UDA5 Net5 API Reference

Contents

1. About Net5 API

Net5 API indicates an API set designed to access and manage a network device among UDA5 APIs. UDA5 refers to API version 5 among the various APIs we provide. UDA5 consists of Cap5, Aud5, Cod5, Ovr5, and Net5 API sets. For further information concerning each API set except for Net5 API, refer to the SDK Manual for UDA5.

The state of a local device driver is indicated as Init State, Setup State, or Run State. A network device contains an additional two states, Disconnected State and Connected State. A SDK user of a network device can access a network device by using a Net5Connection API. If access is successful, the SDK state will be changed from Disconnected state to Connected state. The user can also disconnect a network device by using a Net5Disconnection API. If the network device is disconnected, the SDK state will be changed from Connected state to Disconnected state.

Network devices in Connected state support various state control APIs including Setup, Run, Stop and Endup of Aud5, Cap5 and Cod5. When one of these state control APIs is called up, the driver will be in Init State, Setup State, or Run State. For additional information regarding local device state, refer to the UDA5 SDK Manual.

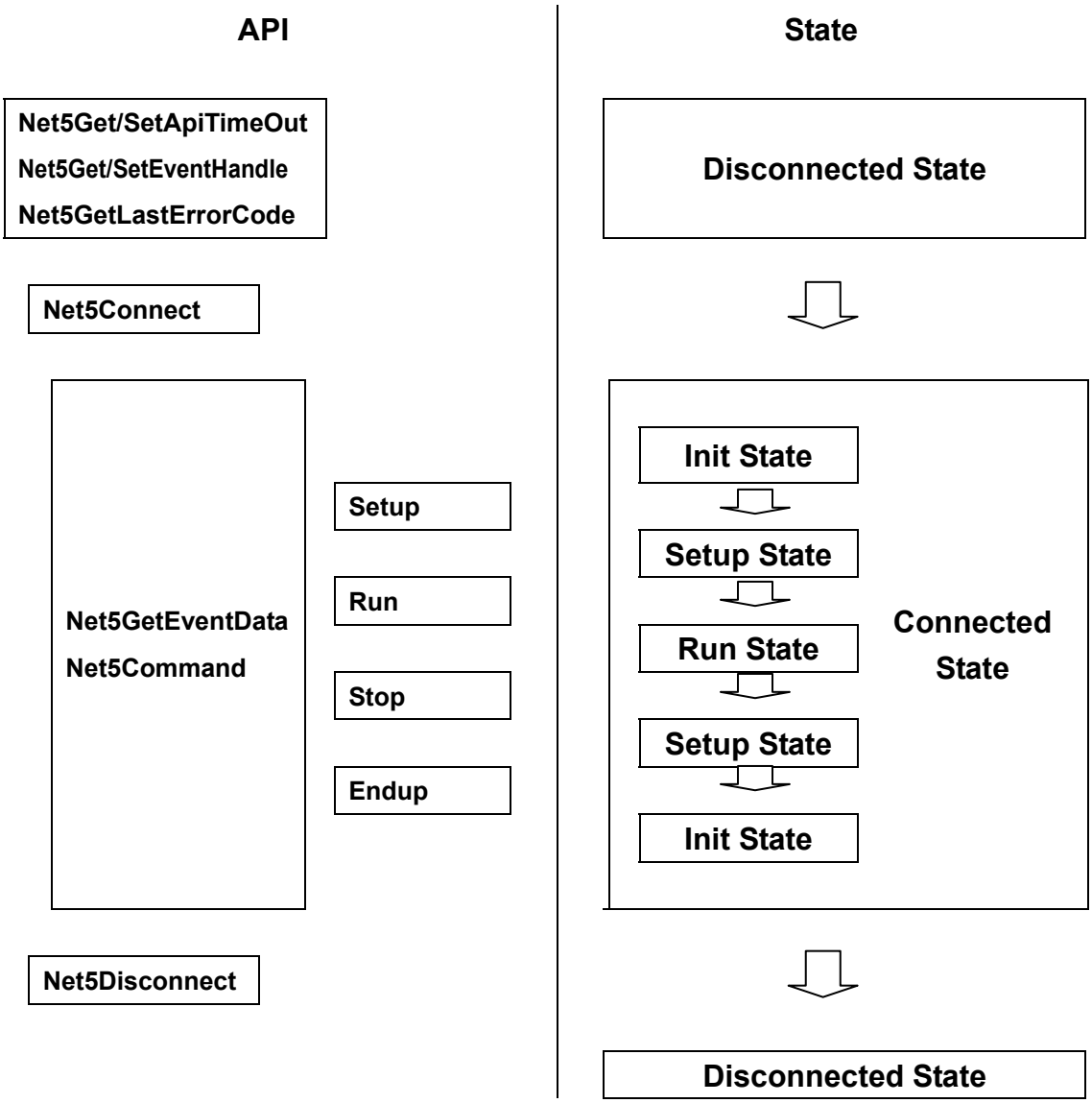

Figure. Network Device State Control

2. Using Net5 SDK

2.1 Connection

With the Net5Connect API a user can hook up to a network device or directory server. If attempts at access fail, an APP developer can take relevant action, for example, by displaying a Connection failed message. For details pertaining to Net5Connect, refer to Chapter 3. Net5 API Reference.

```
class CCardManager{
private:
```

```
    // Generate an interface-typed pointer to be retrieved
    INet5      *m_pNet5;
    ICod5      *m_pCod5;
    …
};
void CCardManager::Init()
{
    IUnknown       * pUnknown;
    // Generate an IUnKnown-typed instance by using Cmn5CreateInstance.
    Cmn5CreateInstance((void **) & pUnknown);

    // Retrieve an interface pointer by using the IUnKnown-typed instance
generated above.
    pUnknown->QueryInterface(IID_INet5, (PVOID*) & m_pNet5);

    // Release the instance no longer employed.
    pUnknown-> Release();

    // Access to a network device.
    if(!m_pNet5->Net5Connect(pSvrIpAddr, pUserId, pPassword,
                &uServerType, &m_uAuthorityData, 10000)) {
        MessageBox(NULL, "Connection Fail", "Error", MB_OK);
    }
}
```

2.2 Net5 Event Processing

