UDA5 NVE Series
SDK Manual

Table of Contents

1. Overview

1.1. Introduction

NVE is a Network Video Server that compresses the video data using the MPEG-4/MPEG-2/MJPEG method and transmits it to the network in real time. In addition to 4 channel video MPEG-4/MPEG-2/MJPEG compression and transmission, it also supports 4 channel PCM audio capture and up to 4 sets of DI/DO.

Composition	Single
Video Channel	4
Audio Channel	4
Digital Input Sensor	4
Digital Output	4

1.1.1. Relevant platform and operating system

The provided SDK Package supports Microsoft Windows2000, and Microsoft Windows XP, running on Intel Microprocessors (Celeron, Pentium Ⅲ, Pentium Ⅳ). The provided drivers, DLLs, and sample program are compiled with Microsoft SDK and Microsoft DDK using Microsoft Visual C/C++ 6.0 compiler.

1.2. Provided Files

1.2.1. Library

UdaNVE.dll
UdaNVE.lib
Cmn5BoardLibEx.h
Cod5BoardLibEx.h
Net5BoardLibEx.h

1.2.2. Sample programs

PrismII

NVE uses the PrismII, which is UDP's integrated APP. For further details, refer to the NVE User's Manual.
Use the API referring to the source code of the MgrNVE (NVE Card Manager).

StartUDA5

This is a simple application for beginner. It describes the basic usage of UDA5.

1.3. Installation

The NVE SDK does not require any particular installation procedure. Just insert the sample program and DLL in the same folder and execute them.

2. Supported API

NVE supports Cod5 and Net5 API. This section describes the limitation on the individual API. Function call method enables Interface method only. WIN32 API method not supported.

NVE supports following API set.

API	Supports
Cap5	X
Aud5	X
Ovr5	X
Cod5	○
Net5	○

2.1. Net5 APIs

Refer to the Net5 API Manual for details of API.

2.1.1. Net5 API Set

API	Supported	Remarks
Net5Connect	X	
Net5ConnectEx	○	
Net5Disconnect	○	
Net5Login	○	
Net5Logout	○	
Net5Command	○	Refer to the Net Command.
Net5SetApiTimeOut	○	
Net5GetApiTimeOut	○	
Net5SetEventHandle	X	
Net5GetEventHandle	○	
Net5GetEventData	○	Refer to the Data Type.
Net5GetLastErrorCode	○	

2.1.2. Net Command

Command	Supported	Remarks
NET5_NC_ADD_USER_ID	○	
NET5_NC_DEL_USER_ID	○	
NET5_NC_GET_USER_PW	○	
NET5_NC_GET_ALL_USER_LIST	○	
NET5_NC_GET_CONNECTION_NUM	○	

NET5_NC_GET_CONNECTION_LIST	○	
NET5_NC_CHANGE_USER_ID_DATA	○	
NET5_NC_SET_ADDR	○	
NET5_NC_SET_PORT	○	
NET5_NC_SET_SYSTEM_NAME	○	
NET5_NC_SET_SYSTEM_TIME	○	
NET5_NC_FIRMWARE_UPDATE	X	
NET5_NC_RESTART	○	
NET5_NC_SET_SNTP_ADDRESS	○	
NET5_NC_FACTORY_DEFAULT	○	
NET5_NC_SET_DHCP	○	
NET5_NC_SET_IP_ADDR_CONFIG	○	
NET5_NC_SET_DIR_SERVER_INFO	○	
NET5_NC_CHANGE_PW	○	
NET5_NC_GET_ID_DATA	○	
NET5_NC_GET_SERVER_ADDR	○	
NET5_NC_GET_SYSTEM_NAME	○	
NET5_NC_GET_SYSTEM_TIME	○	
NET5_NC_IS_CONNECTION	X	
NET5_NC_GET_PORT	○	
NET5_NC_SEND_SERIAL_DATA	○	
NET5_NC_RECV_SERIAL_DATA	○	
NET5_NC_SET_COMM_STATE	○	
NET5_NC_GET_COMM_STATE	○	
NET5_NC_GET_VERSION	○	
NET5_NC_GET_LOG	X	
NET5_NC_GET_SNTP_ADDRESS	○	
NET5_NC_GET_DHCP	○	
NET5_NC_GET_IP_ADDR_CONFIG	○	
NET5_NC_GET_DIR_SERVER_INFO	○	
NET5_NC_AUDIO_TX_LOCK	X	
NET5_NC_SET_EVENT_AUTOMATION	○	
NET5_NC_QUERY_EVENT_AUTOMATION_SIZE	○	
NET5_NC_GET_ALL_EVENT_AUTOMATION	○	
NET5_NC_WRITE_USER_FILE_DATA	X	
NET5_NC_GET_USER_FILE_DATA_SIZE	X	
NET5_NC_READ_USER_FILE_DATA	X	
NET5_NC_GET_MULTICAST_INFO	X	
NET5_NC_SET_MULTICAST_INFO	X	

NET5_NC_ENABLE_MULTICAST	X	
NET5_NC_SEND_RTSP_REQUEST	○	
NET5_NC_SEND_RTSP_RESPONSE	○	

2.1.3. Data Type

It supports NET5_NDT_CHANGE_PROPERTY and NET5_NDT_RECV_SERIAL_DATA.

- NET5_NDT_CHANGE_PROPERTY

If settings are changed, NVE sends an event to the connected clients. The client receives the event that the settings are changed. But the client can not know what are changed. Therefore, if the client receives the event, get all settings of NVE and reset the client settings as changed by NVE (ex. Image size, Video codec).

- NET5_NDT_RECV_SERIAL_DATA

NVE send event data to client when some data comes through RS485 port. In other word, a NVE has an RS485 data that client have to read. Use NET5_NC_RECV_SERIAL_DATA of Net command to get RS485 data form NVE.

2.2. Cod5 APIs

Refer to the UDA5 SDK Manual for details of API.

2.2.1. Cod5 API Set

API	Supported	Remarks
Cod5GetLastErrorCode	○	
Cod5QueryInfo	○	Support CMN5_QIC_GET_PROCESS_STATE or CMN5_QIC_GET_HWINFO_STR
Cod5GetSystemInfo	○	
Cod5GetBoardInfo	○	
Cod5Activate	○	
Cod5Setup	○	
Cod5Endup	○	
Cod5Run	○	
Cod5Stop	○	
Cod5ChannelEnable	○	
Cod5SetCaptureMethod	X	NVE supports event method only.
Cod5SetCallback	X	
Cod5ReleaseData	○	
Cod5SetEventHandle	X	
Cod5GetEventHandle	○	
Cod5GetEventData	○	

