Device Network SDK

Programming User Manual

V4.2

(For Decoder)

The information in this documentation is subject to change without notice and does not represent any commitment on behalf of HIKVISION. HIKVISION disclaims any liability whatsoever for incorrect data that may appear in this documentation. The product(s) described in this documentation are furnished subject to a license and may only be used in accordance with the terms and conditions of such license.

This documentation is issued in strict confidence and is to be used only for the purposes for which it is supplied. It may not be reproduced in whole or in part, in any form, or by any means or be used for any other purpose without prior written consent of HIKVISION and then only on the condition that this notice is included in any such reproduction. No information as to the contents or subject matter of this documentation, or any part thereof, or arising directly or indirectly therefrom, shall be given orally or in writing or shall be communicated in any manner whatsoever to any third party being an individual, firm, or company or any employee thereof without the prior written consent of HIKVISION. Use of this product is subject to acceptance of the HIKVISION agreement required to use this product. HIKVISION reserves the right to make changes to its products as circumstances may warrant, without notice.

This documentation is provided "as-is," without warranty of any kind.

Please send any comments regarding the documentation to:
overseasbusiness@hikvision.com

Find out more about HIKVISION at www.hikvision.com

Index

1 SDK Overview

The device network SDK is developed based on private network communication protocol, and it is designed for the remote connection and configuration of embedded devices. This document is mainly for decoder, and the main device types are listed as below:

DS-6300D(-JX), DS-6400HD(-JX/-T) series decoder

This document introduces only the major function supported by decoder, and please get more information about other function and related structures from "Device Network SDK Programming Manual.chm".

The device network SDK has both Windows and Linux version.

1. Windows version supports Windows7/XP/2000/2003/Vista(32bit), and it has the files:

Network Communication Library	HCNetSDK.h	head file
	HCNetSDK.lib	LIB file
	HCNetSDK.dll	DLL file
Qos Library	QosControl.dll	DLL file
RTSP Communication Library	StreamTransClient.dll	DLL file
Software Decode Library	PlayM4.h	head file
	PlayCtrl.lib	LIB file
	PlayCtrl.dll	DLL file
Encapsulation Transformation Library	SystemTransform.dll	DLL file
Hardware decode Library	DataType.h and DecodeCardSdk.h	head file
	DsSdk.lib	LIB file
	DsSdk.dll	DLL file

2. Linux version supports the system(32bit) that gcc-v is 4.1 or above. The tested system have RedHat AS 5/6, (Fedora)FC10/12, CentOS 5, SUSE 10, openSUSE 11, and Ubuntu 9.04/10.04. The SDK has the files:

Network Communication Library	hcnetsdk.h	head file
	libhcnetsdk.so	SO file

Qos Library	libQosControl.so	SO file
RTSP Communication Library	libStreamTransClient.so	SO file
Software Decode Library	playsdkpu.h	head file
	libm4play.so	SO file
Encapsulation Transformation Library	libSystemTransform.so	SO file

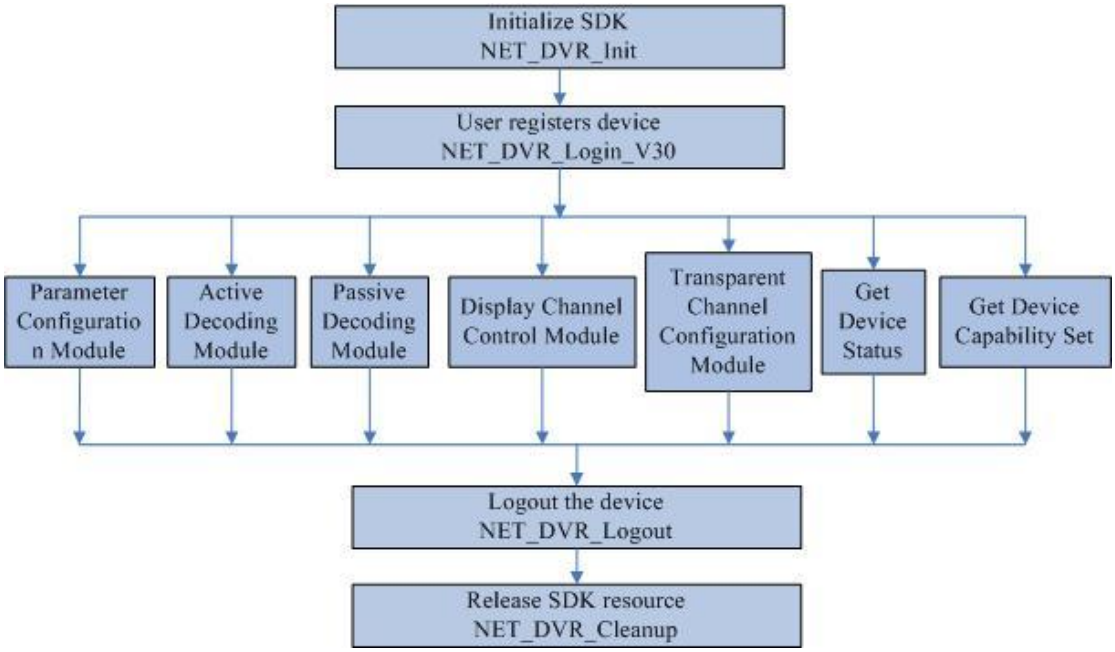
HCNetSDK is required to be loaded for client development, and the other '.dll' files are optional components.

- The Network Communication Library is the main functional part of the device network SDK. It is used for communication between the client and devices, including remote control & configuration, video stream acquiring and handling, etc; and Network communication library will dynamically loading RTSP communication library, Software decoding library, Hardware decoding library, etc. Network communication library combines a lot of functions from the Software decoding library and Hardware decoding library to facilitate the programming work. However, it is suggested the users to get video stream from 'HCNetSDK.dll', and call relative APIs in the Software decoding library or Hardware decoding library directly if you want to build a system with more complete functions, or in a more flexible way.
- The 'QosControl' library is stream bitrate control library, used for push mode SDK.
- RTSP Communication Library only supports IP devices. Users need to load this component for operations like streaming from products which support RTSP protocol.
- Software Decoding Library is used for decoding real-time video stream (remote live view), playback files, etc. It has included standard stream decoding function. If users needs to play real-time stream or recoding data and display(i.e. the second structure parameter play handle of NET_DVR_RealPlay_V30 interface set to effective), must load this component. However, if users just need to use it for capturing data, then do external operation, needn't load this component, this way is more flexible.
- Encapsulation transformation library function can be divided into two pieces: one is converting standard stream data to private encapsulation format stream data. When users need to capture private format stream data from products supporting RTSP protocol(that is setting callback function of NET_DVR_RealPlay_V30 interface for capturing data or call NET_DVR_SetRealDataCallBack interface to capture data), must load this component. Another is converting standard stream data to other package format, such as 3GPP,PS and so on. For example, when users need to capture specific package format real-time stream data from products supporting RTSP protocol(corresponding interface is NET_DVR_SaveRealData), must load this component.
- Hardware Decoding Library can only be used when there is MDI card installed in the PC, and it can output video or video matrix to analog monitors. For decoder, this library is not required.