In case the network driver is disconnected or another user connected to the network device changes Codec/Video/Audio properties of the device, the SDK sets an event. In order to receive such event information, a user must register an event handle to the SDK by using the Net5SetEventHandle or Net5GetEventHandle API. Then, using the Net5GetEventData the user can check event information API when an event occurs. If the user knows what kind of event occurred, he can also know why it occurred by using the UDA5 API. For example, if a NET5_NDT_CONNECTION event occurred, using a NetCommand named NET5_NC_IS_CONNECTION a user can determine whether the event happened as connecting to a network device or as disconnecting from it. The following example illustrates how to retrieve an event related to the Net5.

```
DWORD WINAPI NetEventThreadProc(LPVOID pParam)
{
    if(bSetEventType) {
        m_hNetEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
        m_pNet5->Net5SetEventHandle(CMN5_DT_NET, &m_hNetEvent);
    } else {
        m_pNet5->Net5GetEventHandle(CMN5_DT_NET, &m_hNetEvent);
    }
    while(bRun) {
```

```
        dResult = WaitForSingleObject(m_hNetEvent, INFINITE);
        if(dResult == WAIT_OBJECT_0) {
            Net5DataProc();
        }
    }
}

void Net5DataProc()
{
    NET5_EVENT_DATA NetEventData;
    m_pNet5->Net5GetEventData(CMN5_DT_NET, (PVOID) &NetEventData);

    switch(NetEventData.uNetDataType) {
    case NET5_NDT_CONNECTION:
    {
        BOOL bIsConnection;
        m_pNet5->Net5Command(NET5_NC_IS_CONNECTION,
                    (ULONG*)bIsConnection, NULL, NULL,
                    NULL);
        if(!bIsConnection) {
            DisconnectProc();
        }
        break;
    }
    case NET5_NDT_CHANGE_PROPERTY:
    {
        ULONG uCmdNum = 4;
        CMN5_MULTI_COMMAND CodCmd[4];
        CodCmd[0].uCommand = COD5_CPC_VIDEO_FORMAT;
        CodCmd[1].uCommand = COD5_CPC_VIDEO_IMAGE_SIZE;
        CodCmd[2].uCommand = COD5_CPC_SKIP_FRAME;
        CodCmd[3].uCommand = COD5_CPC_GOP_SIZE;
        for(int i=0; i<4; i++) {
            CodCmd[i].uBoard = 0;
            CodCmd[i].uChannel = 0;
        }
        m_pCod5->Cod5GetMultiCodecProperties(uCmdNum, CodCmd);
        ChangeCodProperty(CodCmd);
        break;
    }
    case NET5_NDT_RECV_SERIAL_DATA:
    {
        ULONG uPortNumber, uDataSize;
        m_pNet5->Net5Command(NET5_NC_RECV_SERIAL_DATA, &uPortNumber,
                    puSerialData, uMaxSize, &uDataSize);
        SerialDataProc(uPortNumber, puSerialData, uDataSize);
        break;
    }
    case NET5_NDT_UNSTABLE_CONNECTION:
        ShowMessage("Unstable connection");
```

```
        break;
    default:
        return;
    }
}
```