Cod5SetEventData	X	
Cod5SetImageSize	X	Use Cod5SetCodecProperty
Cod5GetImageSize	X	Use Cod5GetCodecProperty
Cod5SetVideoFormat	X	Use Cod5SetCodecProperty
Cod5GetVideoFormat	X	Use Cod5GetCodecProperty
Cod5SetCodecProperty	○	Refer to the Codec Property Command.
Cod5GetCodecProperty	○	
Cod5SetCodecAdjust	○	Refer to the Codec Adjust Command.
Cod5GetCodecAdjust	○	
Cod5SetMultiCodecProperties	○	Several codec properties can be set simultaneously.
Cod5GetMultiCodecProperties	○	Several codec properties can be imported simultaneously.
Cod5SetMultiCodecAdjusts	○	Several Codec Adjustment can be set simultaneously.
Cod5GetMultiCodecAdjusts	○	Several Codec Adjustment can be imported simultaneously.
Cod5GetVideoStatus	○	
Cod5SetDO	○	
Cod5GetDO	○	
Cod5GetDI	○	
Cod5SetExtVideoOut	○	Refer to the 3.4 External Video Out
Cod5GetExtVideoOut	○	
Cod5SetWatchdog	○	Refer to the Watchdog Command.
Cod5I2CTransfer	X	
Cod5WriteUserEEPROM	○	
Cod5ReadUserEEPROM	○	

2.2.2. Codec Property Command

Command	Supported	Remarks
COD5_CPC_VIDEO_FORMAT	○	NTSC/PAL
COD5_CPC_VIDEO_IMAGE_SIZE	○	Refer to the Supported Image Size
COD5_CPC_SKIP_FRAME	○	0 ~ 32
COD5_CPC_BITRATE	○	CBR: 256 ~ 10000, VBR: 0 ~ 255
COD5_CPC_GOP_SIZE	○	1 ~ 127
COD5_CPC_AUDIO_ATTR	○	Sampling rate : 8000/16000 Bit rate : if “bits per sample” is 8bit, use the same value of “Sampling rate”. If it is 16bit, use 2times “Sampling rate” value. (Sampling rate x 16bit)
COD5_CPC_STREAM_TYPE	○	Audio, MD
COD5_CPC_CODEC_TYPE	○	Video Codec : Support COD5_VCT_MP4_ES or COD5_VCT_MJPEG Audio Codec : Support

		COD5_ACT_PCM, COD5_ACT_ULAW or COD5_ACT_ALAW.
COD5_CPC_LOOPBACK	○	Refer to the '3.5 Video Loopback'

2.2.3. Support Image Size

Image size	NTSC	PAL
D1	720x480	720x576
VGA	640x480	640x480
QVGA	320x240	320x240
4CIF	704x480	704x576
2CIF	704x240	704x288
CIF	352x240	352x288
QCIF	176x112	176x144

2.2.4. Codec Adjustment Command

Command	Supported	Remarks
COD5_CAC_BRIGHTNESS	○	0 ~ 255
COD5_CAC_CONTRAST	○	0 ~ 255
COD5_CAC_SATURATION_U	○	0 ~ 255
COD5_CAC_SATURATION_V	○	0 ~ 255
COD5_CAC_HUE	○	0 ~ 255
COD5_CAC_AUDIO_GAIN	○	0 ~ 255
COD5_CAC_VM_SEL_CHANNEL	X	
COD5_CAC_OSD_STRING	○	uParam0 is pointer of COD5_OSD_STRING_INFO structure.
COD5_CAC_OSD_SHOW	○	Sets the uParam0 1 to show the OSD string. Sets the uParam0 0 to hide the OSD string.
COD5_CAC_OSD_CLEAR	X	

3. Programming Guide

In this section, the API limitation owing to the characteristics of the hardware, particulars or detail method of API usage will be explained with regard to editing the APP using the API. Any parts not explained here shall be referred to in the UDA5 SDK Standard manual. NVE utilizes Cod5 API.

3.1. Using API

Using the UDA5 API in the NVE requires the procedure to query the API interface and connect to the server.
The API interface pointer should be retrieved per channel basis. One NVE4000 has four channels. You have to create four APIinterface pointers for one NVE4000 device. Following code shows creation of interface pointers. For convenience of the explanation, error processing is not performed.

```
#define NUM_OF_CH       4

struct UDA5GlobAPI{
BOOL (__stdcall *Cmn5CreateInstance)(IUnknown **pUnknown);
};

void GetInterface()
{
      UDA5GlobAPI Uda5Api;
      ICod5* pCod5Api[NUM_OF_CH];
      INet5* pNet5Api[NUM_OF_CH];

      for(ULONG i=0;i< NUM_OF_CH;i++){
      IUnknown * pUnknown;
      Uda5Api.Cmn5CreateInstance(&pUnknown);

      pUnknown->QueryInterface (IID_ICod5,(void**)&pCod5Api[i]);
      pUnknown->QueryInterface (IID_INet5,(void**)&pNet5Api[i]);
      pUnknown->Release();
}
}
```

If you want to connect two NVE4000 devices, the number of interface pointers should be 8.
When you are done with the interface pointers, you have to release the pointers as the following code.

```
void ReleaseInterface()
{
      for(ULONG i=0;i< NUM_OF_CH;i++){
      if(pCod5Api[i]){
            pCod5Api[i]->Release();
            pCod5Api[i] = NULL;
      }

      if(pNet5Api[i]){
```

```
            pNet5Api[i]->Release();
            pNet5Api[i] = NULL;
        }
        }
}
```

3.1.1. Connect

In order to use the Cod5/Cap5 API provided by the NVE, it shall be connected to the server by using the Net5ConnectEx of the Net5 API. For disconnection, call Net5Disconnect. For further details, refer to the Net5 API Manual.

```
// Connect to server
pNet5Api->Net5ConnectEx(”rtsp://192.168.0.10” NET5_ST_RTP_UDP, 5);
pNet5Api->Net5Login(“root”, “pass”);
// {
// Call Cod5 API……
// }
pNet5Api->Net5Logout();
// Disconnect
pNet5Api->Net5Disconnect();
```

3.2. Process Control

In all the cards supplied by our company, the accessible APIs will be determined by Process Control. The NVE supports the Cod5API and the API will work correctly only when the Process Control is applied. Some other APIs related with the above are also affected by this factor.

3.2.1. Process Control API

Normally, as one API is controlled through a DLL and independent hardware, it will work independently even if it is used in conjunction with other API.
The Process Control state will vary depending on the Setup, Run, Stop, Endup functions, and there is a particular API designated that is callable at each state. If any API that is not allowable at the current state is called, there will be an API failure.

```
// Current Process State : INIT
pCod5Api->Cod5Setup();
// Current Process State : SETUP

// {
// do something……
// }

pCod5Api->Cod5Run();
// Current Process State : RUN

pCod5Api->Cod5Stop();
// Current Process State : SETUP
```

```
pCod5Api->Cod5Endup();
// Current Process State : INIT
```

3.2.2. Activate

Though it is not a Process Control API, the Activate() API shall be essentially called to execute the Process control setup.

3.2.3. Error Code

In case of Error Code, the GetLastErrorCode shall be called and brought in.

3.3. Video compression setting

In order to compress the video data, it is necessary to set the relevant parameters.
The applicable parameters in the NVE are as shown in the table below.

Parameter	Description
Video Codec	Video codec selection (MPEG-4/MJPEG)
GOP Size	I Picture Frame interval (MPEG-4 only)
Bit Rate	Bit rate mode selection (CBR/VBR)
CBR Bit Rate	Target bit rate in CBR mode
VBR Quant	Quantization value in VBR mode

3.3.1. GOP Size

Since the number of created I frames increases as the GOP size is reduced, the data size become larger. Therefore, when reducing the GOP size, the Frame Rate and Bit Rate must be reduced appropriately. Otherwise, the data may be lost during transmission through the network.

