortNum; // DVR alarm input no.

BYTE byAlarmOutPortNum; // DVR alarm output no.

BYTE byDiskNum; // DVR HDD quantity

BYTE byDVRType; // DVR type, see NET\_DEVICE\_TYPE

BYTE byChanNum; // DVR channel quantity

} NET\_DEVICEINFO, \*LPNET\_DEVICEINFO;

**3.2 Software Version**

Port：BOOL **CLIENT\_QueryDevState**(LLONG *lLoginID*, int *nType*, char \**pBuf*, int *nBufLen*, int \**pRetLen*, int *waittime*);

Parameter：

|  |  |
| --- | --- |
| [in]lLoginID | CLIENT\_Login() return value |
| [in]nType | DH\_DEVSTATE\_SOFTWARE |
| [in] pBuf | Corresponding structure DHDEV\_VERSION\_INFO |
| [in] nBufLen | Structure Size |
| [in] pRetLen | Return structure size |
| [in] waittime | Overtime, unit is ms |

typedef struct

{

char szDevSerialNo[DH\_DEV\_SERIALNO\_LEN]; // SN

char byDevType; // Device Type, see NET\_DEVICE\_TYPE

char szDevType[DH\_DEV\_TYPE\_LEN]; // Device detailed model, string format, may be null

int nProtocalVer; // Protocol Version

char szSoftWareVersion[DH\_MAX\_URL\_LEN];

DWORD dwSoftwareBuildDate;

char szDspSoftwareVersion[DH\_MAX\_URL\_LEN];

DWORD dwDspSoftwareBuildDate;

char szPanelVersion[DH\_MAX\_URL\_LEN];

DWORD dwPanelSoftwareBuildDate;

char szHardwareVersion[DH\_MAX\_URL\_LEN];

DWORD dwHardwareDate;

char szWebVersion[DH\_MAX\_URL\_LEN];

DWORD dwWebBuildDate;

char reserved[256];

} DHDEV\_VERSION\_INFO;

**3.3 Date Setup/Get**

Port：BOOL **CLIENT\_QueryDeviceTime**(LLONG *lLoginID*, LPNET\_TIME *pDeviceTime*, int *waittime*=1000);

Port Note：Apply to get current front-end device time for sync time

Note：

[in] lLoginID CLIENT\_Login() return value，when it is 0, it means abnormal login

[out] pDeviceTime Receiving device time indicator, see LPNET\_TIME

[in] waittime and waiting overtime，default 1000ms

Return value：successfully return TRUE，failed to return FALSE. Port failed to return please call CLIENT\_GetLastError()to get error code，find cause of error via error code.

BOOL **CLIENT\_SetupDeviceTime**(LLONG *lLoginID*, LPNET\_TIME *pDeviceTime*);

Port note：Apply to sync current front-end device time with this device time.

Parameter note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] pDeviceTime Set device time indicator，see LPNET\_TIME

Return value：successfully return TRUE，failed return to FALSE. Port failed to return please callCLIENT\_GetLastError()to get error code，find error cause via error code.

LPNET\_TIME Structure：

// Time

typedef struct

{

DWORD dwYear; // Year

DWORD dwMonth; // Month

DWORD dwDay; // Day

DWORD dwHour; // Hour

DWORD dwMinute; // Minute

DWORD dwSecond; // Second

} NET\_TIME,\*LPNET\_TIME;

**3.4 MAC**

Port：BOOL **CLIENT\_QueryDevState**(LLONG *lLoginID*, int *nType*, char \**pBuf*, int *nBufLen*, int \**pRetLen*, int *waittime*);

Parameter：

|  |  |
| --- | --- |
| [in]lLoginID | CLIENT\_Login() return value |
| [in]nType | DH\_DEVS TATE\_NETINTERFACE |
| [in] pBuf | Corresponding structure DHDEV\_NETINTERFACE\_INFO |
| [in] nBufLen | Structure size |
| [in] pRetLen | Return structure size |
| [in] waittime | Overtime, unit is ms |

typedef struct tagDHDEV\_NETINTERFACE\_INFO

{

int dwSize;