2 API Calling Procedure

Notes: The part in dashed box is optional and will not affect the function and use of other process and modules.

2.1 The Calling Procedure of Decoder



Fuction modules of multi-channel decoder include display channel control, parameter configuration, dynamic decoding, passive decoding, transparent channel configuration, getting device status information and getting device capability set modules. All modules need to register user to the device, and the user ID returned from NET_DVR_Login_V30 is used as parameter of other APIs.

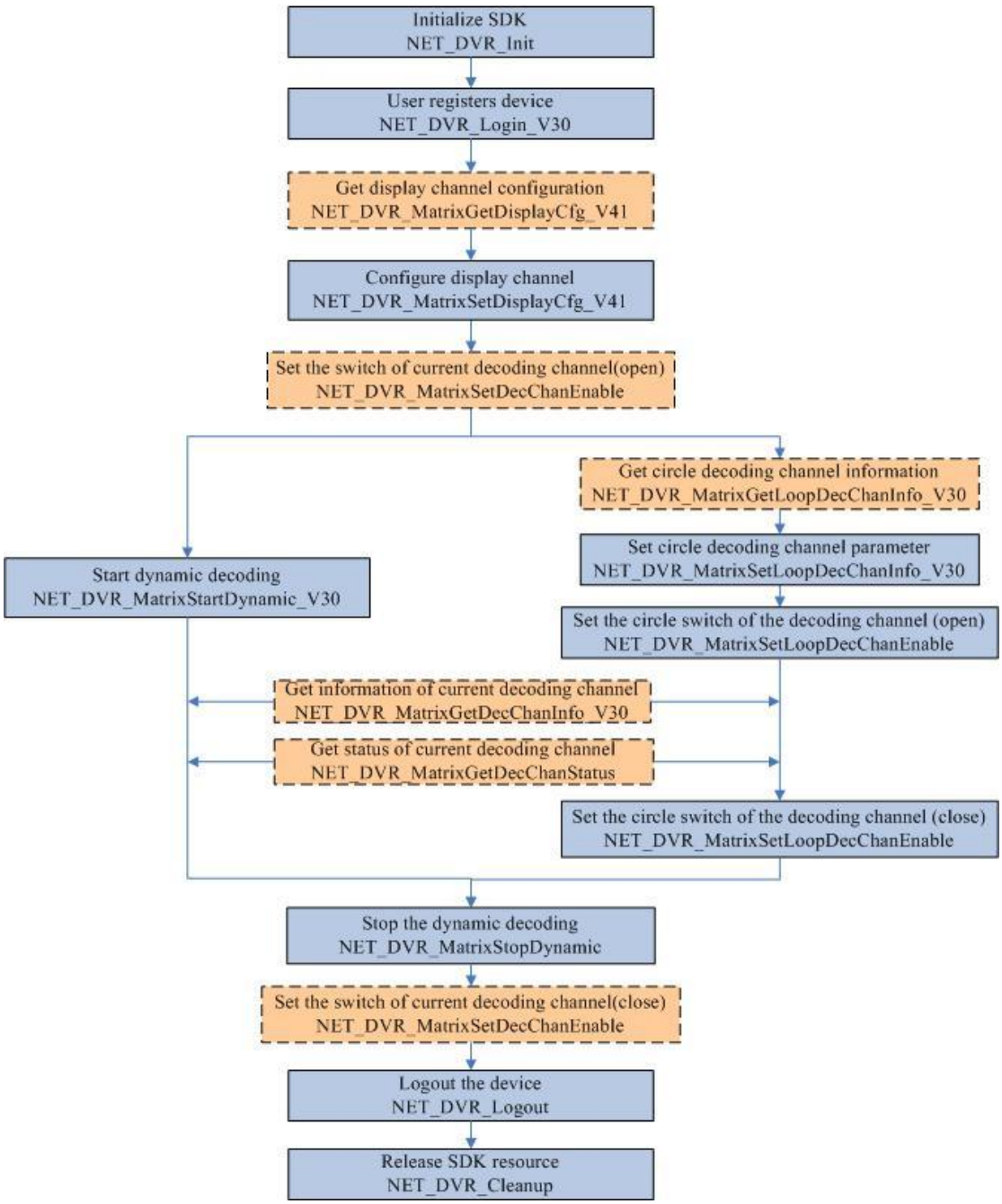
- Display channel control module: configuration of each parameter of display channel, audio turn on & off, and zoom control of child window. For details, please refer to "Configuration and Control of Display Channel".
- Parameter configuration module: to configure the basic parameters of the multi-channel decoder, the related APIs: NET_DVR_GetDVRConfig and NET_DVR_SetDVRConfig. For details, please refer to “Parameter Configuration”.
- Active decoding module: the decoder gets the stream data from encoder devices actively, then decodes the data. The related function has: 1) get the parameter of dynamic decoding; 2)control the decoding, including the starting and stopping the dynamic decoding and circle decoding, controlling the playback of remote files, and getting the status of decoding; 3) upload LOGO to the decoder. For details, please refer to “Active Decoding Procedure”.
- Passive decoding module: start decoding, send data and stop decoding of passive decoding channels. For details, please refer to “Passive Decoding Procedure”.
- Transparent channel configuration module: configure related parameters of transparent

channel. For details, please refer to “Transparent Channel Configuration”.

- Get device status information: It supports to get the status information of decoding, alam input, alarm output, and voice talk by calling NET_DVR_MatrixGetDeviceStatus_V41.
- Get device capability set: It supports to get the capability information ofdisplay and decoding by calling NET_DVR_GetDeviceAbility.

