Cellinx SDK
Reference Guide
Network
Low Level Library

Cellinx SDK Reference Guide
- Network Low Level Library

Document Control No: CELLINX-SDK-NVSNETLIB-V001
Document Version: 2.02
Date: 2009. 09. 11

Cellinx SDK
Reference Guide
2009. 09. 11

Table of Contents

Revision History

Version	*Date*	*Summary of Change*	*Author*
1.0	*Apr. 21, 2005*	*Initial Draft*	*Myeong-Jeong, Park*
1.1	*May 6, 2005*	*API Lists Change*	*Myeong-Jeong, Park*
1.2	*May 11, 2005*	*The return values for the CNetSession::sendAudio, CNetSession::sendControl function are updated.*	*Myeong-Jeong, Park*
1.3	*May 13, 2005*	*Member variables of CNetSession were added.* *They are the sequence, sec, usec and frameType.*	*Myeong-Jeong, Park*
1.4	*May 19, 2005*	*Description for the FCC_AUDIOSET is added and FCC_WINPOSSET is changed and FCC_PCONTROL command and the ControlID paramter of the CNetSession::sendControl are modified. And Appendix A is added.*	*Myeong-Jeong, Park*
1.5	*May 25, 2005*	*Description for the FCC_SERVERFPS is modified.* *The additional NETSESSION_RET_VIDEO_FPS_CHANGED for Video Event is added.*	*Myeong-Jeong, Park*
1.6	*May 26, 2005*	*A SendAudio(PlaybackAudio) limitation is added on NETSession_Init() function.*	*Myeong-Jeong, Park*
1.7	*June 10, 2005*	*NETSESSION_GetVersion() function and NVSVersion structure is added.* *A parameter, url, of NETSESSION_Init() is inserted.*	*Myeong-Jeong, Park*
1.8	*June 14, 2005*	*The "Other required files" section is changed : Static library*	*Myeong-Jeong Park*

		type is added. *Linux and WinCE platform for library deployment are added.*	
1.9	*June 15, 2005*	*The FCC_AVONOFF flag of the CNetSession::sendControl function for start/stop Audio/Video without reconnection is added.* *A return value for CNetSession::doStateMachine function is added ; that is NETSESSION_RET_TYPE_CHANGED.*	*Myeong-Jeong Park*
1.10	*June 16, 2005*	*The PCTL_ RESETDOME flag for CNetSession::sendControl option is added.* *All union members of CONTROLDATA::payload is updated.*	*Myeong-Jeong Park*
1.11	*June 24, 2005*	*NETSESSION_Init parameter description is changed. _connectType parameter is added*	*Sung-Jun Park*
1.12	*July 22, 2005*	*The Required SDK section is added*	*MyeongJeong Park*
1.13	*August 02, 2005*	*Control ID PCTL_TRANSPARENT is added.* *And the transparent_data member of the CONTROLDATA structure is added.*	*MyeongJeong, Park*
1.14	*August 19, 2005*	*Remove the Requred SDK Section, and some description in the NOTE section for windows developers.* *The library version was increased.*	*MyeongJeong, Park*
1.15	*Sept. 10, 2005*	*The description for the NVSVideoInfo::vbr is modified in detail.*	*MyeongJeong, Park*

1.16	*OCT. 13, 2005*	*The winPosition member of the CNetSession structure is inserted.* *FCC_SIMPLEWINPOSSET flag for CNetSession::sendControl() is added.*	*MyeongJeong, Park*
1.17	*OCT. 27, 2005*	*A sample program describing how to use transparent transmission function of the CNetSession::sendControl() is modified because of invalid usage.*	*MyeongJeong, Park*
1.18	*Jan. 11, 2006*	*Apply the software modification*	*Myeong-Jeong, Park*
1.19	*Mar. 20, 2006*	*Update the library's new additional function.*	*Myeong-Jeong, Park*
1.20	*Apr. 06, 2006*	*Update the library's new additional function.*	*Myeong-Jeong, Park*
1.21	*April 7, 2006*	*Update modification of the library.*	*Myeong-Jeong, Park*
1.22	*June 27, 2006*	*Add description for NETSESSION_RET_EVENT_VIDEOSIGNAL*	*Sung-Jun, Park*
2.00	*Dec. 19, 2008*	*The library version was increased.* *Modify description for camera model in the NSServerInfo structure.* *Add description for motion map data in the NETSESSION_RET_GET_CONTROL*	*Woo-Ram, Cho*
2.01	*Oct. 04, 2009*	*The library version was increased.* *NETSESSION_InitEx() function is added. (For 4CH NVS, channel selection parameter is added.)* *Add description for digital input data in the NETSESSION_RET_GET_CONT*	*Woo-Ram, Cho*

		ROL.	
2.02	*Sept. 11, 2009*	*The library version was increased.* *Add description for NETSESSION_RET_EVENT_DOUTPUT.*	*Woo-Ram, Cho*

Software Change History

Version	*Date*	*Summary of Change*	*Author*
1.0.0.1	*Apr. 21, 2005*	*Initial Draft*	*Sung-Jun, Park*
1.0.1.1	*May 13, 2005*	*A Few CNetSession's members were added. They are the sequence, sec, usec, and frametype.*	*Sung-Jun, Park*
1.0.2.1	*May 25, 2005*	*NETSESSION_RET_VIDEO_FPS_CHANGED for a video event is added. This is found at CNetSession::doStateMachine function.*	*Sung-Jun, Park*
1.0.2.2	*May 31, 2005*	*A bug fix of the CNetSession::close() function.*	*Sung-Jun, Park*
1.1.0.0	*June 10, 2005*	*A parameter, url, of the NETSESSION_Init() function is inserted.* *NETSESSION_GetVersion() API function and NVSVersion structure is added.*	*Sung-Jun, Park*
1.1.1.0	*June 16, 2005*	*The PCTL_RESETDOME flag of CNetSession::sendControl function is added.*	*Sung-Jun, Park*
1.2.0.1	*June 24, 2005*	*NETSESSION_Init parameter _isVideo, _isAudio, _isPlayBackAudio are replaced to connectType*	*Sung-Jun, Park*
1.2.0.2	*August 19, 2005*	*Bug fix that the library is blocked when trying to disconnect from an invalid server.*	*Sung-Jun, Park*
1.3.0.1	*OCT 13, 2005*	*A new member winPosition of the CNetSession structure.* *FCC_SIMPLEWINPOSSET flag for CNetSession::sendControlis added.*	*Sung-Jun, Park*
1.3.1.0	*OCT 21, 2005*	*The structure members of the*	*Sung-Jun, Park*

		SIMPLEWINPOSSET structure is modified(int -> unsigned short) *The VIVS_VSIGNAL flag value is modified to 0x40.*	
1.3.5.0	*Jan 08, 2006*	*A bug fix crashing application when receiving data from the video server after abnormal disconnection.*	*Sung-Jun, Park*
1.4.0.0	*Mar 20, 2006*	*Implemented an additional function that is receiving audio/video data only when event(motion/dio) is detected.*	*Sung-Jun, Park*
1.4.1.0	*April 6, 2006*	*Enables the library to communicate in duplex by serial interface.*	*Sung-Jun, Park*
1.4.2.0	*June 16, 2006*	*Bug patch for doStatMachine.* *In old version, sometimes reconnection problem is founded.*	*Sung-Jun, Park*
2.0.0.0	*June 07, 2008*	*The FCC_H264 is inserted for encordingType of NSVideoInfo structure.*	*Woo-Ram, Cho*
2.0.0.1	*July 10, 2008*	*The FCC_G726 is inserted for encordingType of NSAudioInfo structure.*	*Woo-Ram, Cho*
2.0.0.2	*August 04, 2008*	*A bug fix of video status for “3510”.*	*Woo-Ram, Cho*
2.0.0.3	*Nov. 28, 2008*	*The FCC_MOTIONDATA is inserted for motion map status in the NVS.*	*Woo-Ram, Cho*
2.0.0.4	*Oct. 04, 2009*	*NETSESSION_InitEx() function is added. (For 4CH NVS, channel selection parameter is added.)* *Add description for digital input data in the NETSESSION_RET_GET_CONTROL.*	*Woo-Ram, Cho*