BOOL bValid; // Valid or not

BOOL bVirtual; // Virtual network card

int nSpeed; // Network card theoretical speed(Mbps)

int nDHCPState; // 0-not enabled, 1-getting, 2-successfully got

char szName[DH\_NETINTERFACE\_NAME\_LEN]; // Network Name

char szType[DH\_NETINTERFACE\_TYPE\_LEN]; // Network Type

char szMAC[DH\_MACADDR\_LEN]; // MAC address

char szSSID[DH\_MAX\_SSID\_LEN]; // SSID, only valid for wireless network(szType == "Wireless")

char szConnStatus[DH\_MAX\_CONNECT\_STATUS\_LEN]; // Wifi, 3G connection status，"Inexistence" : network inexists "Down"：off "Disconn"：not connected "Connecting"：connecting "Connected"： connected

int nSupportedModeNum; // Actual 3G supported network mode number

char szSupportedModes[DH\_MAX\_MODE\_NUM][DH\_MAX\_MODE\_LEN];// 3G supported network mode "TD-SCDMA", "WCDMA", "CDMA1x", "EDGE", "EVDO"

} DHDEV\_NETINTERFACE\_INFO;

1. **Listen to Event**

0).Set call function to get event info:

void **CLIENT\_SetDVRMessCallBack**(fMessCallBack *cbMessage*, LDWORD *dwUser*);

Port note: set call function to listen to event

Note:

[in] cbMessage message call function，may call device status，if alarm status can get via this call; when setup is 0, it means no call allowed.

typedef BOOL (CALLBACK \*fMessageCallBack)(

LONG *lCommand*, Call type，see “supported event”

LLONG *lLoginID*, CLIENT\_Login return value

Char *\*pBuf*, Receive alarm data buffer，and correspond to lCommand value，see“supported event”

DWORD *dwBufLen*, pBuf length(unit is byte)

Char *\*pchDVRIP*, Device ip

LONG *nDVRPort*, Port

LDWORD *dwUser* User custom data

);

[in] dwUser User custom data

Return value: N/A

1).Enable listening to event:

BOOL **CLIENT\_StartListenEx**(LLONG *lLoginID*);

Port note: enable listening to event

Note:

[in] lLoginID CLIENT\_Login() return value

Return value: TRUE is successful，FALSE is failed

2).Stop listening to event:

BOOL **CLIENT\_StopListen**(LLONG *lLoginID*);

Port note: stop listening to event

Parameter note:

[in] lLoginID CLIENT\_Login() return value

Return value: TRUE is successful，FALSE is failed

**3).Supported Event**

|  |  |  |
| --- | --- | --- |
| Event Name | Command Type | Info(usually as structure) |
| A&C event | DH\_ALARM\_ACCESS\_CTL\_EVENT | ALARM\_ACCESS\_CTL\_EVENT\_INFO |
| A&C not locked event info | DH\_ALARM\_ACCESS\_CTL\_NOT\_CLOSE | ALARM\_ACCESS\_CTL\_NOT\_CLOSE\_INFO |
| Intrusion event info | DH\_ALARM\_ACCESS\_CTL\_BREAK\_IN | ALARM\_ACCESS\_CTL\_BREAK\_IN\_INFO |
| Repeated entry event info | DH\_ALARM\_ACCESS\_CTL\_REPEAT\_ENTER | ALARM\_ACCESS\_CTL\_REPEAT\_ENTER\_INFO |
| Forced card swiping event info | DH\_ALARM\_ACCESS\_CTL\_DURESS | ALARM\_ACCESS\_CTL\_DURESS\_INFO |

1. **Config Port**
   1. BOOL **CLIENT\_GetNewDevConfig**(LLONG *lLoginID*, char\* *szCommand*, int *nChannelID*, char\* *szOutBuffer*, DWORD *dwOutBufferSize*, int \**error*, int *waittime*);

Function note： get config，follow string format，each string included info is analyzed by CLIENT\_ParseData.

Note：

[in] lLoginID CLIENT\_Login() return value，when it is 0, it means abnormal login

[in] szCommand Command parameter， see following CLIENT\_ParseData command parameter note

[in] nChannelID Channel no.