2.2 Active Decoding Procedure

2.2.1 Decode Real-time Stream



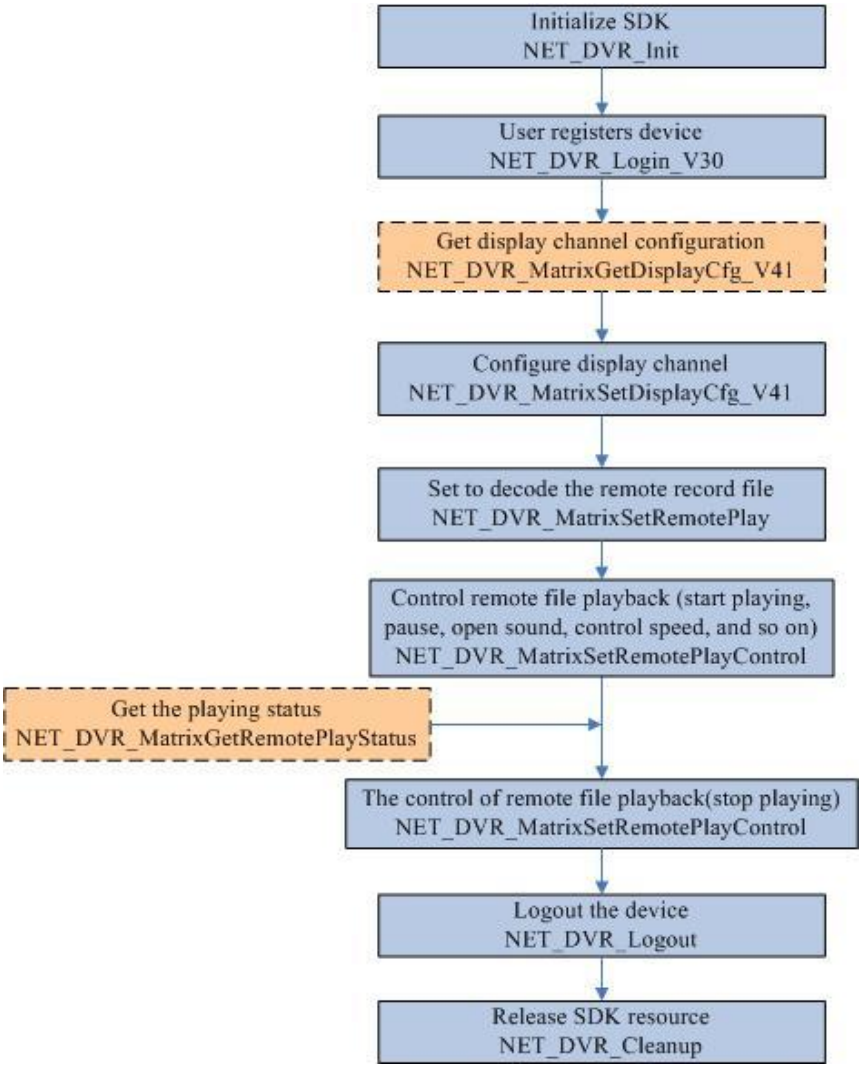
- After login the decoder, it requires to set the display channel firstly. Please set the decoding channel associated with the display channel, otherwise, it is not able to start decoding

normally. The related APIs: NET_DVR_MatrixGetDisplayCfg_V41, NET_DVR_MatrixSetDisplayCfg_V41.

- Call NET_DVR_MatrixStartDynamic_V30 to start dynamic decoding, the decoder will get stream form the encoder device and decode the data.
- It supports to set circle decoding, by calling NET_DVR_MatrixSetLoopDecChanInfo_V30 to set circle group and calling NET_DVR_MatrixSetLoopDecChanEnable to start circle decoding.