2.0.0.5	*Sept. 11, 2009*	*The NETSESSION_RET_EVENT_DOUTPUT is inserted.*	*Woo-Ram, Cho*

Definitions and Acronyms

dll	*Dynamic Linking Library*
API	*Application Programming Interface*
SDK	*Software Development Kit*
NVS	*Network Video Streamer*
FOURCC	*FOUR Character Code(Visit website:http://www.fourcc.org)*
MPEG	*Motion Picture Experts Group : The MPEG standards are an evolving set of standards for video and audio compression and for multimedia delivery developed by the Moving Picture Experts Group.*
H.264	*H.264 is a standard for video compression, and is equivalent to MPEG-4 Part 10, or MPEG-4 AVC (for Advanced Video Coding).*
DivX	*DIgital Video eXpress. DivX is a video codec developed my DivXNetwroks Co.*
Codec	*COder and DECoder; COmpressor and DECompressor*
NTSC	*National Television Standards Committee : A kind of TV Protocol*
PAL	*Phase Alternation Line : Standard of Analog TV Display*
ADPCM	*Adaptive Differential Pulse-Code Modulation. One of audio compression types.*
G.726	*G.726 is an ITU-T ADPCM speech codec standard covering the transmission of voice.*
FPS	*Frames Per Second*
video source	*The video camera that the video stream comes from.*
channel	*Equivalent to the video source.*
SendAudio	*The client-side sending-audio.*

Description

The NVSNetLib Library is a network library meaning a heart of Cellinx SDK family. It is parsing TCP network packets from the Network Video Streamer (NVS from now) into video stream, audio stream and kinds of control messages.

The library APIs are easily understandable.

NOTE:

The NVSNetLib Library is a DLL-based library. We distribute the SDK with the necessary header files and examples written in C and C++.

To Windows developers. The windows sample program that is describing the APIs usage requires the Windows 2000 SDK® or later (the former name is the Windows Platform SDK®) to build. In addition, we are recommending to install the Direct 9.0 SDK or later as well.

Required Header

NVSNetLib.h

Other Required Files

Window System

Library Type	Required Files
Static Link Library	NVSNetLibStc.lib
Dynamic Link Library	NVSNetLib.lib NVSNetLib.dll

WinCE System

Library Type	Required Files
Static Link Library	NVSNetLibStc.lib
Dynamic Link Library	NVSNetLib.lib NVSNetLib.dll

Linux System

Library Type	Required Files
Static Link Library	NVSNetLib.a
Shared Library	NVSNetLib.so

Data structure lists:

NSVideoInfo

- NSAudioInfo
- NSServerInfo
- CNetSession
- NVSVersion

Function lists:

- NETSESSION_Init
- NETSESSION_InitEx
- NETSESSION_GetVersion

NSVideoInfo

The **NVSVideoInfo** structure defines the video information that the NVS server currently transfers of video stream.

```
struct NSVideoInfo {
            unsigned int        encodingType;
            unsigned char       frameMode;
            unsigned int        fps;
            unsigned char       vbr;
            unsigned int        bitrate;
            unsigned short      videoWidth;
            unsigned short      videoHeight;
            unsigned char       videoStatus1;
            unsigned char       videoStatus2;
            unsigned char       videoStatus3;
            unsigned char       videoStatus4;
            unsigned char       videoType;
            unsigned char       groupSize;
};
```

Members

encodingType

This indicates that the compressor type in FOURCC format.

Values	**Description**
FCC_MP4S	Microsoft MPEG-4
FCC_DX50	DivX MPEG-4 version 5
FCC_MJPG	Motion JPEG
FCC_MPG2	MPEG version 2
FCC_MPG1	MPEG version 1
FCC_H263	H.263 video codec.
FCC_H264	H.264 video codec.

frameMode

The *frameMode* indicates how the NVS server compresses video frames.

Values	**Description**
VIFM_IONLY	The NVS server will compress 'I' frames only
VIFM_IPFRAME	Only 'I' or 'P' frames will be compressed.
VIFM_IPB	All the frame types, 'I', 'P' and 'B', will be compressed.

fps

This indicates the number of frames that the NVS server captures per a second. The range is 1~30.

vbr

Describes the various bit-rate value. If this is zero, the bitrate is constant(CBR mode). If the *vbr* value takes 2 to 31(Q value), the bitrate is various(VBR mode). The smaller Q value, The better video quality.

goto the declaration.

bitrate

bit-rate in kilo bit per second. The following values are available.

Values	Description
BITRATE_4M	4 mega bps
BITRATE_2M	2 mea bps
BITRATE_1_5M	1.5 mega bps.
BITRATE_1M	1 mega bps
BITRATE_750K	750 kilo bps
BITRATE_500K	500 kilo bps
BITRATE_384K	384 kilo bps
BITRATE_256K	256 kilo bps
BITRATE_128K	128 kilo bps
BITRATE_64K	64 kilo bps
BITRATE_32K	32 kilo bps

goto the declaration.

videoWidth

frame width in pixel.

goto the declaration.

videoHeight

frame height in pixel.

goto the declaration.

videoStatus1

The video status of the channel #1. To get the value, do bitwise mask operation with the following flags.

Flag	Description
VIVS_VSIGNAL	If this flag indicates that the channel has video signal.

goto the declaration.

videoStatus2

The video status of the channel #2. The flags are same as *videoStatus1*. Not used on the NVS V20A models.

goto the declaration.

videoStatus3

The video status of the channel #3. The flags are same as *videoStatus1*. Not used on the NVS V20A models.

goto the declaration.

videoStatus4

The video status of the channel #4. The flags are same as *videoStatus1*. Not used on the NVS V20A models.

goto the declaration.

videoType

This member describes the video standard type. It can take one of the followings.

Flag	**Description**
VIVT_NTSC	The NVS server will send the NTSC-format video.
VIVT_PAL	The NVS server will send the PAL-format video.
VIVT_SECAM	The NVS server will send the SECAM-format video.

goto the declaration.

groupSize

The size of group of picture. One of 3, 15, 30, 60 and 90 is available.

goto the declaration.

Remarks

This structure is a member of the **NSServerInfo** structure.

See Also

NSServerInfo

Goto the table of contents.

NSAudioInfo

The **NSAudioInfo** structure defines the audio information about the audio stream that the NVS server currently transfers.

```
struct NSAudioInfo {
            unsigned int        encodingType;
            unsigned char       channel;
            unsigned char       bitPerSample;
            unsigned int        samplingRate;
};
```

Members

encodingType

This indicates the audio encoding type in FOURCC format. The following values are available.

FOURCC	Description
FCC_IMAACPCM	The NVS server is encoding IMA Adpcm
FCC_MSADPCM	The NVS server is encoding MS adpcm
FCC_PCM	The NVS server is encoding PCM Audio data.
FCC_G726	The NVS server is encoding G726.

channel

The audio channel. The available channels are :

Channels	Description
AUDIO_MONO	1 channel audio.
AUDIO_STEREO	2 channels audio.

bitPerSample

The audio bit-per-sample value. There are 4Bit, 8bit and 16bit for the value.

samplingRate

The audio sampling rate. There are 8000, 11025, 16000, 22050, 44100 and 48000 for the value.