2.3 End Connection

This is the opposite process to that explained in Section 3.1 Connection. Using a Net5Disconnect a user can end the connection with a network.

```
m_pNet5Api->Net5Disconnect();
```

3. Net5 API Reference

3.1 Functions

Functions	Connection State		State Change
	Disconnected	Connected	
Net5GetLastErrorCode	O	O	
Net5Connect	O		Connected
Net5Disconnect		O	Disconnected
Net5Command		O	
Net5SetApiTimeout	O	O	
Net5GetApiTimeout	O	O	
Net5SetEventHandle	O	O	
Net5GetEventHandle	O	O	
Net5GetEventData	O	O	
Net5ConnectEx	O		
Net5Login		O	
Net5Logout		O	

Net5GetLastErrorCode

The Net5GetLastErrorCode function retrieves the Net5 API's last error code and additional information. The API's error codes are maintained on the stack structure. This function provides not only the last error code but also the internal error code information.

```
BOOL Net5GetLastErrorCode(
    CMN5_ERROR_CODE_ITEM *pEcode      //
);
```

Parameters

pEcode

[out] The pointer to a variable of CMN5_ERROR_CODE_ITEM type to retrieve the error information.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails or no more error is in the error stack, it returns a zero value.

Remarks

When the function fails by calling the Net5 API, call Net5GetLastErrorCode to obtain information related to the error. CMN5_ERROR_CODE_ITEM structure includes the error code, error time, error display and error occurring session. This function, unlike all the other Net5 APIs, does not require calling Net5GetLastErrorCode if the return is FALSE. It simply indicates that there is no more error.

Unlike GetLastError of the Win32 API, this function continually saves error information in stack type. Therefore, there is no immediate and compulsory need to call up this function. Besides, it does not operate a separate stack for each thread. Since this function internally saves the error information in stack type, the complete information of the internally occurred error – up to SYS level – can be verified when calling of any API fails.

Msg can be of great help with debugging. It is a programming option used to specify whether original messages will be exposed to users or not, but it is not recommended since there is no currently existing multilingual version driver.

Connection State

Disconnected, Connected

Example Code

```
BOOL rs;
rs=Net5SetApiTimeOut(30);
if(!rs){
    CMN5_ERROR_CODE_ITEM errCode;
    Net5GetLasError(&errCode);
    TRACE("Error-%s\n", errCode. AuxMsg);
}
```

See Also

UDA5 SDK Manual

Net5Connect

This function connects to a video server.

```
BOOL Net5Connect(
    LPCSTR sAddress       //
    LPCSTR sUserID        //
        LPCSTR sUserPW        //
        ULONG  uServerType    //
        ULONG  *puIdData      //
        ULONG  uTimeOut       //
);
```

Parameters

sAddress

[in] IP address of the server to be connected. Enter in <IP Address:Port> form.

sUserID

[in] ID of the user to be connected.

sUserPass

[in] Pass of the user to be connected.

uServerType

[in] The supported values are as the below.

NET5_ST_VIDEO_SERVER : When NVS400 uses static IP address, specify this value. A client will connect NVS400 directly.

NET5_ST_DIRECTORY_SERVER : When NVS400 uses dynamic IP address, specify this value. A client will connect NVS400 via Directory server.

NET5_ST_MULTICAST : A client gets media data when multicast. Please note that a client should use static IP address.

puIdData

[out] Retrieves the ID data of the connected user. This ID data may be used for authorization or other purposes.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

To use a video server, a user must connect to the server. If access is successful, he can utilize the video server. The access authority of a user can be set by using his ID data. If a fixed IP is employed, a direct connection will be made to the server. For a dynamic IP, however, the connection will be made via a server, whose address, ID, and password must be registered and whose server type must be set as directory server in advance.