Example Code

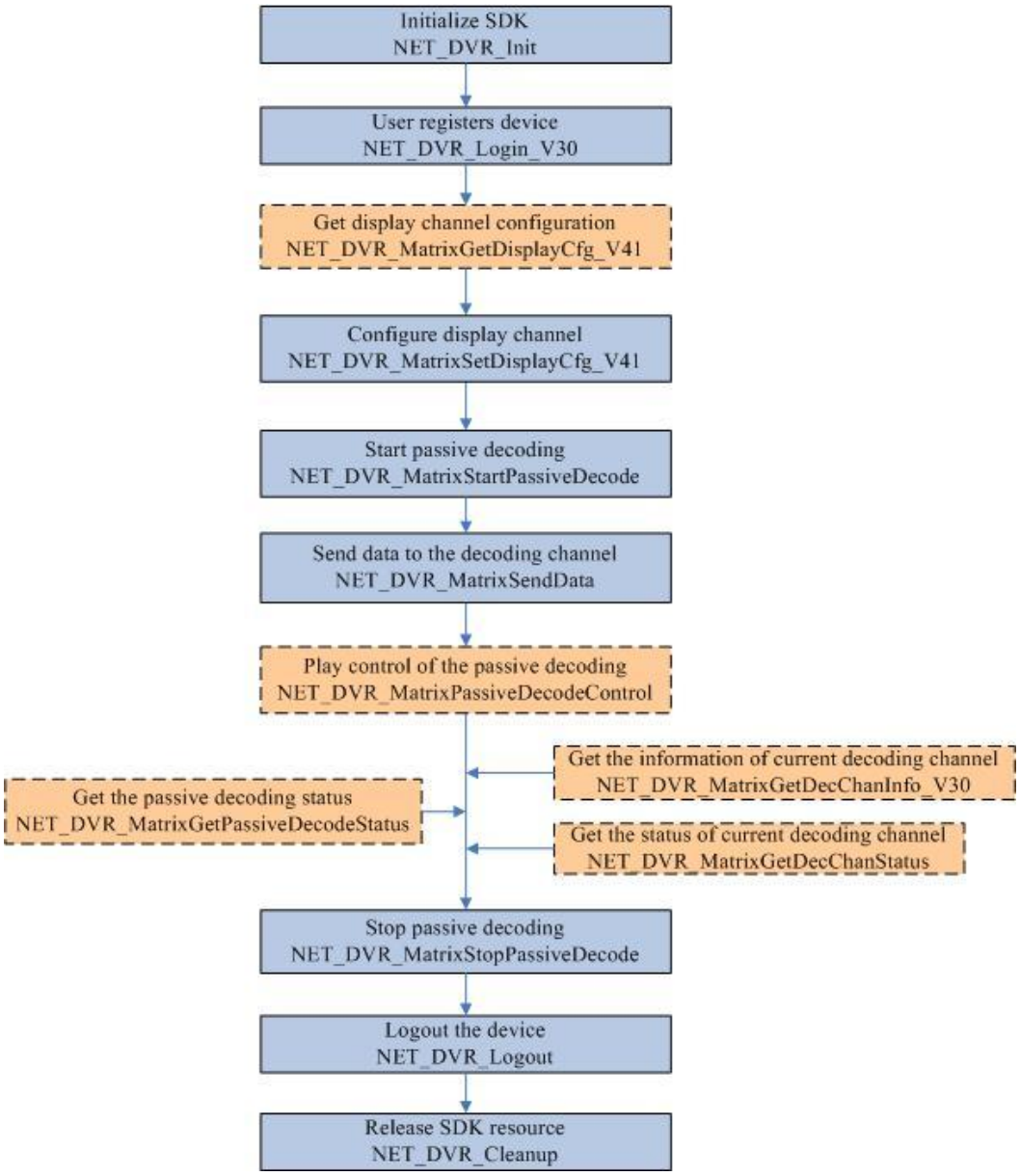
2.2.2 Remote File Playback



- After login the decoder, it requires to set the display channel firstly. Please set the decoding channel associated with the display channel, otherwise, it is not able to start decoding normally. The related APIs: NET_DVR_MatrixGetDisplayCfg_V41, NET_DVR_MatrixSetDisplayCfg_V41.
- Set to decode remote record files: firstly, please call NET_DVR_MatrixSetRemotePlay to set playback by time or by file name, and then call NET_DVR_MatrixSetRemotePlayControl to start encoding.

Example Code

2.3 Passive Decoding Procedure



- After login the decoder, it requires to set the display channel firstly. Please set the decoding channel associated with the display channel, otherwise, it is not able to start decoding normally. The related APIs: NET_DVR_MatrixGetDisplayCfg_V41, NET_DVR_MatrixSetDisplayCfg_V41.
- After calling NET_DVR_MatrixStartPassiveDecode to start passive decoding, please call NET_DVR_MatrixSendData to send data to the decoding channel. The data to be decoded can be get from remote device or read from record file, and the size of data for each sending should be less than 30K bytes.
- Decoding control: pause, fast play, slow play, open or close sound, clear buffer, and so on. Related APIs: NET_DVR_MatrixPassiveDecodeControl.

Example Code

3 API Calling Example

3.1 Example Code of Dynamic Decoding

3.1.1 Decode Real-time Stream

Related procedure chart

```
#include <stdio.h>
#include <iostream>
#include "Windows.h"
#include "HCNetSDK.h"
using namespace std;

void main() {

   //---------------------------------------
   //Initialize SDK
   NET_DVR_Init();
   //Set connect time and reconnect time
   NET_DVR_SetConnectTime(2000, 1);
   NET_DVR_SetReconnect(10000, true);

   //---------------------------------------
   // Login the device (Login the decoder)
   NET_DVR_DEVICEINFO_V30 struDeviceInfo;
   memset(&struDeviceInfo, 0, sizeof(NET_DVR_DEVICEINFO_V30)); //The structure to save device information
   LONG lUserID = NET_DVR_Login_V30("172.0.0.100", 8000, "admin", "12345", &struDeviceInfo);

   if (lUserID < 0)
   {
            if (NET_DVR_GetLastError() == NET_DVR_PASSWORD_ERROR)//Password error
            {
                     ..... //Handle the error message
            }
            else if(NET_DVR_GetLastError() == NET_DVR_OVER_MAXLINK)
                              //The count of connection to the device over the limit
            {
                     ..... // Handle the error message
            }
            ..... // Handle other error message
   }
```

```
  //Get display capability of the decoder
    NET_DVR_MATRIX_ABILITY m_matrixability;
    NET_DVR_GetDeviceAbility(lUserID, MATRIXDECODER_ABILITY, NULL, 0, (char*)&m_matrixability,
sizeof(NET_DVR_MATRIX_ABILITY));

    //Configure the display channel
    DWORD DispChanNum=1;//Display channel, can get it from capability set
    NET_DVR_MATRIX_VOUTCFG VoutCfg;
    if(!NET_DVR_MatrixGetDisplayCfg_V41(lUserID, DecChanNum, &VoutCfg))
    {
                ..... // Handle the error message
    }
    VoutCfg.dwWindowMode = 4; //Set window to 4 screens
    VoutCfg.byJoinDecChan[0] = 1;
        //The decoding channel associated with the upper left screen is set to channel 1
    ..... //To set other display channels

    if(!NET_DVR_MatrixSetDisplayCfg_V41(lUserID, DecChanNum, &VoutCfg))
    {
                ..... // Handle the error message
    }

    //Active decoding(includes dynamic decoding and circle decoding)
    int DecChanNum = 1;//Decoding channel number
    DWORD dec = 0;

    if(!NET_DVR_MatrixGetDecChanEnable(lUserID, DispChanNum, &dec)) //Get the switch of decoding channels
    {
                ..... // Handle the error message
    }

    dec = 1; //Open the decoding channel: 0- close, 1- open
    if(!NET_DVR_MatrixSetDecChanEnable(lUserID, DecChanNum, dec))
    // Set the switch of decoding channels, if set to close, the channel will stop decoding
    {
                ..... // Handle the error message
    }
    // Dynamic decoding
    NET_DVR_PU_STREAM_CFG dt;
    dt.struDevChanInfo.struIP.sIpV4 = "172.0.0.101";//IP address of front-end device(encoder device)
    dt.struDevChanInfo.wDVRPort = 8000; //Port number of front-end device
    dt.struDevChanInfo.byChannel = 1;//Channel number
    dt.struStreamMediaSvrCfg.byValid = 0;
    //Whether enable to get stream from stream server: 0- disable, not 0-enable
```

```
…… //Set other parameters of dynamic decoding

if(!NET_DVR_MatrixStartDynamic_V30(lUserID, DecChanNum, &dt))    //Start dynamic decoding
{
        CString tmp;
        tmp.Format("Error: NET_DVR_MatrixStartDynamic = %d\n", NET_DVR_GetLastError());
        //Get error code
        AfxMessageBox(tmp);
}

//Get decoding status
NET_DVR_MATRIX_DEC_CHAN_STATUS m_DecChanStatus;
memset(&m_DecChanStatus, 0, sizeof(NET_DVR_MATRIX_DEC_CHAN_STATUS));
if(!NET_DVR_MatrixGetDecChanStatus(lUserID, DecChanNum, &m_DecChanStatus))
//Get the status of the decoding channel
{
        …… // Handle the error message
}
NET_DVR_MATRIX_DEC_CHAN_INFO_V30    m_DecChanInfo;
memset(&m_DecChanInfo, 0, sizeof(NET_DVR_MATRIX_DEC_CHAN_INFO_V30));
if(!NET_DVR_MatrixGetDecChanInfo_V30(lUserID, DecChanNum, &m_DecChanInfo))
//Get the information of the decoding channel
{
        …… // Handle the error message
}

//Other operation

if(!NET_DVR_MatrixStopDynamic(lUserID, DecChanNum)) //Stop dynamic decoding
{
        CString tmp;
        tmp.Format("Error: NET_DVR_MatrixStartDynamic = %d\n", NET_DVR_GetLastError());
        //Get error code
        AfxMessageBox(tmp);
}

//Loop decoding
NET_DVR_MATRIX_LOOP_DECINFO_V30    m_MatLoopDec;
memset(&m_MatLoopDec, 0, sizeof(NET_DVR_MATRIX_LOOP_DECINFO_V30));
NET_DVR_MatrixGetLoopDecChanInfo_V30(lUserID, DecChanNum, &m_MatLoopDec);
//Get parameters of loop decoding channel

CString m_DVRIP = "172.0.0.101";
sprintf(m_MatLoopDec.struchanConInfo.struDecChanInfo.struIP.sIpV4,"%s", m_DVRIP);
```

```
    //IP address of the device to be decoded
    …… // Set other parameters of loop decoding channel
    NET_DVR_MatrixSetLoopDecChanInfo_V30 (lUserID, DecChanNum, &m_MatLoopDec);
    //Set the parameter of loop decoding channel

    DWORD chanNum = 0;
    NET_DVR_MatrixGetLoopDecChanEnable(lUserID, DecChanNum, &chanNum);
    //Get decoding switch of current channel, if chanNum=0, it is closed; if    chanNum=1, it is open

    chanNum = 1; //Open switch of decoding
    NET_DVR_MatrixSetLoopDecChanEnable(lUserID, DecChanNum, chanNum);
    /*Close the switch of current decoding channel. If the loop switch is closed, the decoding channel stopped the
loop and switch to dynamic decoding*/

    NET_DVR_MatrixGetLoopDecEnable(lUserID, &chanNum);
    //Get the decoding switch of all channels, indicated by byte: 0- closed, 1- open
    // E.g. chanNum&0x01==0 means the channel no.1 is closed

    dec = 0; //Close the decoding channel: 0- close, 1- open
    if(!NET_DVR_MatrixSetDecChanEnable(lUserID, DecChanNum, dec))
    // Set the switch of decoding channels, the channel stops decoding
    {
                ……// Handle the error message
    }

    // Logout
    NET_DVR_Logout(lUserID);
    // Release SDK resource
    NET_DVR_Cleanup();
    return;
}
```