Remarks

This structure is a member of the **NSServerInfo** structure.

See Also

NSServerInfo

Goto the table of contents.

NSServerInfo

The **NSServerInfo** structure informs how the NVS server is encoding video stream and audio stream before receiving them. The application will receive this structured information as soon as completing the connection successfully.

```
struct NSServerInfo {
            unsigned char           cameraModel[4];
            unsigned char           reserved;
            unsigned char           videoCount;
            unsigned char           audioCount;
            unsigned char           serverName[32];
            unsigned char           channelName[32];
            unsigned char           channelNumber;
            struct NSVideoInfo      videoInfo;
            struct NSAudioInfo      audioInfo;
};
```

Members

cameraModel

The NVS server model name. This is not null-terminated string but just 4 characters.

The 2 kinds of models are available :

“V20A” : The one-channel video server (MPEG4).

“V24A” : The 4-channel video server. Users can replace the positions of the output video channels.

“3510” : The one-channel video server (H.264).

reserved

Not used.

videoCount

The number of video sources. For 20A or 3510 the value is 1, but for V24A, it’s 4.

audioCount

The number of audio sources. The value is always 1.

serverName

This is the null-terminated string to specify the server name.

channelName

This is the null-terminated string to specify the channel name.

channelNumber

The number of channels that an application can choose.

videoInfo

This member indicates the video information. For more information, see the **NSVideoInfo** structure.

audioInfo

This member indicates the audio information. For more information, see the **NSAudioInfo** structure.

Remarks

This structure is a member of the **CNetSession** structure. And the structure is valid since receiving the **NETSESSION_RET_LOGIN_SUCCESS** value that the *doStateMachine* member function of **CNetSession** structure returns. If one of the elements is changed, the NVSNetLib library will update them.

See Also

NSVideoInfo, NSAudioInfo, CNetSession

Goto the table of contents.

NVSVersion

The **NVSVersion** structure defines the version of the NVSNetLib library.

```
typedef struct NVSVersion {
            int major;
            int minor;
            int release;
            int build;
} NVSVersion, *pNVSVersion;
```

Members

major

The major version number of the library.

minor

The minor version number of the library

release

The release member describes how many the library has been released.

build

The build number.

See Also

NETSESSION_GetVersion

Goto the table of contents.

CNetSession

The **CNetSession** structure defines kinds of properties and methods as C plusplus does.

```
typedef struct CNetSession {
        int                     status;
        int                     fps;
        unsigned int            sequence;
        unsigned int            sec;
        unsigned int            usec;
        unsigned char           frameType;
        unsigned int            dataSize;
        char                    *recvData;
        char                    *winPosition;
        int                     isVideo;
        int                     isAudio;
        int                     isPlayBackAudio;
        int                     isConfig;
        int                     isEvent;
        int                     isEventVideo;

        struct NSServerInfo     serverInfo;
        int                     (*close)(void *_this);
        int                     (*release)(void *_this);
        int                     (*sendAudio)(void *_this, char *_buf, unsigned int _size);
        int                     (*sendControl)(void *_this, unsigned int _type, char *_buf, unsigned int _size);
        int                     (*doStateMachine)(void *_this);
} CNetSession, *pCNetSession;
```

Members

status

It indicates the status of the library. This member can take one of the following values.

Values	**Description**
NETSESSION_STAT_INIT	The library is currently not initialized. So, the next action is initialization.
NETSESSION_STAT_CONNECTING	The connection is currently not established. So, the next action is connection.
NETSESSION_STAT_SEND_LOGIN	The connection is currently established but login is not negotiated. So, the next action is to negotiate login.

NETSESSION_STAT_WAIT_LOGIN	Now the library is trying to login. So, wait for the grant message until timeout.
NETSESSION_STAT_RECV	All things to do for negotiation completed successfully. The next action is to read streams or send audio.
NETSESSION_STAT_CLOSE	The next action is to close the library.
NETSESSION_STAT_SLEEP	The next action is to wait and sleep.

fps

It indicates the current fps, not the server's capturing fps. The range is from 0 to 30.

sequence

The sequential number of each frame. This *sequence* starts from zero since a connection is established.

sec

The server-captured time in second. This *sec* starts from zero since a connection is established.

usec

The server-captured time in micro-second. This *usec* starts from zero since a connection is established.

frameType

This indicates a MPEG-based video frame type. This is valid until the return value of DoStateMachine function is NETSESSION_RET_VIDEO. The value for this is 'I', 'P' or 'B'.

dataSize

The data size of audio or video stream that is stored in the *recvData* member. This member is filled when the library receives video or audio stream and .the *doStateMachine* member function of the **CNetSession** structure returns the NETSESSION_RET_GET_VIDEO for video stream or the NETSESSION_RET_GET_AUDIO for audio stream.

goto the declaration.

recvData

This is the pointer to the buffer that is storing audio or video stream data. This member is filled when the library receives video or audio stream and .the *doStateMachine* member function of the **CNetSession** structure returns the NETSESSION_RET_GET_VIDEO for video stream or the NETSESSION_RET_GET_AUDIO for audio stream.

goto the declaration.

winPosition

Pointer to the buffer that describes the window position information for the V24A video server model.

NOTE)

If this member is NULL, the video server is not V24A or does not support multi-channel.

This member is newly inserted as of the version of **1.3.0.0**.

You must type-cast to the **WINPOSSET** structure that is used by **FCC_WINPOSSET** for the **sendControl** function before using this.

isVideo

This member indicates that a video stream is on the connection to the NVS server. If not zero, the NVS server sends a video stream else, it doesn't.

goto the declaration.

isAudio

This member indicates that an audio stream is on the connection to the NVS server. If not zero, the NVS server sends an audio stream else, it doesn't.

goto the declaration.

isPlayBackAudio

This member indicates that an application can send audio to the NVS server. If this member is not zero the server is prepared to receive audio data, else the application shall not send.

The NVS Audio server allows only one connection. So, second SendAudio-connection trial will be denied. In that case, however the connection trial is successful, the **doStateMachine** function will return NETSESSION_RET_PARTIAL_LOGIN_SUCCESS and **isPlayBackAudio** will be zero.

goto the declaration.

isConfig

This member indicates that the library is able to change configuration settings.

goto the declaration.

isEvent

This member indicates the video server is ready to send only event information.

If this member is not 0, the NVS server will send only event information and no audio or video data.

Use the NETSESSION_LOGIN_TYPE_EVENT flag when calling the **NETSESSION_Init**() function.

This member is valid for the version 1.4.0.0 or later.

goto the declaration.

isEventVideo

This member indicates that the trigger system to motion or dio event is enabled or not. If the value is not zero, the NVS server is ready to do the function.

If this function is enabled, the NVS server transfers video/audio data only when event(motion/dio) is detected.

Use the NETSESSION_LOGIN_TYPE_EVENT_VIDEO flag when calling the **NETSESSION_Init**() function.

This member is valid for the version 1.4.0.0 or later.

goto the declaration.

serverInfo

The *serverInfo* structure defines the NVS server's information. The *serverInfo* structure is automatically refreshed whenever the NVS server status is changed by the library. When you refer the *serverInfo* structure, be aware that the *status* member is NETSESSION_STAT_RECV. For more information about the *serverInfo* data structure, see the **NSServerInfo** structure.

goto the declaration.

close

pointer to a function that closes the connection to the NVS server.

Parameters

_pThis

[IN] pointer to a **CNetSession** structure that calls this function.