Connection State

Disconnected -> Connected

Example Code

See Also

Net5Disconnect

This function disconnects the video server.

```
BOOL Net5Disconnect(
        \\
);
```

Parameters

This function has no parameters.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

This function disconnects the currently accessed video server.

Connection State

Connected -> Disconnected

Example Code

See Also

Net5Command

This function is used to manage the system and users of a video server.

```
BOOL Net5Command(
    ULONG   uVideoServerCmd        //
        ULONG   *puParam1          //
        ULONG   *puParam2          //
        ULONG   *puParam3          //
        ULONG   *puParam4          //
);
```

Parameters

uVideoServerCmd

[in] Command type to be used.

Parameter	Description	Parameter
Admin Level – User Managerment		
NET5_NC_ADD_USER_ID	Add a user.	*puParam1*[in] is a user ID to be accessed. *puParam2*[in] is a user password. *puParam3*[in] is user data related to ID information.
NET5_NC_DEL_USER_ID	Delete a user.	*puParam1*[in] is a user ID.
NET5_NC_GET_USER_PW	Retrieve user password.	*puParam1*[in] is a user ID. *puParam2*[out] is a user password.
NET5_NC_GET_ALL_USER_LIST	Retrieve a list of all the registered users.	*puParam1*[out] is the array of character string with the user list.
NET5_NC_GET_CONNECTION_NUM	Retrieve the number of currently accessed users.	*puParam1*[out] is the number of currently accessed users.
NET5_NC_GET_CONNECTION_LIST	Retrieve the list of currently accessed users.	*puParam1*[out] is the array of character string with the list of currently accessed users.
NET5_NC_CHANGE_USER_ID_DATA	Change user ID data.	*puParam1*[in] is a user ID. *puParam2*[in] is a user ID to be changed.
Admin Level – System Managerment		

NET5_NC_SET_ADDR	Set server address.	*puParam1*[in] is a structure variable of NET5_ADDR_INFO.
NET5_NC_SET_PORT	Set server port.	*puParam1*[in] is a Web server port. *puParam2*[in] is a command port. *puParam3*[in] is a streaming port.
NET5_NC_SET_SYSTEM_NAME	Set server name.	*puParam1*[in] is the system name.
NET5_NC_SET_SYSTEM_TIME	Set server time.	*puParam1*[in] is a structure variable of SYSTEMTIME, which is set by system time.
NET5_NC_FIRMWARE_UPDATE	Update firmware.	*puParam1*[in] is a file path of the firmware to be updated.
NET5_NC_RESTART	Restart server.	None
NET5_NC_SET_SNTP_ADDRESS	Set SNTP addresses	*puParam1[in]* is first SNTP address. *puParam2[in]* is second SNTP address. *puParam3[in]* is third SNTP address.
NET5_NC_FACTORY_DEFAULT	설정을 초기화한다.	*puParam1[in]* 는 factory default의 level이다. 1이면 모든 설정을 초기화하고 2이면 일부 설정만 초기화한다. (제품에 따라 초기화하는 는 값이 다르다.)
NET5_NC_SET_IP_ADDR_CONFIG	주소의 유형을 설정한다.	*puParam1[in]* 는 다음 두가지 값을 설정할 수 있다. NET5_IAC_STATIC : 고정 IP를 사용한다. NET5_IAC_DHCP : 유동 IP를 사용한다.
User Level – ID Info		
NET5_NC_CHANGE_PW	Change user password.	*puParam1*[in] is a user ID. *puParam2*[in] is a user password to be changed.
NET5_NC_GET_ID_DATA	Retrieve user ID data.	*puParam1*[in] is a user ID. *puParam2*[out] is a user ID to be changed.
User Level – System Info		
NET5_NC_GET_SERVER_ADDR	Retrieve server address.	*puParam1*[out] is a structure variable of NET5_ADDR_INFO.
NET5_NC_GET_SYSTEM_NAME	Retrieve server name.	*puParam1*[out] is the server name.
NET5_NC_GET_SYSTEM_TIME	Retrieve server time.	*puParam1*[out] is a structure variable of SYSTEMTIME. Since it is retrieved by system