3.1.2 Decode Remote File Recorded in the Device

Related procedure chart

```
#include <stdio.h>
#include <iostream>
#include "Windows.h"
#include "HCNetSDK.h"
using namespace std;

void main() {

    //---------------------------------------
```

```
//Initialize SDK
NET_DVR_Init();
//Set connect time and reconnect time
NET_DVR_SetConnectTime(2000, 1);
NET_DVR_SetReconnect(10000, true);

//---------------------------------------
// Login the device
NET_DVR_DEVICEINFO_V30 struDeviceInfo;
memset(&struDeviceInfo, 0, sizeof(NET_DVR_DEVICEINFO_V30));//The structure to save device information
LONG lUserID = NET_DVR_Login_V30("172.0.0.100", 8000, "admin", "12345", &struDeviceInfo);

if (lUserID < 0)
{
        if (NET_DVR_GetLastError() == NET_DVR_PASSWORD_ERROR)//Password error
        {
                ..... // Handle the error message
        }
        else if(NET_DVR_GetLastError() == NET_DVR_OVER_MAXLINK)
                        // The count of connection to the device over the limit
        {
                ..... // Handle the error message
        }
        ..... // Handle other error message
}
// Get display capability of the decoder
NET_DVR_MATRIX_ABILITY m_matrixability;
NET_DVR_GetDeviceAbility(lUserID, MATRIXDECODER_ABILITY, NULL, 0, (char*)&m_matrixability,
sizeof(NET_DVR_MATRIX_ABILITY));

// Configure the display channel
DWORD DispChanNum=1;// Display channel, can get it from capability set
NET_DVR_MATRIX_VOUTCFG VoutCfg;
if(!NET_DVR_MatrixGetDisplayCfg_V41(lUserID, DecChanNum, &VoutCfg))
{
        ..... // Handle the error message
}
VoutCfg.dwWindowMode = 1; // Set window to 4 screens
VoutCfg.byJoinDecChan[0] = 1;
//The decoding channel associated with the upper left screen is set to channel 1
..... //To set other display channels

if(!NET_DVR_MatrixSetDisplayCfg_V41(lUserID, DecChanNum, &VoutCfg))
{
```

```
        …… // Handle the error message
    }

    int DecChanNum = 1; //Decoding channel number

    if(!NET_DVR_MatrixGetDecChanEnable(lUserID, DispChanNum, &dec)) //Get the switch of decoding channels
    {
        …… // Handle the error message
    }

    dec = 1; //Open the decoding channel: 0- close, 1- open
    if(!NET_DVR_MatrixSetDecChanEnable(lUserID, DecChanNum, dwEnable))
    // Set the switch of decoding channels, if set to close, the channel will stop decoding
    {
        …… // Handle the error message
    }

    //Playback remote file in the front-end device (encoder device)
    NET_DVR_MATRIX_DEC_REMOTE_PLAY m_struPlay;
    m_struPlay. sDVRIP = "172.0.0.101"; //IP address of front-end device(encoder device)
    m_struPlay.wDVRPort = m_PlayBackPort; //Port number of front-end device
    m_struPlay.byChannel = (BYTE)m_PlayBackChan; //Channel number of the front-end device to be decoded
    …… //Set other parameters of playback
    NET_DVR_MatrixSetRemotePlay(lUserID, DecChanNum, &m_struPlay); //Configure the remote playback

    //Start playback
    NET_DVR_MatrixSetRemotePlayControl(lUserID, DecChanNum, NET_DVR_PLAYSTART, 0, NULL);
    //Open sound
    NET_DVR_MatrixSetRemotePlayControl(lUserID, DecChanNum, NET_DVR_PLAYSTARTAUDIO, 0, NULL);
    NET_DVR_MatrixSetRemotePlayControl(lUserID, DecChanNum, …, 0, NULL);//Other playback control
    NET_DVR_MATRIX_DEC_REMOTE_PLAY_STATUS m_struState;
    NET_DVR_MatrixGetRemotePlayStatus(lUserID, DecChanNum, &m_struState); //Get playback status

    //Stop decoding
    NET_DVR_MatrixSetRemotePlayControl(lUserID, DecChanNum, NET_DVR_PLAYSTOP, 0, NULL);

    //Logout
    NET_DVR_Logout(lUserID);
    // Release SDK resource
    NET_DVR_Cleanup();
    return;
}
```