Return value

If successful, it returns NETSESSION_OK else it returns NETSESSION_ERROR.

Example

Refer to the example at the end of the document.

Remark

This function pointer is assigned by the library when an application calls the **NETSESSION_Init** function. Do not override.

goto the declaration.

release

pointer to a function that releases the memory block allocated by the **NETSESSION_Init** function..

Parameters

_pThis

[IN] pointer to a **CNetSession** structure calling this function.

Return value

If successful, it returns NETSESSION_OK else it returns NETSESSION_ERROR.

Example

Refer to the example at the end of the document.

Remark

This function pointer is assigned by the library when an application calls the **NETSESSION_Init** function. Do not override.

goto the declaration.

sendAudio

pointer to a function that sends audio data.

Parameters

_pThis

[IN] pointer to a **CNetSession** structure that calls this function.

_buf

[IN] pointer to a buffer that contains audio data.

_size

[IN] the size of the *_buf* member.

Return value

If successful, it returns NETSESSION_OK else it returns the following values.

Values	**Description**
NETSESSION_RET_AU_INVALID_ARGUMENT	One or more arguments are invalid.
NETSESSION_RET_AU_INVALID_SOCKET	Socket handle is invalid.
NETSESSION_RET_AU_AUDIO_NOT_AVAILABLE	The NVS server is not ready to SendAudio or the client does not want to SendAudio.

NETSESSION_RET_AU_PACKET_IS_TOO_LARGE	The requested SendAudio Packet is too large.
NETSESSION_RET_AU_NETWORK_ERROR	Error occurred on sending packets. For getting error code, call WSAGetLastError() for windows or use errno for Linux.

Example

Refer to the example at the end of the document.

Remark

This function pointer is assigned by the library when an application calls the **NETSESSION_Init** function. Do not override.

The audio data must be compressed in MPEG Layer-3(MP3) before sending to the NVS server. We also distribute an mp3 encoder either named in 'lame mp3 encoder'. For more information about the encoder, see the manual.

If you send audio by calling the *sendAudio* function, you can listen to the sound on the NVS server box.

goto the declaration.

sendControl

pointer to a function that sends control data.

Parameters

_pThis

[IN] pointer to a **CNetSession** structure that calls the function.

_ type

[IN] This specifies a control code in FOURCC code. The available fourcc codes are at the remark.

_buf

[IN] pointer to a buffer that contains control data.

_size

[IN] the size of the *_buf* member.

Return value

If successful, it returns NETSESSION_OK else it returns the following values.

Values	**Description**
NETSESSION_RET_SC_INVALID_ARGUMENT	One or more arguments are invalid.
NETSESSION_RET_SC_INVALID_SOCKET	Internally used socket handle is invalid.
NETSESSION_RET_SC_QUEUE_FULL	The internal queue is full so wait till the queue is available.
NETSESSION_RET_SC_PACKET_IS_TOO_LARGE	The requested SendAudio Packet is too big.
NETSESSION_RET_SC_NETWORK_ERROR	Error occurred on sending packets. For getting error code, call WSAGetLastError() for windows or use errno for Linux.

goto the declaration.

Example

Refer to the example at the end of the document.

Remark

This function pointer is assigned by the library when an application calls the **NETSESSION_Init** function. Do not override. The control codes and the values are as following.

_type	_buffer	Description
FCC_STOPEVENTVIDEO	There is no additional data. set to NULL.	This control stops the NVS server transferring the triggered audio and video data. This control is valid for the version 1.4.0.0 or later.

FCC_PCONTROL	struct CONTROLDATA
	{
	unsigned short channel;
	unsigned short controlID;
	union {
	// omitted. Refer to the “NVSNetLib.h” for detail.
	}payload;
	};
		Controls the camera of specified video source specified in the *channel*.
		channel : The channel number to control.
		Channels : 1, 2, 3 or 4
		controlID : Control ID type.
		Look at the appendix for control ids.
		payload : *controlID*-specified data.
		Look at the APPENDIX A for more information.

FCC_CH_BIT	unsigned int bit_rate;	Changes the video bitrate. 32-4000
FCC_CH_RESOL	unsigned int resolution;	Changes the Video Resolution QCIF(1), QVGA(2), CIF(3), VGA(4), 4CIF(5), D1(6)
FCC_CH_FRAMEMODE	unsigned int frameType;	Changes the video frame type that the NVS server encodes. **VIFM_IONLY** : I-Frame only **VIFM_IPFRAME** : I and P frames **VIFM_IPB** : I, P and B frames
FCC_GET_FRAMEMODE	unsigned int getFrameType;	Changes the video frame type that the application will receive. **VIFM_IONLY** : I-Frame only **VIFM_IPFRAME** : I and P frames **VIFM_IPB** : I, P and B frames
FCC_SERVERFPS	unsigned int serverFPS;	Changes the frames per second that the NVS server captures. The value range is 1 to 15 and 30. The fps value from 16 to 29 results in 30 fps. The NVS server’s configuration is changed with such fps value though.

FCC_DIGITALOUT	unsigned int onDigitalOut;	This indicates whether to generate a digital output signal. Set zero to stop a digital output signal and non zero for a digital output signal.
FCC_CH_VIDEO_GROUP	unsigned int groupSize;	group size
FCC_CH_ENCODINGTYPE	unsigned int encodingType;	Changes the video encoding type. **FCC_MP4S** : Microsoft MPEG-4 **FCC_DX50** : DivX MPEG-4 version 5 **FCC_MJPG** : Motion JPEG **FCC_MPG2** : MPEG version 2 **FCC_MPG1** : MPEG version 1

FCC_AUDIOSET	struct AUDIOFORMATSET
	{
	DWORD encodingType;
	DWORD channel;
	DWORD bitPerSample;
	DWORD samplingRate;
	};
		Changes the audio setting
		encodingType:
		FCC_IMAACPCM : IMA ADPCM Audio.
		FCC_MSADPCM: Microsoft ADPCM Audio
		FCC_PCM : PCM Audio
		channel
		AUDIO_MONO : 2-channel stereo audio
		AUDIO_STEREO : 1-channel mono audio
		bitPerSample : 4, 8, 16
		samplingRate : Sampling rate in 1Kilo Hz unit.
		Set 8 for 8000Hz.
		Set 11 for 11025Hz
		Set 16 for 16000Hz
		Set 22 for 22050Hz
		Set 44 for 44100Hz
		Set 48 for 48000Hz

FCC_WINPOSSET	struct WINPOSSET
	{
	unsigned char x1;
	unsigned char y1;
	unsigned char w1;
	unsigned char h1;
	unsigned char x2;
	unsigned char y2;
	unsigned char w2;
	unsigned char h2;
	unsigned char x3;
	unsigned char y3;
		These are the coordination of output video channel. The channel is 1 to 4. The number at the end of each member describes the channel number of the video source.
		x : x-coordination. 0 to 255.
		y : y-coordination 0 to 255.
		w : size of the frame width. 0 to 255.
		h : size of the frame height. 0 to 255.
		C : the scale value. For the each channel.
		pip : pip embedded channel in pip mode.
		0 for ch#1. 1 for ch#2 and so forth.

	unsigned char w3;
	unsigned char h3;
	unsigned char x4;
	unsigned char y4;
	unsigned char w4;
	unsigned char h4;
	unsigned char C1;
	unsigned char C2;
	unsigned char C3;
	unsigned char C4;
	unsigned char pip;
	};	

FCC_AVONOFF	unsigned int avstate;	This value changes audio and video state and controls the start or stop of audio or video stream without disconnection. The value is a combination of the followings. **STYP_AUDIO** : Enable the NVS server audio streaming. **STYP_VIDEO** : Enable the NVS server video streaming. **STYP_SENDAUDIO** : Enable the NVS server audio listening; client will send audio stream.