[out] szOutBuffer Output buffer

[out] dwOutBufferSize Output buffer size

[out] error Error code

|  |  |
| --- | --- |
| 0 | Successful |
| 1 | Failed |
| 2 | Invalid data |
| 3 | Temporarily cannot set |
| 4 | No right |

[in] waittime and waiting overtime

* 1. BOOL **CLIENT\_ParseData**(char\* *szCommand*, char\* *szInBuffer*, LPVOID *lpOutBuffer*, DWORD *dwOutBufferSize*, void\* *pReserved*);

Note：This port is used with CLIENT\_GetNewDevConfig()，after using CLIENT\_GetNewDevConfig() to get string config info，use this port get info from config info.

Parameter note：

[in] szCommand command parameter

|  |  |
| --- | --- |
| Command Parameter | Corresponding Structure |
| CFG\_CMD\_NETWORK | Network config（IP, mask, gateway），corresponding to CFG\_NETWORK\_INFO |
| CFG\_CMD\_ACCESS\_GENERAL | Access control basic config, corresponding to CFG\_ACCESS\_GENERAL\_INFO |
| CFG\_CMD\_ACCESS\_EVENT | A&C(each door) config，corresponding to CFG\_ACCESS\_EVENT\_INFO |
| CFG\_CAP\_CMD\_RECORDFINDER | Search record capacity set  CFG\_CAP\_RECORDFINDER\_INFO |

[in] szInBuffer Input buffer，string config buffer

[out] lpOutBuffer Output buffer，corresponding to szCommand structure type

[out] dwOutBufferSize Output buffer size

[out] pReserved Return data length

* 1. BOOL **CLIENT\_PacketData**(char\* *szCommand*, LPVOID *lpInBuffer*, DWORD *dwInBufferSize*, char\* *szOutBuffer*, DWORD *dwOutBufferSize*);

Note：This port is used with CLIENT\_SetNewDevConfig(). After using CLIENT\_PacketData()，set packed string info via CLIENT\_SetNewDevConfig() to device.

Note：

[in] szCommand Command parameter， see above CLIENT\_ParseData chart

[in] lpInBuffer Input buffer，structure see CLIENT\_ParseData chart

[in] dwInBufferSize Input buffer size

[out] szOutBuffer Output buffer

[out] dwOutBufferSize Output buffer size

* 1. BOOL **CLIENT\_SetNewDevConfig**(LLONG *lLoginID*, char\* *szCommand*, int *nChannelID*, char\* *szInBuffer*, DWORD *dwInBufferSize*, int\* *error*, int\* *restart*, int *waittime*=500);

Function：Set config，by string format，each string pack info is formed by CLIENT\_PacketData.

Note：

[in] lLoginID CLIENT\_Login() return value，when it is 0, it means abnormal login

[in] szCommand Please refer to CLIENT\_ParseData note

[in] nChannelID Channel No.

[in] szInBuffer Input buffer

[in] dwInBufferSize Input buffer size

[out] error Error code

|  |  |
| --- | --- |
| 0 | Successful |
| 1 | Failed |
| 2 | Data invalid |
| 3 | Temporarily cannot set |
| 4 | No right |

[out] restart After config, you shall reboot device，1 means reboot启，0 means no reboot

[in] waittime and waiting overtime

1. **Control Port**

BOOL **CLIENT\_ControlDevice**(LLONG *lLoginID*, CtrlType *type*, void \* *param*, int *waittime* = 1000);

Note：device control port function

Note：

[in] lLoginID CLIENT\_Login() return value, when it is 0, it means abnormal login

[in] type Control type，see following chart CtrlType

[in] param During control, input parameter structure，matching type

|  |  |
| --- | --- |
| Control Type | Corresponding Structure |
| DH\_CTRL\_ACCESS\_OPEN | A&C –unlock (corresponding structure NET\_CTRL\_ACCESS\_OPEN) |
| DH\_CTRL\_REBOOT | Reboot device |
| DH\_CTRL\_RESTOREDEFAULT | Restore default setup |
| DH\_CTRL\_ACCESS\_CLOSE | A&C control-lock(corresponding structure NET\_CTRL\_ACCESS\_CLOSE) |

[in] waittime and waiting overtime，default 1000ms

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code.