3.2 Example Code of Passive Decoding

Related procedure chart

```
#include <stdio.h>
#include <iostream>
#include "Windows.h"
#include "HCNetSDK.h"
using namespace std;

void main() {
   //---------------------------------------
   //Initialize SDK
   NET_DVR_Init();
   //Set connect time and reconnect time
   NET_DVR_SetConnectTime(2000, 1);
   NET_DVR_SetReconnect(10000, true);

   //---------------------------------------
   // Login the device
   NET_DVR_DEVICEINFO_V30 struDeviceInfo;
   memset(&struDeviceInfo, 0, sizeof(NET_DVR_DEVICEINFO_V30)); //The structure to save device information
   LONG lUserID = NET_DVR_Login_V30("172.0.0.100", 8000, "admin", "12345", &struDeviceInfo);

   if (lUserID < 0)
   {
            if (NET_DVR_GetLastError() == NET_DVR_PASSWORD_ERROR) //Password error
            {
                     …… // Handle the error message
            }
            else if(NET_DVR_GetLastError() == NET_DVR_OVER_MAXLINK)
                        //The count of connection to the device over the limit
            {
                     …… // Handle the error message
            }
            ……// Handle other error message
   }
   //Get display capability of the decoder
   NET_DVR_MATRIX_ABILITY m_matrixability;
   NET_DVR_GetDeviceAbility(lUserID, MATRIXDECODER_ABILITY, NULL, 0, (char*)&m_matrixability,
sizeof(NET_DVR_MATRIX_ABILITY));

   //Configure the display channel
   DWORD DispChanNum=1; //Display channel, can get it from capability set
   NET_DVR_MATRIX_VOUTCFG VoutCfg;
```

```
    if(!NET_DVR_MatrixGetDisplayCfg_V41(lUserID, DecChanNum, &VoutCfg))
    {
                ..... // Handle the error message
    }
    VoutCfg.dwWindowMode = 1; //Set window to 4 screens
    VoutCfg.byJoinDecChan[0] = 1;
        //The decoding channel associated with the upper left screen is set to channel 1
    ......//To set other display channels

    if(!NET_DVR_MatrixSetDisplayCfg_V41(lUserID, DecChanNum, &VoutCfg))
    {
                ..... // Handle the error message
    }
    int DecChanNum = 1;// Decoding channel number
    if(!NET_DVR_MatrixGetDecChanEnable(lUserID, DispChanNum, &dec)) // Get the switch of decoding channels
    {
                ..... // Handle the error message
    }
    dec = 1; // Open the decoding channel: 0- close, 1- open
    if(!NET_DVR_MatrixSetDecChanEnable(lUserID, DecChanNum, dwEnable))
    // Set the switch of decoding channels, if set to close, the channel will stop decoding
    {
                ..... // Handle the error message
    }
    //Passive decoding
    DWORD m_PassivePort = 8000;
    LONG lPassiveModeHandle = -1; //The handle of passive decoding
    NET_DVR_MATRIX_PASSIVEMODE m_PassiveMode;
    m_PassiveMode.wPassivePort = m_PassivePort;
                            //The port number of UDP, when the transmission mode is TCP, it is defaulted to 8000.
    ..... //Other parameters of passive decoding
    lPassiveModeHandle = NET_DVR_MatrixStartPassiveDecode(lUserID, DecChanNum, &m_PassiveMode);
    //Start passive decoding
    NET_DVR_MatrixSendData(lPassiveModeHandle, pSendBuf, dwBufSize);
    //Send data to the decoder, pSendBuf is the buffer that saves the data, and dwBufSize is the size of the data

    NET_DVR_MatrixStopPassiveDecode(lPassiveModeHandle); //Stop passive decoding
    //Logout
    NET_DVR_Logout(lUserID);
    // Release SDK resource
    NET_DVR_Cleanup();
    return;
}
```

4 API Description

4.1 SDK Initialization

4.1.1 Initialize SDK: NET_DVR_Init

API: BOOL NET_DVR_Init()
Parameters: None
Return: Return TRUE on success, FALSE on failure.
Remarks: This API is used to initialize SDK. Please call this API before calling any other API.

Return to index

4.1.2 Release SDK resource: NET_DVR_Cleanup

API: BOOL NET_DVR_Cleanup()
Parameters: None
Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.
Remarks: This API is used to release SDK resource. Please calling it before closing the program.

Return to index

4.2 Get Error Message

4.2.1 Get the error code of last operation: NET_DVR_GetLastError

API: DWORD NET_DVR_GetLastError()
Parameters:
Return: The error code of last operation.
Remarks: Return the error code. Generally, there are 3 different types of error information: error of network communication library, error of RTSP library, and error of software/hardware decoding library, see detail to **macro definition of error code**.