FCC_SIMPLEWINPOSSET	struct SIMPLEWINPOSSET
	{
	unsigned char ntype;
	unsigned char host;
	unsigned char emb;
	unsigned char embLoc;
	unsigned char embScal;
	};
		ntype : this indicates how to align the channels for the multi-channel video server. Following values are available.
		SIMWPOS_CH1 : only CH#1 appears.
		SIMWPOS_CH2 : only CH#2 appears.
		SIMWPOS_CH3 : only CH#3 appears.
		SIMWPOS_CH4 : only CH#3 appears
		SIMWPOS_QUAD : all 4 channels appear.
		SIMWPOS_PIP : Picture in picture mode.
		host : the host channel. One of the pip options.
		The value range is 1 to 4.
		emb : the embedded channel.
		The value range is 1 to 4. exclusive to the *host*.
		embLoc : where to locate the embedded channel. Following values are available.
		SIMWPOS_EMBPOS_TOPLEFT

		- The embedded channel will be on the top left of the screen. **SIMWPOS_EMBPOS_TOPRIGHT** - The embedded channel will be on the top right of the screen. **SIMWPOS_EMBPOS_BOTTOMRIGHT** - The embedded channel will be on the bottom right of the screen. **SIMWPOS_EMBPOS_BOTTOMLEFT** - The embedded channel will be on the bottom left of the screen. embScale : how big is the embedded channel? Following values are available. **SIMWPOS_EMBSCALE_HALF** - the 1/2 scale. **SIMWPOS_EMBSCALE_THIRD** - the 1/3 scale. **SIMWPOS_EMBSCALE_QUARTER** - the 1/4 scale.

goto the declaration.

doStateMachine

pointer to a function that runs and executes the negotiation and protocol parsing choirs.

Parameters

_pThis

[IN] pointer to a **CNetSession** structure that calls this function.

Return values

The function's return value has one or more of following flags. To get the value as specifies below, you may do mask operation.

Flags	**Description**
NETSESSION_ERROR	The parameter _pThis is NULL or invalid.
NETSESSION_RET_NORMAL	No status. Just returns ok.
NETSESSION_RET_LOGIN_OK_MASK	Bitmask to retrieve the login specific flags.
NETSESSION_RET_LOGIN_SUCCESS	This flag is set when the login is successful.
NETSESSION_RET_GET_VIDEO	This flag indicates that the recvData buffer is filled with video data.
NETSESSION_RET_GET_AUDIO	This flag indicates that the recvData buffer is filled with audio data.
NETSESSION_RET_GET_ACK	This flag means that one of the isEvent or

	isEventVideo is not zero and the video server is idle on detecting motion or dio event. This value is returned every 3 seconds.
NETSESSION_RET_GET_EVENT	This flag means that the video server has detected motion or dio event. This return value is valid only when isEvent is not zero. This value is valid for the version 1.4.0.0 or later.
NETSESSION_RET_GET_EVENT_START	This flag means that the video server has detected motion or dio event and starts sending audio or video data until the user sends FCC_STOPEVENTVIDEO control data in the sendControl function. This return value is valid only when isEventVideo is not zero. This value is valid for the version 1.4.0.0 or later.
NETSESSION_RET_GET_EVENT_STOP	This flag means that the NVS server now stops transferring audio or video data. This return value is valid only when isEventVideo is not zero. This value is valid for the version 1.4.0.0 or later.
NETSESSION_RET_GET_CONTROL	This flag means that to retrieve control data from the NVS. FCC_SERIALDATA : The NVS server received serial data as a response from the camera controller. The recvData member forms the data as following.

NETSESSION_RET_GET_CONTROL

```
typedef struct SerialData{
        unsigned int      dwControlType;
        unsigned int      dwSerialType;
        unsigned int      dwDataSize;
        unsigned int      dwChannel;
        unsigned char     bSerialData[256];
} SerialData, *pSerialData;
```

	dwControlType : Indicates the data type if it is serial data or not. **FCC_SERIALDATA** : The received data was from the serial port of the NVS server. dwSerialType : Indicates where the data was from, The following values are available. **RS_232** : The received data was from RS232. **RS_485** : The received data was from RS485. dwDataSize : Indicates the size of the data buffer containing the serial data in byte size. dwChannel : Indicates which channel has sent the data. bSerialData : Pointer to a buffer that contains the serial data that the video server received. FCC_MOTIONDATA : To retrieve motion map status in the NVS. typedef struct MotionMapData{ unsigned int dwControlType; unsigned int dwDataSize; unsigned short motionData[256]; } MotionData, *pMotionData; dwControlType : Indicates the data type if it is motion map data or not. **FCC_MOTIONDATA** : The received data was from the motion map data of the NVS server. dwDataSize : Indicates the size of the data buffer containing the motion map data in byte size. bMotionData : Pointer to a buffer that contains the motion map data.

	FCC_DIGITALIN : To retrieve digital input status in the NVS.
	typedef struct DigitalInData{
	unsigned int dwControlType;
	unsigned int dwDataSize;
	unsigned int dwStatus;
	} DigitalInData, *pDigitalInData;
	dwControlType : Indicates the data type if it is digital input data or not.
	FCC_DIGITALIN : The received data was from the digital input status of the NVS server.
	dwDataSize : Indicates the size of the data buffer containing the digital input status in byte size.
	dwStatus : digital input status.

NETSESSION_RET_PARTIAL_LOGIN_SUCCESS	This flag indicates that the login is granted but the requested sending audio is disabled.
NETSESSION_RET_TYPE_CHANGED	This flag indicates that the audio or video state is changed. This means if audio stream is stopped or video stream is stopped or send-audio is stopped.
NETSESSION_RET_LOGIN_FAIL_MASK	Bitmask to retrieve the login-specific error.
NETSESSION_RET_LOGIN_FAIL_MAX	Login failed because maximum user limitation is exceeded.
NETSESSION_RET_LOGIN_FAIL_AUTH	Login failed because administrator’s id or password is wrong.
NETSESSION_RET_CONNECT_ERROR	Connection lost with unknown reason. Call the WSAGetLastError() for error code on Windows System or use the *errno* on Linux System.
NETSESSION_RET_CONNECTION_TIMEOUT	Time is out in trying to connect or login.
NETSESSION_RET_NETWORK_ERROR	Connection lost with network error. Call the WSAGetLastError() for error code on Windows System or use the *errno* on Linux System.
NETSESSION_RET_EVENT_MASK	Bitmask to retrieve the event values. Events are one or combined of followings.