		time, the time zone must be set to local time to display in local time.
NET5_NC_IS_CONNECTION	Verify current connection state.	*puParam1*[out] is a variable to verify connection state. 1 indicates connected state while 0 indicates disconnected state.
NET5_NC_GET_PORT	Verify server port	*puParam1*[in] is a Web server port. *puParam2*[in] is a command port. *puParam3*[in] is a streaming port.
NET5_NC_SEND_SERIAL_DATA	Send serial data.	*puParam1*[in] is a port number. The port number must be set to 0 to send data to COM1. *puParam2*[in] is the data to be sent. *puParam3*[in] is the size of the data to be set.
NET5_NC_RECV_SERIAL_DATA	Retrieve serial data.	*puParam1*[out] is the port number where the serial data event to be retrieved occurred. *puParam2*[out] is the serial data to be retrieved. *puParam3*[in] is the maximum size of serial data. *puParam*4[out] is the actual size of serial data.
NET5_NC_GET_COMM_STATE	Configure serial communication device.	*puParam1*[in] is port number. *puParam2*[in] is pointer of DCB structure.
NET5_NC_GET_COMM_STATE	Get configuration of serial communication device.	*puParam1*[in] is port number. *puParam2*[in] is pointer of DCB structure.
NET5_NC_GET_VERSION	Get version infomation of server and client.	*puParam1[in]* is the pointer of structure that includes explanation of NET5_VERSION_INFO *puParam2[out]* is pointer of NET5_VERSION_INFO structure.
NET5_NC_GET_LOG	Get log infomation of server.	*puParam1[in]* is pointer of CMN5_ERROR_CODE_ITEM structure.
NET5_NC_GET_SNTP_ADDRESS	Get SNTP addresses	*puParam1[out]* is first SNTP address. *puParam1[out]* is second SNTP address. *puParam1[out]* is third SNTP address.
NET5_NC_GET_IP_ADDR_CONFIG	설정된 주소 유형을 가져온다.	*puParam1[out]* 는 설정된 주소 유형이다.
Resource Management		

NET5_NC_AUDIO_TX_LOCK	Lock or unlock a audio resource of server.	[1]*puParam1[in]* is sub command. *puParam2[in]* is audio channel. *puParam3[out]* is status.
Automation		
NET5_NC_SET_EVENT_AUTOMATION	Set event automation information.	*puParam1[in]* is number of event automation information structures. *puParam2[in]* is event automation information structures.
NET5_NC_QUERY_EVENT_AUTOMATION_SIZE	Query size of event automation information structures.	*puParam1[out]* is total size of event automation information structures. *puParam2[out]* is number of event automation information structures.
NET5_NC_GET_ALL_EVENT_AUTOMATION	Get all event automation information structures.	*puParam1[in]* is buffer size to get automation information structures. *puParam2[out]* is event automation information structures.
NET5_NC_WRITE_USER_FILE_DATA	Write user data to file	*puParam1[in]* is file name. *puParam2[in]* is user data to write. *puParam3[in]* is data size. *puParam4[in]* is flag for appending.
NET5_NC_GET_USER_FILE_DATA_SIZE	Get user file size	*puParam1[in]* is file name. *puParam2[out]* is data size.
NET5_NC_READ_USER_FILE_DATA	Read user data from file	*puParam1[in]* is file name. *puParam2[out]* is user data to read. *puParam3[in]* is data size.
NET5_NC_GET_MULTICAST_INFO	Set multicast information	*puParam1[in]* is MULTICAST_INFO structure value.
NET5_NC_SET_MULTICAST_INFO	Get multicast information	*puParam1[out]* is MULTICAST_INFO structure value.
NET5_NC_ENABLE_MULTICAST	Enable or disable multicast	*puParam1[in]* is boolean value for enable or disable multicast.
RTSP command		
NET5_NC_SEND_RTSP_REQUEST	RTSP command를 보낸다.	*puParam1[in]* 는 요청할 RTSP command이다..
NET5_NC_RECV_RTSP_	RTSP command의	*puParam1[in]* 는 RTSP command의 응답이다.

RESPONSE	응답을 받는다.	

Note 1. Audio Tx Commands which has four values as the below.
NET5_ATL_LOCK : Lock audio resource of NVS400 to send audio data. If another client locked audio resource, this command returns FALSE.
NET5_ATL_FORCED_LOCK : If another client locked audio resource, unlock current client audio resource and then lock audio resource for new client.
NET5_ATL_UNLOCK : Unlock audio resource.
NET5_ATL_GET_STATUS : Get status of audio resource. (1 is locked, 0 is not locked)

puParam1

Additional parameter, whose meaning varies according to *uVideoServerCmd*

puParam2

Additional parameter, whose meaning varies according to *uVideoServerCmd.*

puParam3

Additional parameter, whose meaning varies according to *uVideoServerCmd.*

puParam4

Additional parameter, whose meaning varies according to *uVideoServerCmd.*

Return Values

If the function succeeds, the return value is nonzero.
If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

Net5Command is an API that allows management of the system and users as a whole. puParam1 to puParam4 are defined by uVideoServerCmd. Enter NULL for unnecessary value, and it will be ignored.