3.3.2. Bit Rate

Cod5 API supports 4 types of bit rate such as COD5_BM_CBR, COD5_BM_VBRR, COD5_BM_VBRQ, COD5_BM_VBRH. The NVE supports COD5_BM_CBR and COD5_BM_VBRQ. Other bit rate modes are not implemented. Each bit rate is determined by using the COD5_CPC_BITRATE index of the Cod5SetCodecProperty API. In accordance with the index, the meaning of the second parameter will change the value of the remaining parameters. More details regarding the Parameter setting shall be referred to in the Cod5 API Reference manual.

3.4. External Video Out

Only NVE4000 can use external video out. Other models cannot support it. NVE4000 is available external video out by two modes. One is Multiview mode and the other is Switching mode. Multiview mode display multiple video sources on a single screen. And Switching mode display the selected one channel image on a single screen. External video out mode can be set by Cod5SetExtVideoOut. To use multiview mode, set ‘uCh’ parameter to 0xff, and to use switching mode, set ‘uCh’ parameter to

channel number (0~3).

3.5. Video Loopback and Dual Stream

The dual stream is for making two streams of different video settings (image size, codec type, frame rate and so on) using one video source. For an example MPEG-4 for one channel and MJPEG for the other. For using dual stream, it needs to set video loopback function which is to link one input (VIN1) to the other input (VIN2).

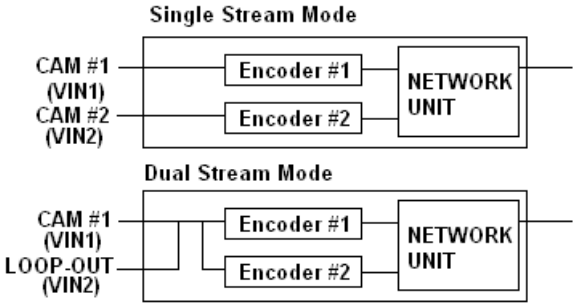

Figure 1. Single Stream and Dual Stream of NVE2000

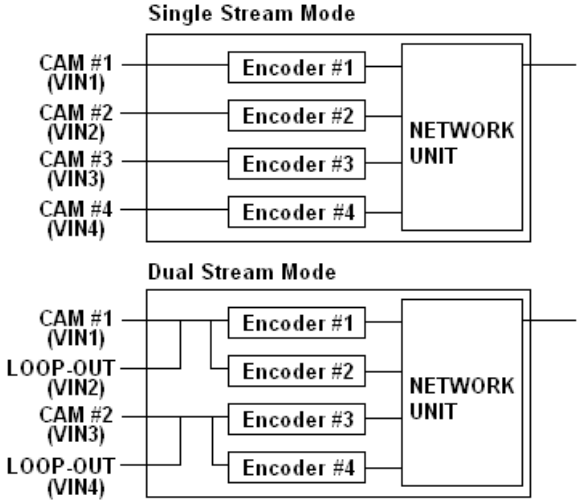

Figure 2. Single Stream and Dual Stream of NVE4000

COD5_CPC_LOOPBACK

```
Cod5SetCodecProperty(uBdID, uCh, uVideoPropertyCmd, uParam0, uParam1, uParam2, uParam3);
```

Parameters

uBdID
Not used. Just set 0.

uCh
Not used. Just set 0.

uVideoPropertyCmd
COD5_CPC_LOOPBACK

uParam0
It is bitwise format. The available video loopback setting is as following table.

	NVE1000	NVE2000	NVE4000
VIN1-->VIN2 or VIN1-->Loopout	1	1	1
VIN3-->VIN4	X	X	2
VIN1-->VIN2 and VIN3-->VIN4	X	X	3

uParam1
Not used. Just set 0.

uParam2
Not used. Just set 0.

uParam3
Not used. Just set 0.

See also

Cod5SetCodecProperty, Cod5SetMultiCodecProperties described at “UDA5 Cod5 API Reference - Eng.pdf”.

Example

```
// enable video loopback

ULONG bwCh = 0;
for (ULONG i=0; i<2; i++) {     // only NVE4000 has two video loopback
bwCh |= 1 << i                  // bitwise format
}

CMN5_MULTI_COMMAND Cmd;
Cmd.uBoard = 0;
Cmd.uChannel = 0;
Cmd.uCommand = COD5_CPC_LOOPBACK;
Cmd.uParam[0] = bwCh;

m_pCod5Api->Cod5SetMultiCodecProperties(1, &Cmd);
```

Caution

Insert only one video source (0 or 1, 2 or 3). Otherwise two sources are mixed.

3.6. Serial Port

NVE series support RS-232C and RS-422/485. Please be aware that not all models support RS-422. Basically, RS-232C is used for system's terminal and RS-485 is used for controlling camera (ex. Pan/Tilt). User can change only RS-485 parameters. The parameters of RS-232C are fixed, if it is set to 'Internal mode'. Changeable settings are baudrate, parity, stopbit, flow_control. Available values are as the below.

Parameters	Values available
baudrate	1200, 2400, 4800, 9600, 19200, 38400, 115200
parity	none, event, odd
stopbit	1, 2
Flow_control	none, software(Xon/Xoff), hardware(RTS/CTS)

3.6.1. Internal and External modes of serial ports

NVE uses RS-232C and RS-422/485 ports exclusively. Other programs are not allowed to use these ports. However, programs such as add-on application and 'tcp2ser' need to access these serial ports. 'External mode' enables it.
Each port can be configured as either 'Internal mode' or 'External mode'. In 'Internal mode', NVE controls the port exclusively. Other programs should not access the port. In 'External mode', NVE releases accessing of the port. Other programs can access the ports freely.
RS-422/485 port can be configured as either 'Internal mode' or 'External mode' in run time. However, changing mode of RS-232C port needs rebooting of NVE. NVE maintains the mode of RS-232C port until it reboots.
The configured modes of serial ports are stored at the non-volatile memory (flash memory). It preserves the modes across rebooting. The hard-factory default modes of serial ports are 'Internal mode'.
RTSP command is used to change the mode of serial ports. Followings are the commands.

RTSP command for 'Internal mode' of RS-232C port:

```
EXT_CMD / RTSP/1.0
CSeq: 1
CmdCount: 1
SET SERIAL_DATA portid=0, mode="internal"
```

RTSP command for 'External mode' of RS-232C port:

```
EXT_CMD / RTSP/1.0
CSeq: 1
CmdCount: 1
SET SERIAL_DATA portid=0, mode="external app_name"
```

RTSP command for 'Internal mode' of RS-422/485 port:

```
EXT_CMD / RTSP/1.0
CSeq: 1
CmdCount: 1
SET SERIAL_DATA portid=1, mode="internal"
```

RTSP command for 'External mode' of RS-422/485 port:

```
EXT_CMD / RTSP/1.0
CSeq: 1
CmdCount: 1
SET SERIAL_DATA portid=1, mode="external app_name"
```

'portid' for RS-232 is '0', for RS-422/485 is '1'. 'app_name' specifies a program name that NVE executes when it changes serial port mode to 'External mode'. 'app_name' should contain absolute path and file name. It can also contain additional command line parameters. For example, you can specify '… mode="external /mnt/user1/tcp2ser 9600 8 …". If you leave 'app_name' blank, NVE only changes serial port mode as specified.