1. **Record Operation**

BOOL **CLIENT\_ControlDevice**(LLONG *lLoginID*, CtrlType *type*, void \* *param*, int *waittime* = 1000);

Port note：Device control port function

Note：

[in] lLoginID CLIENT\_Login() return value，when it is 0, it means abnormal login

[in] type Control type，see following chart CtrlType

[in] param During control, input parameter structure info，marching type

|  |  |
| --- | --- |
| Control Type | Corresponding Structure |
| DH\_CTRL\_RECORDSET\_INSERT | Add record, get record set no.(corresponding NET\_CTRL\_RECORDSET\_INSERT\_PARAM) |
| DH\_CTRL\_RECORDSET\_UPDATE | Update come record no.(corresponding NET\_CTRL\_RECORDSET\_PARAM) |
| DH\_CTRL\_RECORDSET\_REMOVE | According to record set no., delete record(corresponding NET\_CTRL\_RECORDSET\_PARAM) |
| DH\_CTRL\_RECORDSET\_CLEAR | Clear all record set info(corresponding NET\_CTRL\_RECORDSET\_PARAM) |

[in] waittime and waiting overtime，default 1000ms

Return value：successfully return TRUE，failed return FALSE. Port return failed please call CLIENT\_GetLastError() to get error code，find error cause via error code..

BOOL **CLIENT\_QueryDevState**(LLONG *lLoginID*, int *nType*, char\* *pBuf*, int *nBufLen*, int\* *pRetLen*, int *waittime*=1000);

Port note：search device status port

Note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] nType Search info type

[out] pBuf Output parameter, used to get search return data buffer. According to search type, return data structures are different, see

|  |  |
| --- | --- |
| Search Info Type | Corresponding Structure |
| DH\_DEVSTATE\_DEV\_RECORDSET | Search device record info(corresponding NET\_CTRL\_RECORDSET\_PARAM) |

[in] nBufLen Buffer length，unit is byte

[out] pRetLen Output parameter，actual return data length，unit is byte

[in] waittime Search status and waiting time， default 1000ms

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

1. **Record Set Search**

BOOL **CLIENT\_QueryNewSystemInfo** (LLONG *lLoginID*, char\* *szCommand*, int *nChannelID*, char\* *szOutBuffer*, DWORD *dwOutBufferSize*, int \**error*, int *waittime*=1000);

Port note：get record search capacity.

Parameter note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] szCommand Command parameter，CFG\_CAP\_CMD\_RECORDFINDER

[in] nChannelID Channel no.

[out] szOutBuffer Output buffer，see following CFG\_CAP\_RECORDFINDER\_INFO

[in] dwOutBufferSize Output buffer size

[out] error Error Code

[in] waittime Overtime

typedef struct tagCFG\_CAP\_RECORDFINDER\_INFO

{

int nMaxPageSize; // Max page size

}CFG\_CAP\_RECORDFINDER\_INFO;

BOOL **CLIENT\_FindRecord**(LLONG *lLoginID*, NET\_IN\_FIND\_RECORD\_PARAM\* *pInParam*, NET\_OUT\_FIND\_RECORD\_PARAM\* *pOutParam*, int *waittime*=1000);

Port note：Search record by filter，generate search handle.

Note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] pInParam Search record parameter，see NET\_IN\_FIND\_RECORD\_PARAM

[out] pOutParam Return search handle，see NET\_OUT\_FIND\_RECORD\_PARAM

[in] waittime and waiting overtime

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

Input parameter：

typedef struct \_NET\_IN\_FIND\_RECORD\_PARAM

{

DWORD dwSize;

EM\_NET\_RECORD\_TYPE emType; // Record type to search

void\* pQueryCondition; // Search type corresponding filter

} NET\_IN\_FIND\_RECORD\_PARAM;

Output parameter:

typedef struct \_NET\_OUT\_FIND\_RECORD\_PARAM

{

DWORD dwSize;

LLONG lFindeHandle; // Search record handle，SN search

} NET\_OUT\_FIND\_RECORD\_PARAM;

BOOL **CLIENT\_FindNextRecord**(NET\_IN\_FIND\_NEXT\_RECORD\_PARAM\* *pInParam*, NET\_OUT\_FIND\_NEXT\_RECORD\_PARAM \* *pOutParam*, int *waittime*=1000);

Port note：get record，search handle as CLIENT\_FindRecord port to get.