NETSESSION_RET_EVENT_MOTION1	This flag is set when a motion has been detected on ch #1 video source.
NETSESSION_RET_EVENT_MOTION2	This flag is set when a motion has been detected on ch #2 video source.
NETSESSION_RET_EVENT_MOTION3	This flag is set when a motion has been detected on ch #3 video source.
NETSESSION_RET_EVENT_MOTION4	This flag is set when a motion has been detected on ch #4 video source.
NETSESSION_RET_EVENT_DINPUT	This flag is set when a digital input signal is detected.
NETSESSION_RET_EVENT_VIDEOSIGNAL	This flas is set only when video signal status is changed.
NETSESSION_RET_EVENT_DOUTPUT	This flag is set when a digital output signal is set.
NETSESSION_RET_VIDEO_CHANED_MASK	Bitmask to retrieve the video-changed notification. Flags are one or combined of followings.
NETSESSION_RET_VIDEO_ENCODING_CHANGED	This flag is set when the video’s compressor type is changed.
NETSESSION_RET_VIDEO_FRAMEMODE_CHANGED	This flag is set when the video’s frame mode is changed.
NETSESSION_RET_VIDEO_VBR_CHANGED	This flag is set when the video’s vbr value is changed.
NETSESSION_RET_VIDEO_BITRATE_CHANGED	This flag is set when the video’s bitrate is changed. Not set on MJPEG movie.
NETSESSION_RET_VIDEO_WIDTH_CHANGED	This flag is set when the video’s frame width is changed.
NETSESSION_RET_VIDEO_HEIGHT_CHANGED	This flag is set when the video’s frame height is changed.
NETSESSION_RET_VIDEO_GROUPSIZE_CHANGED	This flag is set when the video’s group-of-picture value is changed.
NETSESSION_RET_VIDEO_FPS_CHANGED	This event means that the server side fps is changed. This fps equivalent to sampling-fps.
NETSESSION_RET_AUDIO_CHANED_MASK	Bitmask to retrieve the audio-changed notification. Flags are one or combined of followings.
NETSESSION_RET_AUDIO_ENCODING_CHANGED	This flag is set when the audio’s compressor

	type is changed.
NETSESSION_RET_AUDIO_CHANNEL_CHANGED	This flag is set when the audio's number of channels is changed. (mono or stereo)
NETSESSION_RET_AUDIO_BPS_CHANGED	This flag is set when the audio's bit-per-sample value is changed.
NETSESSION_RET_AUDIO_SAMPLERATE_CHANGED	This flag is set when the audio's sampling rate value is changed.

Example

Refer to the example at the end of the document.

Remark

This function pointer is assigned by the library when an application calls the **NETSESSION_Init** function. Do not override.

goto the declaration.

Remarks

The pointer of the *recvData* member is assigned by the library. Do not allocate memory block in the *recvData*.

goto the declaration.

See Also

NETSESSION_Init

Goto the table of contents.

NETSESSION_Init

The **NETSESSION_Init** function initializes the NVSNetLib library and creates a CNetSession-typed object. All works in the library starts from the **NETSESSION_Init** function.

```
void * NETSESSION_Init(
            char *_address,
            unsigned short _port,
            char * _url,
            char *_id,
            char *_passwd,
            unsigned char _connectType
);
```

Parameters

_address

Pointer to a null-terminated string that specifies the NVS server.

_port

Port number that the NVS is listening for client's connection. If the NVS server's port is not port-forwarded, the *_port* is 1852.

_url

Pointer to a null-terminated string that specifies the alias of the NVS server. This parameter can be NULL in direct-connected environment. This parameter must not be SET when connecting to NVS server over a relay server.

_id

Pointer to a null-terminated string that specifies the user id for authentication. For a guest, the *_id* is "guest".

_passwd

Pointer to a null-terminated string that specifies the password relevant to the _id for authentication. For the guest account, the *_passwd* is "guest".

_connectType

bitwise value that indicate connection type. The following values can be one or more in combination.

Flags	Description
NETSESSION_LOGIN_TYPE_VIDEO	If this flag is set, the NVS server streams video.
NETSESSION_LOGIN_TYPE_AUDIO	If this flag is set, the NVS server streams audio.
NETSESSION_LOGIN_TYPE_AUDIO_PLAYBACK	If this flag is set, the NVS server prepares receive audio data for playback.
NETSESSION_LOGIN_TYPE_EVENT_VIDEO	If this flag is set, the NVS server does not transfer audio and video until any event(motion or dio) is detected.

	This flag must be combined with NETSESSION_LOGIN_TYPE_VIDEO or NETSESSION_LOGIN_TYPE_AUDIO.
NETSESSION_LOGIN_TYPE_EVENT	If this flag is set, the NVS server transfers only event information of whether motion or dio event is detected or not. This flag must be set alone. Do not combine other flags.

for example, if you want to get video and audio, set _connetType to 0x03

The value will be set like the following figure.

Bit Order	7	6	5	4	3	2	1	0
Bit value			eventVideo	event	reserved	sendAudio	audio	video

Return Value

If the **NETSESSION_Init** function completed successfully, it returns a pointer to an object of **CNetSession**, else it returns NULL.

Remarks

Note that an application must call periodically the *doStateMachine* function of the **CNetSession** structure to run the library properly. All things, parsing protocol or negotiations with NVS server, are implemented in it.

Do not call the malloc function or the malloc-like memory allocating functions because the **NETSESSION_Init** function allocates the memory block internally. And call the release member function of the **CNetSession** structure to release the memory block allocated by **NETSESSION_Init**. Calling the free function may cause the application crashed.

The NVS Audio server allows only one connection. A second connection will get access denial. In that case, however the connection trial is successful, the **CNetSession::doStateMachine** function will return NETSESSION_RET_PARTIAL_LOGIN_SUCCESS and **CNetSession::isPlayBackAudio** will be zero.

The NETSESSION_LOGIN_TYPE_EVENT_VIDEO and NETSESSION_LOGIN_TYPE_EVENT flag of the *_connectType* is valid on the Version 1.4.0.0 or later.

The NETSESSION_LOGIN_TYPE_EVENT_VIDEO flag of the *_connectType* is a long-waited function that the NVS server transfer a/v data only when motion or dio event is detected. If the flag is set, and the event is detected, the CNetSession::DoStateMachine function returns the NETSESSION_RET_GET_EVENT_START and then you can get the audio/video data. If the CNetSession::DoStateMachine returns the NETSESSION_RET_GET_EVENT_STOP, it means the NVS server does not transfer audio or video data any more. To stop the NVS server transferring triggered audio or video data, send FCC_STOPEVENTVIDEO control data though CNetSession::sendControl function.

The NVS server accepts only one connection with the flag. If more than 1 client with the flag is trying to

connect to the video server, it will be denied the access with the CNetSession::DoStateMachine returning the NETSESSION_RET_LOGIN_FAIL_MAX
The NETSESSION_LOGIN_TYPE_EVENT flag means the NVS server transfers only event information of whether the video server detected motion or dio event or not. And the CNetSession::DoStateMachine function returns the NETSESSION_RET_GET_EVENT when the event is detected.

Application using older than 1.4.0.0 version is required to be recompiled to use the NETSESSION_LOGIN_TYPE_EVENT_VIDEO or NETSESSION_LOGIN_TYPE_EVENT flag.

See Also

CNetSession

Goto the table of contents.

NETSESSION_InitEx

The **NETSESSION_InitEx** function initializes the NVSNetLib library and creates a **CNetSession**-typed object. All works in the library starts from the **NETSESSION_InitEx** function.

This function used to multi channel NVS server.

```
void * NETSESSION_InitEx(
        char *_address,
        unsigned short _port,
        unsigned char _which_ch,
        char * _url,
        char *_id,
        char *_passwd,
        unsigned char _connectType
);
```

Parameters

_address

Pointer to a null-terminated string that specifies the NVS server.

_port

Port number that the NVS is listening for client's connection. If the NVS server's port is not port-forwarded, the *_port* is 1852.

_which_ch

This value is channel selection parameter.

_url

Pointer to a null-terminated string that specifies the alias of the NVS server. This parameter can be NULL in direct-connected environment. This parameter must not be SET when connecting to NVS server over a relay server.

_id

Pointer to a null-terminated string that specifies the user id for authentication. For a guest, the *_id* is "guest".

_passwd

Pointer to a null-terminated string that specifies the password relevant to the _id for authentication. For the guest account, the *_passwd* is "guest".

_connectType

bitwise value that indicate connection type. The following values can be one or more in combination.