3.7. OSD String

NVE series can insert a string on the image before compression. It can support up to 90 characters. Character size on the image is fixed to 16x16 pixels. So the string is moved by 16x16 pixels.

```
// show osd string

CMN5_MULTI_COMMAND Cmds[2];

COD5_OSD_STRING_INFO Info;
Info.uX = 0;                        // x coordinate
Info.uY = 1;                        // y coordinate
strcpy(Info.String, “osd”);         // string to show on the image
Info.uBgColor = 0;                        // not used
Info.uFgColor = 255;                // gray color. 0 ~ 255
Info.uLength = strlen(Info.String);   // length of bytes. If the string is
// UNICODE characters, ‘uLength’ is twice
// of number of the string.

Cmds[0].uBoard = 0;
Cmds[0].uChannel = 0;
Cmds[0].uCommand = COD5_CAC_OSD_STRING;
Cmds[0].uParam[0] = (ULONG)&Info;

Cmds[1].uBoard = 0;
Cmds[1].uChannel = 0;
Cmds[1].uCommand = COD5_CAC_OSD_SHOW;
```

```
Cmds[1].uParam[0] = 1;

m_pCod5Api->Cod5SetMultiCodecAdjusts(2, Cmds);
```

Note: When character is supported over 90 , OSD string function doesn't work correctly. Moreover, when the character is supported over 100, video image transferred from NVE is very slow.

3.8. Export APIs

UDA5 of NVE basically uses "COM-like" APIs, but it supports "Export" API from firmware version (NVE SDK V1.5.0 or later). "COM-like" APIs manages the connections of NVE devices by each COM instance however Export APIs should be set device ID. The first parameter of "Export" APIs is device ID and it can be retrieved through parameter of exNet5ConnectEx. The below table shows the differences between "COM-like" APIs and "Export" API type.

COM	Export
Net5 API	
Net5Connect	exNet5Connect
Net5Disconnect	exNet5Disconnect
Net5Command	exNet5Command
Net5SetApiTimeOut	exNet5SetApiTimeOut
Net5GetApiTimeOut	exNet5GetApiTimeOut
Net5SetEventHandle	exNet5SetEventHandle
Net5GetEventHandle	exNet5GetEventHandle
Net5GetEventData	exNet5GetEventData
Net5GetLastErrorCode	exNet5GetLastErrorCode
Net5ConnectEx	exNet5ConnectEx
Net5Login	exNet5Login
Net5Logout	exNet5Logout
Cod5 API	
Cod5GetLastErrorCode	exCod5GetLastErrorCode
Cod5QueryInfo	exCod5QueryInfo
Cod5Setup	exCod5Setup
Cod5Endup	exCod5Endup
Cod5Run	exCod5Run
Cod5Stop	exCod5Stop
Cod5ReleaseData	exCod5ReleaseData
Cod5SetEventHandle	exCod5SetEventHandle
Cod5GetEventHandle	exCod5GetEventHandle
Cod5GetEventData	exCod5GetEventData
Cod5SetEventData	exCod5SetEventData

* The undefined Cod5 APIs on the above table, they can be used by replace device ID to board ID. The device ID can be retrieved by using exNet5ConnectEx.

```
// export APIs

ULONG uID;
exNet5ConnectEx(&uID, “rtsp://192.168.0.10”, NET5_ST_RTP_UDP, 10);
exNet5Login(uID, “root”, “pass”);

exCod5Setup(uID);
exCod5Run(uID);

//    process data

exCod5Stop(uID);
exCod5Endup(uID);

exNet5Disconnect(uID);
```

3.9. Add-on application

Add-on application lets a user add his/her own functionalities on NVE by executing special application on NVE itself. For example, add-on application transfers image data to a remote server at events that it is interested in.
Add-on application is previously known as 'user application'.

Restrictions

Generally, the add-on application cannot control hardware resources directly. However, RS-232C and RS-422/485 ports can be accessible after changing NVE configuration.

Where does add-on application exist?

/mnt/user1 and /mnt/user2 are user application area. Add-on application is uploaded here.

How to upload add-on application

You uses IPAdminTool to upload add-on application to the user application area.

Execution of add-on application

After power-up, NVE loads and executes kernel OS, device drivers, and server applications. After loading and executing are completed, NVE looks for /mnt/user1/user1.rc. If it finds user1.rc file, it executes the script file. The script file contains the name of add-on applications.

The script file, user1.rc

NVE uses Adventure shell (ash). user1.rc file is not NVE's default files. You have to make user1.rc at client PC. Then, you have to upload created user1.rc file to NVE's user application area using IPAdminTool. The path used in user1.rc file should be absolute path.
Example)
We assume that 'myapp' is add-on application. At first, upload 'myapp' to /mnt/user1 path. Create user1.rc file that contains /mnt/user1/myapp. Upload user1.rc file to /nmt/user1 path.

Communication between add-on application and NVE's server application

Add-on application can communicate NVE with the same APIs, RTSP, HTTP, which are APIs that a remote client application uses. UDA5, which is [UDP Co.]'s own API cannot be used.
NVE treats add-on application as a client connected through network. A network client connects from a remote site. Add-on application connects from inside. Add-on application connects with local IP address.
This connection method is not optimal, because it uses same amount of CPU resource as connection through RTSP or HTTP. However, there is no other interfaces other than RTSP and HTTP that NVE can provide at this time.

Build environment

You can build add-on application at Red Hat V9.x or higher. Two libraries are needed. At first, install 'sdelinux-5.01-4.i386.rpm'. Use cross compiler 'mipsel-linux-gnu-gcc.3.3.4'. These libraries are provided on the request.
NVE uses Linux kernel 2.6.12 for your reference.

RS-232C and RS-422/485

These ports are controlled exclusively by NVE. When NVE controls these ports, you are not allowed to control these ports. However, there is a way that NVE releases controlling of these ports so that add-on application can access these ports. Please refer section 3.6 Serial Port.

3.10. How to send RTSP extension command

UDA5 API does not provide all function of NVE. So UDA5 supports API for sending RTSP extension command and receiving response to control NVE. Regarding the syntax of RTSP extension command, please refer to ’NVE RTSP Reference Manual.pdf’.

Example

```
char szCmd[NVE_MAX_EXT_CMD_LENGTH];

sprintf(szCmd, "EXT_CMD / RTSP/1.0\r\n"
      "CSeq: 1\r\n"
      "CmdCount: 1\r\n"
      "GET HARDWARE_INFO\r\n\r\n");

m_pNet5Api->Net5Command(NET5_NC_SEND_RTSP_REQUEST, (ULONG*)szCmd,
      NULL, NULL, NULL);
m_pNet5Api->Net5Command(NET5_NC_RECV_RTSP_RESPONSE, (ULONG*)szCmd,
      NULL, NULL, NULL);
```

3.11. How to send HTTP command

UDA5 API does not provide all function of NVE. So UDA5 supports API for sending HTTP command and receiving response to control NVE. Regarding the syntax of HTTP command, please refer to ‘NVE HTTP API Manual-Eng.pdf’.