Note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] pInParam Search record parameter，see NET\_IN\_FIND\_NEXT\_RECORD\_PARAM

[out] pOutParam Return record，see NET\_OUT\_FIND\_NEXT\_RECORD\_PARAM

[in] waittime and waiting overtime

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

Input parameter：

typedef struct \_NET\_IN\_FIND\_NEXT\_RECORD\_PARAM

{

DWORD dwSize;

LLONG lFindeHandle; // Search handle

int nFileCount; // Current search record item number

} NET\_IN\_FIND\_NEXT\_RECORD\_PARAM;

Output parameter:

typedef struct \_NET\_OUT\_FIND\_NEXT\_RECORD\_PARAM

{

DWORD dwSize;

void\* pRecordList; // Record parameter，user allocate memory

int nMaxRecordNum; // List record number

int nRetRecordNum; // Searched record item，when searched item is less then designated item quantity, search ends

} NET\_OUT\_FIND\_NEXT\_RECORD\_PARAM;

BOOL **CLIENT\_QueryRecordCount**(NET\_IN\_QUEYT\_RECORD\_COUNT\_PARAM\* *pInParam*, NET\_OUT\_QUEYT\_RECORD\_COUNT\_PARAM\* *pOutParam*, int *waittime*=1000);

Port note：get record number，search handle as CLIENT\_FindRecord port to get.

Note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] pInParam Search record parameter，see NET\_IN\_QUEYT\_RECORD\_COUNT\_PARAM

[out] pOutParam Retuen record number，see NET\_OUT\_QUEYT\_RECORD\_COUNT\_PARAM

[in] waittime and waiting overtime

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

Input parameter：

typedef struct \_NET\_IN\_QUEYT\_RECORD\_COUNT\_PARAM

{

DWORD dwSize;

LLONG lFindeHandle; // Search handle

} NET\_IN\_QUEYT\_RECORD\_COUNT\_PARAM;

Output parameter:

typedef struct \_NET\_OUT\_QUEYT\_RECORD\_COUNT\_PARAM

{

DWORD dwSize;

int nRecordCount; //Device return record item

} NET\_OUT\_QUEYT\_RECORD\_COUNT\_PARAM;

BOOL **CLIENT\_FindRecordClose**(LLONG *lFindHandle*);

Port：end search.

Parameter note：

[in] lFindHandle CLIENT\_FindRecord () return value

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

1. **Alarm Record(Log Search)**

BOOL **CLIENT\_QueryDeviceLog**(LLONG *lLoginID*, QUERY\_DEVICE\_LOG\_PARAM\* *pQueryParam*, char \**pLogBuffer*, int *nLogBufferLen*, int\* *pRecLogNum*, int *waittime*=3000);

Port note：Search device log，search by page.

Parameter note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] pQueryParam Search log parameter，see QUERY\_DEVICE\_LOG\_PARAM

[out] pLogBuffer Return log buffer，Return log is one [DH\_DEVICE\_LOG\_ITEM\_EX](mk:@MSITStore:D:\NETSDK\Tools\doc\网络sdk开发手册.chm::/大华网络skd开发手册/数据结构定义/DH_DEVICE_LOG_ITEM_EX.htm) structure

[in] nLogBufferLen pLogBuffer buffer size

[out] pRecLogNum Return to received log item

[in] waittime and waiting overtime

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

Input parameter：

typedef struct \_QUERY\_DEVICE\_LOG\_PARAM

{

DH\_LOG\_QUERY\_TYPE emLogType; // Search log type

NET\_TIME stuStartTime; // Search log start time

NET\_TIME stuEndTime; // Search log end time

int nStartNum; // Search starts from which log, first search can set to 0

int nEndNum; // Search until which log, search ends, log return item max is 1024

BYTE nLogStuType; // Log data structure type,0:means DH\_DEVICE\_LOG\_ITEM；1:表示DH\_DEVICE\_LOG\_ITEM\_EX