Flags	**Description**
NETSESSION_LOGIN_TYPE_VIDEO	If this flag is set, the NVS server streams video.
NETSESSION_LOGIN_TYPE_AUDIO	If this flag is set, the NVS server streams audio.
NETSESSION_LOGIN_TYPE_AUDIO_PLAYBACK	If this flag is set, the NVS server prepares receive

	audio data for playback.
NETSESSION_LOGIN_TYPE_EVENT_VIDEO	If this flag is set, the NVS server does not transfer audio and video until any event(motion or dio) is detected. This flag must be combined with NETSESSION_LOGIN_TYPE_VIDEO or NETSESSION_LOGIN_TYPE_AUDIO.
NETSESSION_LOGIN_TYPE_EVENT	If this flag is set, the NVS server transfers only event information of whether motion or dio event is detected or not. This flag must be set alone. Do not combine other flags.

for example, if you want to get video and audio, set _connetType to 0x03

The value will be set like the following figure.

Bit Order	7	6	5	4	3	2	1	0
Bit value			eventVideo	event	reserved	sendAudio	audio	video

Return Value

If the **NETSESSION_InitEx** function completed successfully, it returns a pointer to an object of **CNetSession,** else it returns NULL.

Remarks

Note that an application must call periodically the *doStateMachine* function of the **CNetSession** structure to run the library properly. All things, parsing protocol or negotiations with NVS server, are implemented in it.

Do not call the malloc function or the malloc-like memory allocating functions because the **NETSESSION_InitEx** function allocates the memory block internally. And call the release member function of the **CNetSession** structure to release the memory block allocated by **NETSESSION_InitEx**. Calling the free function may cause the application crashed.

The NVS Audio server allows only one connection. A second connection will get access denial. In that case, however the connection trial is successful, the **CNetSession::doStateMachine** function will return NETSESSION_RET_PARTIAL_LOGIN_SUCCESS and **CNetSession::isPlayBackAudio** will be zero.

The NETSESSION_LOGIN_TYPE_EVENT_VIDEO and NETSESSION_LOGIN_TYPE_EVENT flag of the *_connectType* is valid on the Version 1.4.0.0 or later.

The NETSESSION_LOGIN_TYPE_EVENT_VIDEO flag of the *_connectType* is a long-waited function that the NVS server transfer a/v data only when motion or dio event is detected. If the flag is set, and the event is detected, the CNetSession::DoStateMachine function returns the NETSESSION_RET_GET_EVENT_START and then you can get the audio/video data. If the CNetSession::DoStateMachine returns the NETSESSION_RET_GET_EVENT_STOP, it means the NVS server does not transfer audio or video data any more. To stop the NVS server transferring triggered audio or video data, send FCC_STOPEVENTVIDEO control data though CNetSession::sendControl function.

The NVS server accepts only one connection with the flag. If more than 1 client with the flag is trying to connect to the video server, it will be denied the access with the CNetSession::DoStateMachine returning the NETSESSION_RET_LOGIN_FAIL_MAX

The NETSESSION_LOGIN_TYPE_EVENT flag means the NVS server transfers only event information of whether the video server detected motion or dio event or not. And the CNetSession::DoStateMachine function returns the NETSESSION_RET_GET_EVENT when the event is detected.

Application using older than 1.4.0.0 version is required to be recompiled to use the NETSESSION_LOGIN_TYPE_EVENT_VIDEO or NETSESSION_LOGIN_TYPE_EVENT flag.

See Also

CNetSession

Goto the table of contents.

NETSESSION_GetVersion

The **NETSESSION_GetVersion** function is used to get the library version. The version is represented as the **NVSVersion** structure.

```
NVSVersion NETSESSION_GetVersion();
```

Parameters

There are no parameters.

Return Value

The function returns the library version. The version consists of the **NVSVersion** structure.

See Also

NVSVersion

Goto the table of contents.

API Example Source Code

This example illustrates how to use the NVSNetLib library.

```
void usage(int argc, char **argv) {
   printf("USAGE : %s\n", argv[0]);
   printf("%s server_address server_port id passwd\n", argv[0]);
}

int main(int argc, char **argv) {
   pCNetSession       netSession;
   int                status = 0, ret = 0, sendtimeout = 0;
   BYTE               audioBuf[1024];
   HANDLE             hFile =    NULL;
   DWORD              dwByteRead = 0;
   BOOL               bFinishLoop = 0;
   char          connectType = 0x07;     // All of Video, Audio, SendAudio.

   if(argc < 5) {
      usage(argc, argv);
      exit(0);
   }

   // file open
   hFile = CreateFile("test.mp3", GENERIC_READ, 0, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
   if( hFile == NULL )
   {
      printf("Error opening file ... \n");
      return 0;
   }

   printf("Now connecting to %s[%d] with iD:[%s] pass:[%s]\n", argv[1], atoi(argv[2]), argv[3],
argv[4] );

   // this, address, port, id, pass, isVideo, isAudio, isPlaybackAudio
   netSession = (pCNetSession)NETSESSION_Init(argv[1], atoi(argv[2]), NULL, argv[3], argv[4],
connectType );
   if(netSession == NULL) {
      printf("Error on initializing the library...\n");
      return 0;
   }

   while(!bFinishLoop)
   {
      Sleep(1);
      status = netSession->doStateMachine(netSession);

      // Checking login
      switch( status & NETSESSION_RET_LOGIN_OK_MASK )
      {
      case NETSESSION_RET_LOGIN_SUCCESS:
      case NETSESSION_RET_PARTIAL_LOGIN_SUCCESS:
         printf("Login Success 1\n");
         printf("isVideo        : %d\n", netSession->isVideo);
         printf("isAudio        : %d\n", netSession->isAudio);
         printf("isPlaybackAudio   : %d\n", netSession->isPlayBackAudio);
         break;
      case NETSESSION_RET_GET_VIDEO:
         // do something with the video data.
         printf("[VIDEO RECEIVED] : %d bytes\n", netSession->dataSize);
         break;
      case NETSESSION_RET_GET_AUDIO:
         // do something with the audio data.
         printf("[AUDIO RECEIVED] : %d bytes\n", netSession->dataSize);
         break;
      default:
         // Login failure?? why?
         switch( status & NETSESSION_RET_LOGIN_FAIL_MASK )
         {
         case NETSESSION_RET_LOGIN_FAIL_MAX:
            printf("Too many connected users...\n");
            Sleep(3000);
            break;
         case NETSESSION_RET_LOGIN_FAIL_AUTH:
            printf("Id or password is wrong...\n");
            Sleep(3000);
            break;
         case NETSESSION_RET_CONNECT_ERROR:
            printf("Unknown connection error...\n");
            Sleep(3000);
            break;
         case NETSESSION_RET_CONNECTION_TIMEOUT:
            printf("Time is over on connection...\n");
            Sleep(3000);
            break;
         case NETSESSION_RET_NETWORK_ERROR:
            printf("Network error occured...\n");
            Sleep(3000);
            break;
         }
         break;   // error?? start from the begining.
      }

      // checking that the server is able to receive audio data.
      // when NETSESSION_RET_PARTIAL_LOGIN_SUCCESS flag is set in the status,
isPlayBackAudio will be false at all times.
      if(netSession->isPlayBackAudio)
      {
         if( sendtimeout ++ > 5 )
         {
            sendtimeout = 0;
            switch( netSession->status )
            {
            // The library is connected successfully...
            case NETSESSION_STAT_RECV:
               if( !ReadFile( hFile, audioBuf, 512, &dwByteRead, NULL ) )
               {
                  printf("ReadFile Error!!! : %d(0x%.8x)\n", GetLastError(), GetLastError() );
                  bFinishLoop = TRUE;
                  break;
               }

               // End of file?? then go to the begging of the file.
               if( dwByteRead == 0 ){    SetFilePointer( hFile,    0,    0, FILE_BEGIN    ); continue; }

               // sending audio buffer...
```

```
                    if( netSession->sendAudio(netSession, audioBuf, dwByteRead) == 0 )
                    {
                        printf(".");
                    }
                    // if the send queue is full??
                    else
                    {
                        SetFilePointer( hFile,   -512,   0, FILE_CURRENT   );
                        printf(";");
                    }
                    break;
                default:
                        printf("*");
                    break;
                }
            }
        }
/*
      - To access and refer the netSession->serverInfo
          1. the netSession->doStatusMachine() must return the
NETSESSION_RET_LOGIN_SUCCESS.
          2. the netSession->status must be NETSESSION_STAT_RECV.
          3. the netSession->serverInfo will be refreshed automatically.

      - If a mask operation of NETSESSION_RET_GET_VIDEO the result of the netSession-
>doStatusMachine() is NETSESSION_RET_GET_VIDEO,
                the video data is filled in netSession->recvData, and its size is in netSession-
>dataSize.
      - If a mask operation of NETSESSION_RET_GET_AUDIO the result of the netSession-
>doStatusMachine() is NETSESSION_RET_GET_AUDIO,
                the video data is filled in netSession->recvData, and the buffer size is in netSession-
>dataSize.
    */
    }

    netSession->close(netSession);
    // freeing allocated memory block...
    netSession->release(netSession );
    netSession = NULL;
    return 0;
}
```