Example

```
char szCmd[NVE_MAX_HTTP_CMD_LENGTH];

sprintf(szCmd, "/enc-cgi/admin/param.cgi?action=list\r\n\r\n");

m_pNet5Api->Net5Command(NET5_NC_SEND_CGI_REQUEST, (ULONG*)szCmd,
      NULL, NULL, NULL);
m_pNet5Api->Net5Command(NET5_NC_RECV_CGI_RESPONSE, (ULONG*)szCmd,
      NULL, NULL, NULL);
```

3.12. Multicast

The multicast of NVE can be configured by RTSP extension command. For more detailed information, please refere to RTSP Method Reference.chm.

3.13. Motion Detection

NVE has three layers for motion detection and each layer has the eight motion areas as rectangle shape. Each layer can be configured by activity and threshold. Each rectangle is configured as each pre configured layer that the rectangle is included at.
The image is divided by macro block as 16 pixels by 16 pixels. Eg. CIF 320x240 has 20x15 macro blocks. The threshold means the sensitivity of each macro block and its range is from 1 to 255. The lower number can detect the motion more sensitively. The activity means the proportion of macro block that motion is detected under the motion detection area. Its range is from 1 to 255. The lower number can detect the motion sensitively. Eg. if this value is 128, the motion detection actives when the motion is detected on more than 50 % of macro blocks.

The motion detection can be configured by *Cod5SetCodecProperty* / *Cod5SetMultiCodecProperties* and the information of the motion detection can be retrieved by *Cod5GetCodecProperty* / *Cod5GetMultiCodecProperties.*

COD5_CPC_MD_PROPERTY

This property command is for getting the total number of layer, the maximum rectangle per each layer and MD area type supported.

```
Cod5GetCodecProperty(uBdID, uCh, uVideoPropertyCmd,
                              uParam0, uParam1, uParam2, uParam3);
```

Parameters

uBdID
Not used. Just set 0.

uCh
Not used. Just set 0.

uVideoPropertyCmd
COD5_CPC_MD_PROPERTY

uParam0
Not used. Just set 0.

uParam1
Not used. Just set 0.

uParam2
COD5_MD_PROPERTY structure pointer

uParam3
Not used. Just set 0.

See also

Cod5GetCodecProperty, Cod5GSetMultiCodecProperties described at "UDA5 Cod5 API Reference - Eng.pdf".

COD5_CPC_MD_INFO
This property command is for getting or setting the configuration of motion detection as threshold, activity of each and so on.

```
Cod5SetCodecProperty(uBdID, uCh, uVideoPropertyCmd,
                     uParam0, uParam1, uParam2, uParam3);
```

```
Cod5GetCodecProperty(uBdID, uCh, uVideoPropertyCmd,
                     uParam0, uParam1, uParam2, uParam3);
```

Parameters

uBdID
Not used. Just set 0.

uCh
Not used. Just set 0.

uVideoPropertyCmd
COD5_CPC_MD_INFO

uParam0
Not used. Just set 0.

uParam1
Not used. Just set 0.

uParam2
COD5_MD_INFO structure pointer

uParam3
Not used. Just set 0.

See also

Cod5GetCodecProperty, Cod5SetCodecProperty, Cod5SetMultiCodecProperties, Cod5GetMultiCodecProperties described at “UDA5 Cod5 API Reference - Eng.pdf”.

Example

Example for getting the MD property and the configuration

```
#define MAX_LAYER_COUNT     3
#define MAX_RECT_COUNT      8

void GetMDInfo()
{
    COD5_MD_PROPERTY MDProp;
    COD5_CPC_MD_INFO MDInfo[MAX_LAYER_COUNT];
    COD5_MD_AREA_RECT AreaRect[MAX_RECT_COUNT];

    m_pCod5Api->Cod5GetCodecProperty(0, 0, COD5_CPC_MD_PROPERTY,
                                     (ULONG*)&MDProp, 0, 0, 0);

    BYTE *pMDInfo = new BYTE[MDProp.uInfoSize];
    m_pCod5Api->Cod5GetCodecProperty(0, 0,
                                    COD5_CPC_MD_INFO, (ULONG*)pMDInfo,
                                    0, 0, 0);

    ULONG uIndex = 0;
    for (ULONG i=0; i<MDProp.uMaxLayerCount; i++) {
        COD5_MD_INFO *pInfo = (COD5_MD_INFO*)(pMDInfo + uIndex);
        CopyMemory(&MDInfo[i], pInfo, sizeof(COD5_MD_INFO));
        uIndex += sizeof(COD5_MD_INFO);

        for (ULONG j=0; j<pInfo->uAreaCount; j++) {
            ULONG uType = *(ULONG*)(pMDInfo + uIndex);
            uIndex += sizeof(ULONG);
            ULONG uSize = *(ULONG*)(pMDInfo + uIndex);
            uIndex -= sizeof(ULONG);
            if (uType == COD5_MDT_RECT) {
                COD5_MD_AREA_RECT *pRect =
                        (COD5_MD_AREA_RECT*)(pMDInfo + uIndex);
                CopyMemory(&AreaRect[pRect->uRectID], pRect, uSize);
            }
            uIndex += uSize;
        }
    }

    delete[] pMDInfo;
}
```

Example for setting the MD configuration

```
void SetMDInfo()
{
    COD5_CPC_MD_INFO MDInfo[MAX_LAYER_COUNT];
    COD5_MD_AREA_RECT AreaRect[MAX_RECT_COUNT];
```

```
    // The settings of eeach layer must be configured .
    for (ULONG i=0; i<MAX_LAYER_COUNT; i++) {
        COD5_MD_AREA_RECT *pRect =
            new COD5_MD_AREA_RECT[MAX_RECT_COUNT];
        BYTE *pMDInfo;

        MDInfo[i] uVersion = 1;
        MDInfo[i].uLayerID = i
        MDInfo[i].uEnable = 1;
        MDInfo[i].uActivity = 128;
        MDInfo[i].uThreshold = 128;

        for (ULONG j=0; j<MAX_RECT_COUNT; j++) {
            AreaRect[j].uType = COD5_MDT_RECT;
            AreaRect[j].uSize = sizeof(AreaRect[j]);
            AreaRect[j].uRectID = j;
            AreaRect[j].uX = j*64;
            AreaRect[j].uY = 64;
            AreaRect[j].uWidth = 64;
            AreaRect[j].uHeight = 64;
            CopyMemory(pRect+j, &AreaRect[j], sizeof(COD5_MD_AREA_RECT));
        }
        MDInfo[i].uAreaCount = MAX_RECT_COUNT;
        MDInfo[i].uAreaStructSize =
                        sizeof(COD5_MD_AREA_RECT)*MDInfo[i].uAreaCount;
        MDInfo[i].uInfoType = COD5_MDI_BITMAP;

        pMDInfo = new BYTE[sizeof(COD5_MD_INFO) + MDInfo[i].uAreaStructSize];
        CopyMemory(pMDInfo, &MDInfo[i], sizeof(COD5_MD_INFO));
        CopyMemory(pMDInfo+sizeof(COD5_MD_INFO),
                                        pRect, MDInfo[i].uAreaStructSize);

        m_pCod5Api->Cod5SetCodecProperty(0, 0,
                        COD5_CPC_MD_INFO, (ULONG)pMDInfo, 0, 0, 0)) {
        delete[] pMDInfo;
        delete[] pRect;
    }
}
```