Connection State

Connected

Example Code

See Also

NET5_ADDR_INFO

Net5SetApiTimeout

This function sets the timeout for server response.

```
BOOL Net5SetApiTimeout(
    ULONG   uTimeout        //
);
```

Parameters

uTimeout

[in] Set the timeout for server response as seconds.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

When the connection with a server is abnormally disconnected, the client cannot notice such disconnection. With the timeout set, the client assumes that the connection with the server is over if there is no response for that time, and it forcefully ends the connection.

Connection State

Disconnected, Connected

Example Code

See Also

Net5GetApiTimeout

This function retrieves the timeout for server response.

```
BOOL Net5GetApiTimeout(
    ULONG   *uTimeout           \\
);
```

Parameters

uTimeout

[out] Retrieves the timeout for server response as seconds.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

Connection State

Disconnected, Connected

Example Code

See Also

Net5SetEventHandle

Using this function, a user can set an event as desired.

```
BOOL Net5GetEventHandle(
    ULONG   uType           \\
    HANDLE  hHandle         \\
);
```

Parameters

uType

[in] The type of event to be set. Use enum DataType. CMN5_DT_NET is available.

hHandle

[in] The handle of event to be set.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

As a single client accesses increasingly more servers, it becomes more difficult to properly handle the events of all servers only using the internal events. It may also be necessary to add events for other purposes besides those internally set. In such cases, a user can set an event as desired. Only CMN5_DT_NET typed events are currently supported.

Connection State

Disconnected, Connected

Example Code

See Also

Cap5SetEventHandle, Cod5SetEventHandle, Aud5SetEventHandle

NEt5GetEventHandle

This function retrieves the event handles of a data type specified for Net5.

```
BOOL Cap5GetEventHandle(
    ULONG uType,           //
    HANDLE *pHandle        //
);
```

Parameters

uType

[in] The type of event to be retrieved. Use enum DataType. CMN5_DT_NET is available.

pHandle

[out] Pointer to the variable that will save the handle values of the retrieved event.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero. To get extended error information, call up Cap5GetLastErrorCode.

Remarks

This function retrieves event handles. The function is available for the events of audio reset type and signals when related data are generated or their status is changed. Do not close the event handle in the APP. Only CMN5_DT_NET typed events are currently supported.

Connection State

Disconnected, Connected

Example Code

See Also

Cap5GetEventHandle, Cod5GetEventHandle, Aud5GetEventHandle

Net5GetEventData

Retrieves data of a specific type from the data generated in the Net5.

```
BOOL Net5GetEventData(
    ULONG uType,        //
    void *pData     //
);
```

Parameters

uType

[in] The type of event to be retrieved. Use enum DataType. CMN5_DT_NET is available.

pData

[out] Pointer to the variable that will receive information about the data starting with CMN5_DATA_INFO_HEADER structure type (or data itself). A proper structure to match uType must be transmitted as follows.

uType	pData
CMN5_DT_NET	NET5_EVENT_DATA*

Using uNetDataType among NET5_EVENT_DATA structure variables, a user can verify data type. Each data type has an enum value as shown in the following table.

uNetDataType	Enum Value	Description
CMN5_DT_NET	NET5_NDT_CONNECTION	Events occurring when the connection has ended.
CMN5_DT_NET	NET5_NDT_CHANGE_PROPERTY	Events occurring when another user has changed server settings.
CMN5_DT_NET	NET5_NDT_RECV_SERIAL_DATA	Events occurring when serial data is found.
CMN5_DT_NET	NET5_NDT_UNSTABLE_CONNECTION	Events occurring when the connection is unstable.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

This function is mainly used to retrieve the data of various API sets simultaneously. Retrieve an event using the GetEventHandle function, check that the event has signaled by using the Wait function of Win32, and call up Net5GetEventData.

It is possible to call up the Net5GetEventData function without checking the event signal but it may have no significance. (It is not internally prohibited to call up the function.) Only CMN5_DT_NET typed events are currently supported.