Required Files

Include <Windows.h> header file for file i/o function.

Include <CLXNetLib.h> header file for the library

Include <stdio.h> header file for the ansi c functions

Goto the table of contents.

APPENDIX A – Daemon Control ID

The Daemon-Control IDs are used to control the Pan-Tilt, Zoom, Focus and so forth to the specified camera source.

First of all, before using the daemon-control command, the *controlID* parameter of the *sendControl* pointer-function member of the **CNetSession** structure must be set to FCC_PCONTROL.

The CONTROLDATA structure that is used to send daemon-control data.

```
typedef struct _tagCONTROLDATA
{
		USHORT channel;
		USHORT controlID;
		union{
				unsigned char brighness;
				unsigned char contrast;
				unsigned char saturation;
				unsigned char sharpness;
				unsigned char whitebalance;
				unsigned char blc;
				unsigned char zoom;
				unsigned char focus;
				unsigned char exposure;
				unsigned char iris;
				unsigned char pan;
				unsigned char tilt;
				unsigned char light;
				unsigned char preset_set;
				unsigned char preset_move;
				unsigned char preset_clear;
				char	focus_cont;
				struct
				{
						char pan;
						char tilt;
						char zoom;
				}ptz_cont;
				unsigned char transparent_data[256];
		} payload;
}CONTROLDATA, *PCONTROLDATA, *LPCONTROLDATA
```

Members

channel

This member specifies a channel number of a video stream source.

controlID

This indicates how the user controls the video source.

payload

union-typed data structure. The value is various depending on the *controlID*..

Remark

The value range of the *channel* member is 1 to 4. Because the V20A NVS video server has only one channel, this must be always 1 for the V20A model. However the V24A model has 4 channels, you can choose the channel number from 1 to 4 to control the video source.

The *controlID* determine the size or length of the *payload*. You should send only the required size but not the full size of the *payload*.

The following table explains the control IDs.

controlID	**Description**
PCTL_BRIGHTNESS	This controls the brightness of the video output. Use the *brightness* member. Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_CONTRAST	This controls the contrast of the video output. Use the *contrast* member. Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_SATURATION	This controls the saturation of the video output. Use the *saturation* member Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_SHARPNESS	This controls the sharpness of the video output. Use the *sharpness* member Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_WHITEBALANCE	This controls the white balance of the video output. Use the *whitebalance* member Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_BLC	This controls the backlight compensation of the camera. Use the *blc* member. Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_ZOOM	This controls the zoom of the camera. Use the *zoom* member. The value is degree unlikely to PCTL_PTZ_CONT. This value is scaled by the NVS Server.
PCTL_FOCUS	This controls the focus of the video output. Use the *focus* member. The value is degree unlikely to PCTL_PTZ_CONT. This value is scaled by the NVS Server.
PCTL_EXPOSURE	This controls the *exposure* of the camera. Use the *exposure* member. Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_IRIS	This controls the degree of closure or open of the iris of the camera. Use the *iris* member

	Input value range is 0 to 255. This value is scaled by the NVS Server.
PCTL_PAN	This controls the pan of the camera. Use the *pan* member. The value is degree unlikely to PCTL_PTZ_CONT. This value is scaled by the NVS Server.
PCTL_TILT	This controls the tilt of the camera. Use the *tilt* member. The value is degree unlikely to PCTL_PTZ_CONT. This value is scaled by the NVS Server.
PCTL_LIGHT	This controls the light attached to the camera to turn on or off. Use the *light* member.
PCTL_PRESET_SET	This command sets the current coordinate as a preset. Use the *preset_set* member to allocate preset number. The input value range is various for each camera model, Refer to the camera manual.
PCTL_PRESET_MOVE	This command lets the camera move to the specified preset point. Use the *preset_move* member. To move to a preset point, first of all, set a preset point with PCTL_PRESET_SET command.
PCTL_PRESET_CLEAR	This command clears the pre-defined preset point. The input value indicates the preset number.
PCTL_FOCUS_CONT	This controls speed of focus of the camera. Use the *focus_cont* member Input value range is -127 to +127. If the value is not zero, the focusing will be operated without stopping. To stop the operation, call this command with zero. The minus value indicates focus out and the plus value indicates focus in.
PCTL_PTZ_CONT	This controls the pan, tilt, zoom and their speed simultaneously of the camera. The greater absolute value, the faster speed. One call of this command with non-zero value causes the camera to start operating. To stop the operation, set all to zero. Use the *ptz_cont* member. Following table describes the members of the *ptz_cont* structure. <table><tr><td></td><td>*pan*</td><td>*tilt*</td><td>*zoom*</td></tr><tr><td>Value > 0</td><td>Start to pan right</td><td>Start to tilt up</td><td>Start to zoom in</td></tr><tr><td>Value < 0</td><td>Start to pan left</td><td>Start to tilt down</td><td>Start to zoom out</td></tr><tr><td>Value == 0</td><td>stop</td><td>stop</td><td>stop</td></tr></table>
PCTL_RESETDOME	Reset pan-tilt zoom module. This flag does not require any data.

PCTL_TRANSPARENT	This command enables users to control the camera and its module directly. You can write the camera-module-specific protocol data to the RS-485 serial port to control PTZ or IRIS etc… Fill the camera-module-specific-protocol data into the *transparent_data* member. And the *_size* parameter of the CNetSession::sendControl function is the sum of 4 (the size of the *controlID* and *channel*) and the written data size in the *transparent_data*. NOTE : The transparent data must be the camera-module-understandable protocol data. EXAMPLE)

```
unsigned char ptz[10] = { 0x00, 0x01, …. };  // must be specified in the protocol of the camera module. maximum 256Bytes
CONTROLDATA ctrl;
ctrl.controlID = PCTL_TRANSPARENT;
ctrl.channel = 1;
memcpy( ctrl.payload.transparent_data, ptz, sizeof(ptz) );
pNetSession->sendControl(pNetSession, FCC_PCONTROL, (char*)&ctrl, 4 + sizeof(ptz));
```