Return to index

4.2.2 Get the error message of last operation: NET_DVR_GetErrorMsg

API: char* NET_DVR_GetErrorMsg(LONG *pErrorNo)

Parameters: [out] pErrorNo The pointer of the error code number

Return: The pointer that saves the error message. Please call NET_DVR_GetLastError to get the error code.

Remarks: Generally, there are 3 different types of error information: error of network communication library, error of RTSP library, and error of software/hardware decoding library, see detail to **macro definition of error code**.

Return to index

4.3 Login the Device

4.3.1 Loin the device: NET_DVR_Login_V30

API: LONG NET_DVR_Login_V30(char *sDVRIP, WORD wDVRPort, char *sUserName, char *sPassword,LPNET_DVR_DEVICEINFO_V30 lpDeviceInfo)

Parameters:

[in] Sdvrip	IP address of the device
[in] wDVRPort	Port number of the devic
[in] sUserName	User name
[in] sPassword	Password
[out] lpDeviceInfo	Device information

Return: Return -1 if it is failed, and other value is the value of returned user ID. The user ID is unique, and next operations should be realized through this ID. Please call NET_DVR_GetLastError to get the error code.

Remarks: Decoder supports 32 different user names and 128 users login at the same time. SDK supports 512 * login. UserID is incremented one by one, from 0 to 511 and then return to 0. Logout and NET_DVR_Cleanup will not initialize the UserID to 0.

Return to index

4.3.2 Logout: NET_DVR_Logout

API: BOOL NET_DVR_Logout(LONG lUserID)

Parameters: [in] lUserID User ID, the return value of NET_DVR_Login_V30

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: It is suggested to call this API to logout.

Return to index

4.4 Get the capability set of the device

4.4.1 Get the capability set: NET_DVR_GetDeviceAbility

API: BOOL NET_DVR_GetDeviceAbility(LONG lUserID, DWORD dwAbilityType, char* pInBuf, DWORD dwInLength, char* pOutBuf, DWORD dwOutLength)

Parameters:

[in] lUserID	The return value of NET_DVR_Login_V30
[in] dwAbilityType	Capability type, details listed below
[in] pInBuf	Pointer of the input buffer (according to description mode of ability parameter, defined by device, it supports XML text or structure format)
[in] dwInLength	Length of input buffer
[out] pOutBuf	Pointer of the output buffer (according to description mode of ability set, defined by device, it supports XML text or structure format)
[in] dwOutLength	Length of output buffer

dwAbilityType Macro Definition	Value	Implication
MATRIXDECODER_ABILITY	0x200	Display and decoding capability of multi-channel decoder
MATRIXDECODER_ABILITY_V41	0x260	Decoder capability set (extended)

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The definitions of pInBuf are different according to different devices, and the input and output parameter format when getting different types of capabilities are defined as below:

Macro Definition	Type of Ability	pInBuf	pOutBuf
MATRIXDECODER_ABILITY	Get display and decoding capability of multi-channel decoder	None	NET_DVR_MATRIX_ABILITY
MATRIXDECODER_ABILITY_V41	Get decoder capability set (extended)	None	NET_DVR_MATRIX_ABILITY_V41

Return to index

4.5 Configuration and Control of the Display Channel

4.5.1 Get the information of display channel:

NET_DVR_MatrixGetDisplayCfg_V41

API:	BOOL NET_DVR_MatrixGetDisplayCfg_V41 (LONG lUserID, LONG dwDispChanNum, LPNET_DVR_MATRIX_VOUTCFG lpVoutCfg)	
Parameters:	[in]lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDispChanNum	Display channel, please get it from capability set
	[out] lpVoutCfg	Display channel information, please kindly refer to the structure: NET_DVR_MATRIX_VOUTCFG
Return:	Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.	
Remarks:		

Return to index

4.5.2 Configure the display channel:

NET_DVR_MatrixSetDisplayCfg_V41

API:	BOOL NET_DVR_MatrixSetDisplayCfg_V41(LONG lUserID, LONG dwDispChanNum, LPNET_DVR_MATRIX_VOUTCFG lpVoutCfg)	
Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDispChanNum	Display channel, please get it from capability set
	[in] lpVoutCfg	Display channel configuration, please kindly refer to the structure: NET_DVR_MATRIX_VOUTCFG
Return:	Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.	
Remarks:		

Return to index

4.5.3 Control the display channel: NET_DVR_MatrixDiaplayControl

API:	BOOL NET_DVR_MatrixDiaplayControl(LONG lUserID, DWORD dwDispChanNum, DWORD dwDispChanCmd, DWORD dwCmdParam)	
Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDispChanNum	Display channel, please get it from capability set
	[in] dwDispChanCmd	Display channel control command, see to the following list

[in] dwCmdParam Command parameter, please set to 0

dwDispChanNum Macro Definition	Value	Implication
DISP_CMD_ENLARGE_WINDOW	1	Enlarge one window of the display channel
DISP_CMD_RENEW_WINDOW	2	Resume the window of the display channel

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.6 Parameter Configuration

4.6.1 Get configuration of the device: NET_DVR_GetDVRConfig

API: BOOL NET_DVR_GetDVRConfig(LONG lUserID, DWORD dwCommand,LONG lChannel, LPVOID lpOutBuffer, DWORD dwOutBufferSize, LPDWORD lpBytesReturned)

Parameters:
[in] lUserID User ID, the return value of NET_DVR_Login_V30
[in] dwCommand Configuration command, please kindly refer to the following list
[in] lChannel Channel number, if the channel parameter is not required, lChannel is invalid, and set it as 0xFFFFFFFF
[out] lpOutBuffer The buffer to save the received data
[in] dwOutBufferSize The size of the buffer (unit: byte), it can't be 0
[out] lpBytesReturned The size of the returned buffer, it can't be NULL

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The structures and command numbers are different according to the various getting functions, and they are listed as below:

Macro Definition of dwCommand	Description	lChannel	lpOutBuffer	Value
NET_DVR_GET_NETCFG_OTHER	Get network configuration of multi-channel decoder	invalid	NET_DVR_NETCFG_OTHER	244
NET_DVR_MATRIX_BIGSCREENCFG_GET	Get screen stitching parameter(supported by 64-T HD decoder)	valid	NET_DVR_BIGSCREENCFG	1140