COD5_MD_PROPERTY

```
struct _COD5_MD_PROPERTY {
    union {
        ULONG           reserved[8];
        struct {
            ULONG           uMaxLayerCount;
            ULONG           uMaxRectCount;
            ULONG           uAreaType;
            ULONG           uInfoSize;
        };
    };
} COD5_MD_PROPERTY;
```

Mmembers

uMaxLayerCount
The maximum number of layer supported.

uMaxRectCount
The maximum number of the rectangle per each layer supported.

uAreaType
The area type supported.
If this value is 0, the area type is Rectangle. If it is 1, the area type is Bitmap.
(Now only Rectangle is supported.)

uInfoSize
sizeof(COD5_MD_INFO) * uMaxLayerCount

COD5_MD_INFO

```
struct _COD5_MD_INFO {
    union {
        ULONG           reserved[16];
        struct {
            ULONG           uVersion;
            ULONG           uLayerID;
            ULONG           uEnable;
            ULONG           uActivity;
            ULONG           uThreshold;
            ULONG           uAreaCount;
            ULONG           uAreaStructSize;
            ULONG           uInfoType;
        };
    };
} COD5_MD_INFO;
```

Members

uVersion
The version of COD5_MD_INFO structure

uLayerID
The ID of layer.

uEnable
Specifies the enable/disable of layer.

uActivity
The minimum proportion of the exceeded threshold of macro block in the #Area.

uThreshold
The sensitivity of each macro block (16 pixel by 16 pixel)

uAreaCount
The number of the MD area

uAreaStructSize
The size of MD area structure * uAreaCount

uInfoType
The information type of MD. It has one among COD5_MDI_FLAG, COD5_MDI_COUNT and COD5_MDI_BITMAP.

Remarks

uInfoType is available from Kernel16X531

COD5_MD_AREA_RECT

```
struct _COD5_MD_AREA_RECT {
    union {
        ULONG           reserved[16];
        struct {
            ULONG           uType;
            ULONG           uSize;
            ULONG           uRectID;
            ULONG           uX;
            ULONG           uY;
            ULONG           uWidth;
            ULONG           uHeight;
        };
    };
} COD5_MD_AREA_RECT;
```

Members

uType
The area type
If this value is 0, the area type is Rectangle. If it is 1, the area type is Bitmap.

uSize
The size of COD5_MD_AREA_RECT structure

uRectID
The ID of Rectangle area

uX
The x-coordinate of the upper-left corner of the rectangle

uY
The y-coordinate of the upper-left corner of the rectangle

uWidth
The width of the rectangle

uHeight
The height width of the rectangle

Retrieving MD information data

MD information data is able to be retrieved using *Cod5GetEventData* function. If *uFrameType* of COD5_DATA_INFO_EX structure is COD5_FT_MD_1, *pDataBuffer* of COD5_DATA_INFO_EX structure is MD information data. *pDataBuffer* has as following value according to *uInfoType* of COD5_MD_INFO that is set.

uInfoType	pDataBuffer
COD5_MDI_FLAG	Layer
COD5_MDI_COUNT	Layer + Macroblock Count + Width + Height
COD5_MDI_BITMAP	Layer + Macroblock Count + Width + Height + MD Bitmap Information

* COD5_MDI_BITMAP is available from Kernel16X531.

Layer : char
The layer ID where MD event occurs. It is bitwise. E.g.) Layer is 3 when MD event occurs at both layer 0 and layer1.
Macroblcok Count : DWORD
The number of macroblock motion detected when MD event occurs.
Width : WORD
The width of a image.
Height : WORD
The height of a image.
MD Bitmap information : char array
The MD macroblock information. One bit indicates the status of one macroblock. If value is 1, it indicates motion detected.

Example

```
#define MACROBLOCK_SIZE      16

typedef struct _MD_INFO_COUNT {
      ULONG uMDCount;
      WORD wWidth;
      WORD wHeight;
} MD_INFO_COUNT;

void GetMDData()
{
      HANDLE hEvent;
      COD5_DATA_INFO_EX DataInfo;

      pCod5Api->Cod5GetEventHandle(CMN5_DT_COD_EX, &hEvent);

      while (1) {
            ULONG uRet = WaitForSingleObjects(hEvent, INFINITE);
            if (uRet == WAIT_OBJECT_0) {
                  pCod5Api->Cod5GetEventData(CMN5_DT_COD_EX, &DataInfo);
                  if(DataInfo->uFrameType == COD5_FT_MD_1) {
                        ShowMDData(DataInfo->pDataBuffer);
                  }
            }
      }
}
```

```
void ShowMDData(char *pMDData)
{
	ULONG uInfoType;
	MD_INFO_COUNT InfoCount;
	char *pBitmap;

	char cLayer = *(char*)pMDData;
	CopyMemory(&InfoCount, pMDData + sizeof(char));
	pBitmap = pMDData + sizeof(char) + sizeof(MD_INFO_COUNT);

	ULONG uPicMBWidth = InfoCount.wWidth/ MACROBLOCK_SIZE;
	ULONG uPicMBHeight = InfoCount.wHeight/ MACROBLOCK_SIZE;
	ULONG uCodMBWidth = uPicMBWidth;

	for(ULONG y=0; y<uPicMBHeight; y++) {
		for(ULONG x=0; x<uPicMBWidth; x++) {
			ULONG uPos = (y*uCodMBWidth + x)/8;
			ULONG uBit = (y*uCodMBWidth + x)%8;
			ULONG d = pBitmap[uPos] & (1<<uBit);
			if (d) {
				// Draw macro block
			}
		}
	}
}
```

3.14. Using DDNS

It's difficult to connect with NVE at dynamic IP environment. Because the IP address can be frequently changed without notification. So DDNS service is made to avoid this kind of problems. NVE provides the function which uses DDNS service of dyndns. This chapter is for explaining to use DDNS service with NVE.

3.14.1. Using Dyndns

(1) To enable DDNS function

At first, to use DDNS service of dyndns, you receive an account of dyndns, and registry domain name to use. And set up the NVE to do automatically update own IP address that is owned by dyndns server. To set up NVE for automatically update, you use NET5_NC_SET_EVENT_AUTOMATION command and NET5_EVENT_AUTOMATION_INFO_

DDNS structure. Setting the dyndns user ID, password, registered domain name and etc at NET5_EVENT_AUTOMATION_INFO_DDNS structure object. And set the value of uCommand member variable by NET5_EAC_ADD. At last, calling Net5Command API with parameters NET5_NC_SET_EVENT_AUTOMATION and NET5_EVENT_AUTOMATION_INFO_DDNS structure that was explained. To deeply understanding, refer the next example.

For using DDNS service of Dyndns needs to create user ID/Password and register domain name from web site Dyndns (www.dyndns.com). After that, it needs to set up for updating NVE dynamic IP address on Dyndns server automatically by using NET5_NC_SET_EVENT_AUTOMATION command and NET5EVENT_AUTOMATION_INFO_DDNS structure.