Connection State

Disconnected, Connected

Example Code

```
DWORD NetThreadProcedure()
{
    BOOL retVal;
    HANDLE events[2];

    events[0] = hStopCaptureEvent;
    Net5GetEventHandle(CMN5_DT_NET,&events[1]);

    while(1) {
        DWORD obj = WaitForMultipleObjects(2, events, FALSE, INFINITE);
        switch(obj){
        case WAIT_OBJECT+1: {
            NET5_EVENT_DATA    data = {0,};
            if(m_pNet5Api->Net5GetEventData(CMN5_DT_NET, &data)){
                switch(data.uNetDataType){
                case NET5_NDT_CONNECTION: {
                    TRACE("Disconnect event detected \n");
                    break; }
                case NET5_NDT_CHANGE_PROPERTY:
                    TRACE("Change property event detected \n");
                    break;
                case NET5_NDT_RECV_SERIAL_DATA: {
                    TRACE("Serial event detected\n");
                    ULONG port, size, maxSize = 1024;
                    BYTE serial[1024];
                    m_pNet5Api->Net5Command(NET5_NC_RECV_SERIAL_DATA, &port,
                        (ULONG*)serial, &maxSize, &size);
                    }
                    break;
                }
            }
            break;
```

```
        }
        default:
            break;
        }
    }
}
```

See Also

Cap5GetEventHandle, Cod5GetEventHandle, Cap5GetEventData, Cod5GetEventData, WaitForMultipleObjects, NET5_EVENT_DATA

Net5ConnectEx

This function connects to a video server.

```
BOOL Net5ConnectEx(
    LPCSTR pszAddress,
    ULONG  uStreamingType,
    ULONG uTimeout
);
```

Parameters

pszAddress

[in] IP address of the server to be connected. Enter in <rtsp://IP Address:Port> form.

uStreamingType

[in] The supported values are as the below.

NET5_ST_RTP_UDP : requests RTP data through UDP.

NET5_ST_RTP_TCP : requests RTP data through TCP with the port number of RTSP.

uTimeout

[in] The time to try to connect to the video server. A unit is second. If the connecting is fail during *uTimeOut*, ***Net5ConnectEx*** returns FALSE.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

This API is supported only on NVE series and IPC series

Connection State

Disconnected -> Connected

Example Code

See Also

Net5Connect, Net5Login, Net5Logout

Net5Login

It is for log-in to the video server.

```
BOOL Net5Login(
    LPCSTR pszUserID,
    LPCSTR pszUserPW,
);
```

Parameters

pszUserID

[in] ID of the user.

pszUserPass

[in] Pass of the user.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

This API is supported only on NVE series and IPC series

Connection State

Example Code

See Also

Net5ConnectEx, Net5Logout

Net5Logout

It is for log-off from the video server.

```
BOOL Net5Logout(
);
```

Parameters

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call up Net5GetLastErrorCode.

Remarks

This API is supported only on NVE series and IPC series

Connection State

Example Code

See Also

Net5ConnectEx, Net5Login

3.2 Structures

NET5_EVENT_DATA

This structure shows information about the event data generated by a Net5 API.

```
typedef struct {
    ULONG          uDataType;
    ULONG          uStructSize;
    ULONG          uErrCode;

    ULONG          uNetDataType;
} NET5_EVENT_DATA;
```

Members

uDataType

uStructSize

uErrCode

uNetDataType

Data type of event

Remarks

NET5_ADDR_INFO

This structure shows the server address information.

```
typedef struct {
    char    Address[NET5_MAX_ADDR_LENGTH];
    char    SubnetMask[NET5_MAX_ADDR_LENGTH];
    char    Gateway[NET5_MAX_ADDR_LENGTH];
    char    DNS[NET5_NUM_OF_DNS][NET5_MAX_ADDR_LENGTH];
} NET5_ADDR_INFO;
```

Members

Address

IP address of server

SubnetMask

Gateway

DNS

Remarks

NET5_VERSION_INFO

This structure shows the version information of server and client.

```
typedef union {
    struct {
        ULONG uCurServerVer;          // Current server version
        ULONG uMinRequiredClientVer;  // Minimum required client version
        ULONG uCurClientVer;          // Current client version
        ULONG uMinRequiredServerVer;  // Minimum required server version
        ULONG uMatchInfo;             // Version match information
    };
    ULONG reserved[CMN5_MAX_RESERVED_BOARD_INFO];
} NET5_VERSION_INFO;
```

Members

uCurServerVer

Current server version.

uMinRequiredClientVer

Minimum required client version.

uCurClientVer

Current client version.

uMinRequiredServerVer

Minimum required server version.

uMatchInfo

Version match information. It has three values as the below.