BYTE reserved[3]; // Reserved aligned

unsigned int nChannelID; // Chanel no. 0: compatible with all channels，so channel no. strarts from 1; 1: first channel

BYTE bReserved[40];

} QUERY\_DEVICE\_LOG\_PARAM;

Output parameter:

// New log info structure，corresponding port CLIENT\_QueryDeviceLog port

typedef struct \_DH\_DEVICE\_LOG\_ITEM\_EX

{

int nLogType; // Log type

DHDEVTIME stuOperateTime; // Date

char szOperator[16]; // Operator

BYTE bReserved[3];

BYTE bUnionType; // union structure type ,0:szLogContext, 1:stuOldLog.

union

{

char szLogContext[64]; // Log note info

struct

{

DH\_LOG\_ITEM stuLog; // Old log structure

BYTE bReserved[48]; // Reserve

}stuOldLog;

};

char szOperation[32]; // Detailed operation content

char szDetailContext[4\*1024]; // Detailed log info description

} DH\_DEVICE\_LOG\_ITEM\_EX, \*LPDH\_DEVICE\_LOG\_ITEM\_EX;

BOOL **CLIENT\_QueryDevLogCount**(LLONG *lLoginID*, NET\_IN\_GETCOUNT\_LOG\_PARAM \**pInParam*, NET\_OUT\_GETCOUNT\_LOG\_PARAM\* *pOutParam* ,, int *waittime*=3000);

Port note：search device log item.

Note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] pInParam Search log parameter，see NET\_IN\_GETCOUNT\_LOG\_PARAM

[out] pOutParam Return log item，see NET\_OUT\_GETCOUNT\_LOG\_PARAM

[in] waittime and waiting overtime

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..

Input parameter：

typedef struct tagNET\_IN\_GETCOUNT\_LOG\_PARAM

{

DWORD dwSize;

QUERY\_DEVICE\_LOG\_PARAM stuQueryCondition; // Search record item filter

} QUERY\_DEVICE\_LOG\_PARAM;

typedef struct \_QUERY\_DEVICE\_LOG\_PARAM

{

DH\_LOG\_QUERY\_TYPE emLogType; // Search log type

NET\_TIME stuStartTime; // Search log start time

NET\_TIME stuEndTime; // Search log end time

int nStartNum; // In period search starts from which record, first search can set to 0

int nEndNum; // During s search, search which item,一次log ends. Log return item max is 1024

BYTE nLogStuType; // Log data structure type ,0: means DH\_DEVICE\_LOG\_ITEM；1: means DH\_DEVICE\_LOG\_ITEM\_EX

BYTE reserved[3]; // Reserved aligned

unsigned int nChannelID; // Channel no. ,0: compatible with previous all channls，so channel no. starts from 1 ; 1: first channel

BYTE bReserved[40];

} QUERY\_DEVICE\_LOG\_PARAM;

Output parameter:

typedef struct tagNET\_OUT\_GETCOUNT\_LOG\_PARAM

{

DWORD dwSize;

int nLogCount; // Log no.(device return)

} NET\_OUT\_GETCOUNT\_LOG\_PARAM;

1. **A&C Status**

BOOL **CLIENT\_QueryDevState**(LLONG lLoginID, int nType, char\* pBuf, int nBufLen, int\* pRetLen, int waittime=1000);

Port note：search device status port

Note：

[in] lLoginID CLIENT\_Login()return value，when it is 0, it means abnormal login

[in] nType Search info type

[out] pBuf Output parameter，used to get search return data buffer, according to different search types, returned data structures are different, see

|  |  |
| --- | --- |
| Search Info Type | Corresponding Structure |
| DH\_DEVSTATE\_DOOR\_STATE | Search A&C status(corresponding structure NET\_DOOR\_STATUS\_INFO) |

[in] nBufLen Buffer length，unit is byte

[out] pRetLen Output parameter，actual return data length，unit is byte

[in] waittime Search status and waiting time，default 1000ms

Return value：successful return TRUE，failed return FALSE. Port return please call CLIENT\_GetLastError() to get error code，find error cause via error code..