NET5_EVENT_AUTOMATION_INFO_DDNS structure object has the information of user ID/Password and registered Domain name. uCommand member variable of

NET5_EVENT_AUTOMATION_INFO_DDNS structure object has NET5_EAC_ADD.
Net5Command API is called by parameter of NET5_NC_SET_EVENT_AUTOMATION and NET5_EVENT_AUTOMSTION_INFO_DDNS structure.
There is example for using DDNS.

```
void EnableDDNS()
{
    // example..
    // dyndns ID: nveuser1
    // password: nveuser1
    // domain name: nveuser1.dvrdns.org

    NET5_EVENT_AUTOMATION_INFO_DDNS Ddnsinfo;
    ULONG uNum = 1;

    Ddnsinfo.uAutomationType = NET5_EAT_DDNS;
    Ddnsinfo.uStructSize = sizeof(NET5_EVENT_AUTOMATION_INFO_DDNS);
    Ddnsinfo.uCommand = NET5_EAC_ADD;
    Ddnsinfo.uDdnsServiceType = NET5_DDNS_SERVICE_TYPE_DYNDNS;
    strcpy(Ddnsinfo.DdnsServerAddr, “www.dyndns.com”);
    strcpy(Ddnsinfo.DdnsUserName, “nveuser1”);
    strcpy(Ddnsinfo.DdnsPassword, “nveuser1”);
    strcpy(Ddnsinfo.DomainName, “nveuser1.dvrdns.org”);
    Ddnsinfo.uUpdateIntervalMin = 60;

    Net5Command(NET5_NC_SET_EVENT_AUTOMATION, &uNum,
    (ULONG*)&Ddnsinfo, NULL, NULL);
}
```

(2) To take DDNS information
To get the number of event automation of NVE and the total size of structures uses NET5_NC_QUERY_EVENT_AUTOMATION_SIZE command for allocating buffer of structure size. This buffer can be set by using NET5_NC_GET_ALL_EVENT_AUTOMATION command for event automation information of NVE.

```
void GetDDNSInfo()
{
    ULONG uSize, uNum;
    BYTE *pBuff;
    ULONG uPos = 0;
    NET5_EVENT_AUTOMATION_INFO_DDNS DdnsInfo;

    Net5Command(NET5_NC_QUERY_EVENT_AUTOMATION_SIZE, &uSize, &uNum,
        NULL, NULL);

    if(uSize == 0) {
        // NVE has no event automation.
        return;
    }

    pBuff = new BYTE[uSize];
    // Parameter 2 is allocated buffer size of parameter 3.
```

```
    Net5Command(NET5_NC_GET_ALL_EVENT_AUTOMATION, &uSize,
        (ULONG*)pBuff, NULL, NULL);
    for (ULONG i=0; i<uNum; i++) {
        NET5_EVENT_AUTOMATION_INFO_HEADER *pInfoHeader =
            (NET5_EVENT_AUTOMATION_INFO_HEADER*)(pBuff + uPos);
        if (pInfoHeader->uAutomationType == NET5_EAT_DDNS) {
            // To get the date that
            CopyMemory(&DdnsInfo, pBuff+uPos,
                sizeof(NET5_EVENT_AUTOMATION_INFO_DDNS));
        }
        uPos += pInfoHeader->uStructSize;
    }
    delete[] pBuff;
}
```

(3) To disable DDNS function

To use NET5_NC_SET_EVENT_AUTOMATION command and NET5_EVENT_AUTOMATION _INFO_DDNS structure disables the DDNS function of NVE. SDK of NVE does not support to compare setting value between user input and NVE saved. NVE compares with uAutomationType, uCommand and uDdnsServiceType of NET5_EVENT_AUTOMATION_ INFO_DDNS to disable DDNS functions. Therefore, to cross check more information configures in your application program.

```
void DisableDDNS()
{
    // User input..
    // dyndns ID: nveuser1
    // password: nveuser1
    // domain name: nveuser1.dvrdns.org

    ULONG uNum;
    ULONG uSize;
    BYTE *pBuff;
    ULONG uPos = 0;
    NET5_EVENT_AUTOMATION_INFO_DDNS Ddnsinfo;

    Net5Command(NET5_NC_QUERY_EVENT_AUTOMATION_SIZE, &uSize, &uNum,
        NULL, NULL);

    if (uSize == 0) {
        // There's no information of event automation
        return;
    }

    pBuff = new BYTE[uSize];
    Net5Command(NET5_NC_GET_ALL_EVENT_AUTOMATION, &uSize,
        (ULONG*)pBuff, NULL, NULL);
    for (ULONG i=0; i<uNum; i++) {
        NET5_EVENT_AUTOMATION_INFO_HEADER *pInfoHeader =
            (NET5_EVENT_AUTOMATION_INFO_HEADER*)(pBuff + uPos);
    if (pInfoHeader->uAutomationType == NET5_EAT_DDNS) {
        CopyMemory(&DdnsInfo, pBuff+uPos,
            sizeof(NET5_EVENT_AUTOMATION_INFO_DDNS));
```

```
            if(Ddnsinfo.uDdnsServiceType != NET5_DDNS_SERVICE_TYPE_DYNDNS ||
                    strcmp(Ddnsinfo.DdnsUserName, " nveuser1") != 0 ||
                    strcmp(Ddnsinfo.DdnsPassword, " nveuser1") != 0 ||
                    strcmp(Ddnsinfo.DomainName, " nveuser1.dvrdns.org") != 0) {
                // Input is wrong
                delete[] pBuff;
                return;
            }
        }
        uPos += pInfoHeader->uStructSize;
    }
    delete[] pBuff;

    Ddnsinfo.uAutomationType = NET5_EAT_DDNS;
    Ddnsinfo.uCommand = NET5_EAC_DELETE;
    Ddnsinfo.uDdnsServiceType = NET5_DDNS_SERVICE_TYPE_DYNDNS;

    Net5Command(NET5_NC_SET_EVENT_AUTOMATION, &uNum,
        (ULONG*)&Ddnsinfo, NULL, NULL);
}
```

(4) To Change DDNS information

If Dyndns server ID and Password is changed, NVE setting value must be changed by using NET5_NC_SET_EVENT_AUTOMATION command and NET5_EVENT_AUTOMATION_ INFO_DDNS structure. In addition, to changing update period uses NET5_NC_SET_EVENT_ AUTOMATION command and NET5_EVENT_AUTOMATION_INFO_DDNS structure. uCommand of NET5_EVENT_AUTOMATION_INFO_DDNS structure has to get NET5_EAC_MODIFY. Also, NET5_EVENT_AUTOMATION_INFO_DDNS structure has to get New dyndns ID/Password and new update period.

It is called that Net5Command has parameters of NET5_NC_SET_EVENT_AUTOMATION and NET5_EVENT_AUTOMATION_INFO_DDNS structure.