NET5_MI_UPDATE_NOT_REQUIRED : Update is not required.

NET5_MI_REQUIRE_SERVER_UPDATE : Server update is required.

NET5_MI_REQUIRE_CLIENT_UPDATE : Client update is required.

Remarks

NET5_EVENT_AUTOMATION_INFO_HEADER

This structure shows common information of event automation.

```
#define _NET5_EVENT_AUTOMATION_INFO_HEADER_BODY            \
    union {                                                \
        ULONG allHdr[NET5_MAX_RESERVED_AUTOMATION_INFO];\
        struct {                                           \
            GUID    uID;                                   \
            ULONG   uAutomationType;                       \
            ULONG   uStructSize;                           \
            ULONG   uCommand;                              \
        };                                                 \
    }                                                      \

typedef struct _NET5_EVENT_AUTOMATION_INFO_HEADER {
    _NET5_EVENT_AUTOMATION_INFO_HEADER_BODY;
} NET5_EVENT_AUTOMATION_INFO_HEADER;
```

Members

uID

Unique ID of event automation information. This value is set internally.

uAutomationType

Type of event automation which has one value as the below.

NET5_EAT_DDNS : Event automation type is DDNS.

uStructSize

Size of event automation information structure.

uCommand

Event automation command whitch has three values as the below.

NET5_EAC_ADD : add new event automation information.

NET5_EAC_DELETE : delete event automation information.

NET5_EAC_MODIFY : modify event automation information.

Remarks

Event automation is function that NVS400 executes the specified action on the event occurred. There are various types of event automation. Event automation structure has member variables of 'NET5_EVENT_AUTOMATION_INFO_HEADER' for classifying event automation.

NET5_EVENT_AUTOMATION_INFO_DDNS

This structure is used for setting/getting event automation information of DDNS.

```
typedef struct _NET5_EVENT_AUTOMATION_INFO_DDNS {
    _NET5_EVENT_AUTOMATION_INFO_HEADER_BODY;
    union {
        ULONG all[NET5_MAX_RESERVED_AUTOMATION_INFO*8];
        struct {
            ULONG   uDdnsServiceType;
            char    DdnsServerAddr[NET5_MAX_DOMAIN_NAME_LENGTH];
            char    DdnsUserName[NET5_MAX_ID_LENGTH];
            char    DdnsPassword[NET5_MAX_PW_LENGTH];
            char    DomainName[NET5_MAX_DOMAIN_NAME_LENGTH];
            ULONG   uUpdateIntervalMin;
        };
    };
} NET5_EVENT_AUTOMATION_INFO_DDNS;
```

Members

uDdnsServiceType

This member variable indicates DDNS server type.

It can have only NET5_DDNS_SERVICE_TYPE_DYNDNS.

DdnsServerAddr

This member variable indicates DDNS server address.

It can have both IP address and domain name address.

DdnsUserName

This member variable indicates the user ID for DDNS server.

DdnsPassword

This member variable indicates the password of DDNS user ID.

DomainName

This member variable indicates the domain name address which is possessed by NVS400.

uUpdateIntervalMin

This member variable indicates the update period of the IP address information in DDNS server

Remarks

The usage of this structure can be differed by each DDNS server types. Refer to the additional manual for using the each DDNS server.

NET5_MULTICAST_INFO

This structure is used for setting/getting information of multicast.

```
typedef union {
    struct {
        char Address[NET5_MAX_DOMAIN_NAME_LENGTH];
        ULONG uPort;
        ULONG uTTL;
    };
    ULONG reserved[NET5_MAX_RESERVED_MULTICAST_INFO];
} NET5_MULTICAST_INFO;
```

Members

Address

Specifies multicast address.

Address range is 224.0.0.0～239.255.255.255.

uPort

Specifies multicast port.

uTTL

Specifies time-to-live value.

Remarks

None

Revision history

Revision	Date	Description
A	-	Created
B	2006-2-06	User file and Multicast are added.
C	2007-04-23	Added Net5ConnectEx Added Net5Login Added Net5Logout