SDK of NVE does not compare setting value between user input and NVE saved.. Therefore, to cross check the information configures in your application program.

```
void ChangeDDNSInfo()
{
    // The information that are going to be changed..
    // dyndns ID: nveuser2
    // password: nveuser2
    // Update interval: 30 minute

    NET5_EVENT_AUTOMATION_INFO_DDNS Ddnsinfo;
    ULONG uNum;
    ULONG uSize;
    BYTE *pBuff;
    ULONG uPos = 0;

    Net5Command(NET5_NC_QUERY_EVENT_AUTOMATION_SIZE, &uSize, &uNum,
        NULL, NULL));

    if (uSize == 0) {
        // There's no information of event automation
        return;
```

```
    }

    pBuff = new BYTE[uSize];
    Net5Command(NET5_NC_GET_ALL_EVENT_AUTOMATION, &uSize,
        (ULONG*)pBuff, NULL, NULL);

    for (ULONG i=0; i<uNum; i++) {
        NET5_EVENT_AUTOMATION_INFO_HEADER *pInfoHeader =
            (NET5_EVENT_AUTOMATION_INFO_HEADER*)(pBuff + uPos);
        if (pInfoHeader->uAutomationType == NET5_EAT_DDNS) {
            CopyMemory(&Ddnsinfo, pBuff+uPos,
                sizeof(NET5_EVENT_AUTOMATION_INFO_DDNS));
            if(Ddnsinfo.uDdnsServiceType != NET5_DDNS_SERVICE_TYPE_DYNDNS ||
                    strcmp(Ddnsinfo.DdnsUserName, " nveuser1") != 0 ||
                    strcmp(Ddnsinfo.DdnsPassword, " nveuser1") != 0 ||
                    strcmp(Ddnsinfo.DomainName, " nveuser1.dvrdns.org") != 0) {
                // Input is wrong
                delete[] pBuff;
                return;
            }
        }
        uPos += pInfoHeader->uStructSize;
    }
    delete[] pBuff;

    Ddnsinfo.uCommand = NET5_EAC_MODIFY;
    strcpy(Ddnsinfo.DdnsUserName, "nveuser2");
    strcpy(Ddnsinfo.DdnsPassword, "nveuser2");
    Ddnsinfo.uUpdateIntervalMin = 30;

    Net5Command(NET5_NC_SET_EVENT_AUTOMATION, &uNum,
    (ULONG*)&Ddnsinfo, NULL, NULL);
}
```

3.15. Sending audio data to NVE

As NVE supports only one audio output port and many clients have to compete to gain the privilege and have an access to audio port at a given time, the following lock functions are provided for proprietary usage of the audio port.

In order to send audio data, follow below procedure.

1. Gaining access to the audio output port
Access privilege is obtained using NET5_ATI_LOCK command. If the audio output port is already used by other client, uStatus will be set to '1' by Net5Command. However a client can obtain the access privilege of audio resource by force, even the audio source is being used by another, through NET5_NC_AUDIO_TX_FORCE_LOCK instead of NET5_NC_AUDIO_TX_LOCK.

```
void LockAudioResource {
    ULONG uSubCmd, uChannel, uStatus;
```

```
    uSubCmd = NET5_ATL_LOCK;
    uChannel = 0;
    Net5Command(NET5_NC_AUDIO_TX_LOCK, &uSubCmd, &uChannel, &uStatus,
        NULL);
}
```

2. Encoding audio data as specified format.

Before sending audio data, audio data must be encoded in the format as set at NVE. Available audio format is as follow.

format	Sampling Frequency	Bit per sample
PCM	16KHz, 8KHz	8bit, 16bit
uLaw	16KHz, 8KHz	8bit
aLaw	16KHz, 8KHz	8bit

For detailed information on the configuration of audio output setting, please refer to “3.10. RTSP extension command” and AUDIO_OUT extension method at NVE RTSP Reference Manual.pdf.

3. Sending audio data

The audio data encoded in a specified format can be sent using Cod5SetEventData API. Audio data can be transmitted upto 8192 bytes at once by calling the API. If the audio data size is set to too small, audio output may stop intermittently or if set to too large, the time delay may exist. So we recommend splitting audio data to 2 or 4 chunks per second before sending it.

```
#define MAX_AUDIO_DATA_SIZE     8192

void SendAudioData {
    COD5_DATA_INFO  DataInfo;
    BYTE pAudioBuff[MAX_AUDIO_DATA_SIZE];

    While(1) {
        // Copy audio data to ‘pAudioBuff’.
        …

        DataInfo.uDataType = DT_COD;
        DataInfo.uStructSize = sizeof(COD5_DATA_INFO);
        DataInfo.uErrCode = CMN5_EC_NO_ERROR;
        DataInfo.uBoardNum = 0;
        DataInfo.uChannelNum = 0;
        DataInfo.pDataBuffer = pAudioBuff;
        DataInfo.uDataSize = MAX_AUDIO_DATA_SIZE;
        DataInfo.uFrameType = COD5_FT_PCM_AUDIO;

        Cod5SetEventData(DT_COD, &DataInfo);

        // If you sent all data, break this loop.
    }
}
```

4. Releasing lock for use by another client

```
void UnlockAudioResource {
```

```
    ULONG uSubCmd, uChannel, uStatus;

    uSubCmd = NET5_ATL_UNLOCK;
    uChannel = 0;
    Net5Command(NET5_NC_AUDIO_TX_LOCK, &uSubCmd, &uChannel, &uStatus,
        NULL);
}
```

3.16. Encrypting the streams

Video stream or Audio stream, especially on multicast, also need to be encrypted for security. So NVE/IPC adopts AES(Advanced Encryption Standard) for encrypting stream. UDA5 API for configuring AES of server side is not provided, you should send RTSP command through NET5_NC_SEND_RTSP_REQUEST command. About RTSP command of AES configuration, refer to 3.2.5 Encryption of NVE RTSP Reference Manual.pdf.
In order to decrypt the encrypted stream, DLL have to know the encryption key. Using NET5_NC_SET_ENCRYPTION_KEY, you can set the encryption key to DLL.

```
#define KEY_SIZE    16

void SetEncryptionKey()
{
    UCHAR pVideoKey[KEY_SIZE] = “NveVideoKey”;
    UCHAR pAudioKey[KEY_SIZE] = “NveAudioKey”;
    ULONG uMediaType;

    uMediaType = NET5_EMT_VIDEO;
    m_pNet5Api->Net5Command(NET5_NC_SET_ENCRYPTION_KEY, &uMediaType,
        (ULONG*)pVideoKey, NULL, NULL);

    uMediaType = NET5_EMT_AUDIO;
    m_pNet5Api->Net5Command(NET5_NC_SET_ENCRYPTION_KEY, &uMediaType,
        (ULONG*)pAudioKey, NULL, NULL);
}
```

Revision history

Revision	Date	Description
A	2006-05-02	Created
B	2006-08-29	Added the descriptions of the available value range, Net Command, External Video Out, Video Loopback, Serial Port and OSD String.
C	2006-11-08	Changed the Cod5 API, Added the Export APIs.
D	2006-11-25	Add the maximum character for OSD string function in the section 3.7
E	2006-12-05	Updated Section 3.5
F	2007-01-16	Added Add-on application, Serial port RTSP
G	2007-01-23	Changed document format
H	2007-03-13	Added Motion Detection, Muticast, RTSP extension command
I	2007-05-30	Added DDNS
J	2007-06-21	Updated Section 3.1
K	2007-06-25	Add the description of motion detection information.
L	2007-07-13	Fixed the parameter of Net5ConnectEx API at the sample code
M	2007-08-09	Added Cod5GetVideoStatus
N	2007-09-17	Added section about sending audio data.
O	2008-01-10	Added the Encryption of stream.
P	2008-02-22	Added how to send HTTP command.