Return to index

4.6.2 Set the parameters of the device: NET_DVR_SetDVRConfig

API: BOOL NET_DVR_SetDVRConfig(LONG lUserID, DWORD dwCommand,LONG lChannel, LPVOID lpInBuffer, DWORD dwInBufferSize)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwCommand	Parameter type. Please kindly refer to the following list
[in] lChannel	Channel number, if it is not the channel parameter, do not use lChannel, and set it as 0xFFFFFFFF
[in] lpInBuffer	Buffer that saves the output parameters
[in] dwInBufferSize	The buffer size (unit: byte)

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The structures and command numbers are different according to the various setting functions, and they are listed as below:

Macro Definition of dwCommand	Description	lChannel	lpInBuffer	Value
NET_DVR_SET_NETCFG_OTHER	Se network parameter channel decoder	invalid	NET_DVR_NETCFG_OTHER	245
NET_DVR_MATRIX_BIGSCREENCFG_SET	Set screen stitching parameter(supported by 64-T HD decoder)	valid	NET_DVR_BIGSCREENCFG	1141

Return to index

4.7 Function about Decoding Channel

4.7.1 Get the configuration of decoding channel:

NET_DVR_MatrixGetDecChanCfg

API: BOOL NET_DVR_MatrixGetDecChanCfg(LONG lUserID, DWORD dwDecChan, LPNET_DVR_MATRIX_DECCHAN_CONTROL lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChan	Decoding channel number
[out] lpInter	Decode channel zoom control, please kindly refer to the structure: NET_DVR_MATRIX_DECCHAN_CONTROL

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.7.2 Configure the decoding channel: NET_DVR_MatrixSetDecChanCfg

API: BOOL NET_DVR_MatrixSetDecChanCfg(LONG lUserID, DWORD dwDecChan, LPNET_DVR_MATRIX_DECCHAN_CONTROL lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChan	Decoding channel number
[in] lpInter	Decode channel zoom control, please kindly refer to the structure: NET_DVR_MATRIX_DECCHAN_CONTROL

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.7.3 Get the video format of the decoding channel:

NET_DVR_MatrixGetVideoStandard

API: BOOL NET_DVR_MatrixGetVideoStandard(LONG lUserID,LONG dwDecChanNum, LPDWORD lpdwVideoStandard)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[out] lpdwVideoStandard	Decode video format: 0- PAL, 1- NTSC

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.7.4 Set the video format of the decoding channel:

NET_DVR_MatrixSetVideoStandard

API: BOOL NET_DVR_MatrixSetVideoStandard(LONG lUserID,DWORD dwDecChanNum, DWORD dwVideoStandard)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[in] dwVideoStandard	Decode video format: 0- PAL, 1- NTSC

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.7.5 Get the status of current decoding channel:

NET_DVR_MatrixGetDecChanStatus

API: BOOL NET_DVR_MatrixGetDecChanStatus(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_MATRIX_DEC_CHAN_STATUS lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[out] lpInter	Status of the decoding channel, please kindly refer to the structure: NET_DVR_MATRIX_DEC_CHAN_STATUS

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: It is used to get the status of the decoding channel, including decoding status, stream transmission rate

Return to index

4.7.6 Get the switch of the decoding channel:

NET_DVR_MatrixGetDecChanEnable

API: BOOL NET_DVR_MatrixGetDecChanEnable(LONG lUserID, DWORD dwDecChanNum, LPDWORD lpdwEnable)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[out] lpdwEnable	0- close, 1- open

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The switch of decoding channel, is used to control the decoding process of the decoding channel. When set the switch to close, whether current decoding channel is during dynamic decoding or loop decoding, it will stop decoding. The display window will turn to black screen after it is effected. If set the switch on, it will resume the last process.

Notes: This function can be used with the switch of loop decoding to control loop decoding.

Return to index

4.7.7 Set the switch of the decoding channel:

NET_DVR_MatrixSetDecChanEnable

API: BOOL NET_DVR_MatrixSetDecChanEnable (LONG lUserID, DWORD

dwDecChanNum, DWORD dwEnable)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[in] dwEnable	0- close, 1- open

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The switch of decoding channel, is used to control the decoding process of the decoding channel. When set the switch to close, whether current decoding channel is during dynamic decoding or loop decoding, it will stop decoding. The display window will turn to black screen after it is effected. If set the switch on, it will resume the last process.

Notes: This function can be used with the switch of loop decoding to control loop decoding.

Return to index

4.8 Active Decoding

4.8.1 Start dynamic decoding: NET_DVR_MatrixStartDynamic_V30

API: BOOL NET_DVR_MatrixStartDynamic_V30(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_PU_STREAM_CFG lpDynamicInfo)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[in] lpDynamicInfo	Dynamic decoding parameter, please kindly refer to the structure: NET_DVR_PU_STREAM_CFG

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: It is used to connect one decoding channel of the multi-channel decoder to one channel of front-end device, and continue to decode until call the stop API or set decoding switch off. If decoding interruption is caused by network interrupted during the decoding process, the multi-channel decoder will automatically re-connect to the front device, till the connection is successful or the stop API is called. During re-connecting, the decoding channel is in the black state.

Return to index

4.8.2 Stop dynamic decoding: NET_DVR_MatrixStopDynamic

API: BOOL NET_DVR_MatrixStopDynamic(LONG lUserID, DWORD dwDecChanNum)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.8.3 Get the information of circle decoding channel:

NET_DVR_MatrixGetLoopDecChanInfo_V30

API: BOOL NET_DVR_MatrixGetLoopDecChanInfo_V30(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_MATRIX_LOOP_DECINFO_V30 lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in]dwDecChanNum	Decoding channel number
[out] lpInter	The info of circle decoding channel, please kindly refer to the structure: NET_DVR_MATRIX_LOOP_DECINFO_V30

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: It is used to set circle decoding parameter of one decode channel in multi-channel decoder, and every decoding channel can connect to 16 front-end channels for circle decoding. The cycle period supports to be set. After set successfully, if the starting flag of connection information is enabled, the channel go into circle status and start circle decoding; If disabled, then not start loop decoding. We can use with the switch API of circle decoding to achieve the circle decoding control, details refer to the part of the circle decoding switch (NET_DVR_MatrixGetLoopDecChanEnable and NET_DVR_MatrixSetLoopDecChanEnable).

Return to index

4.8.4 Set the circle decoding channel:

NET_DVR_MatrixSetLoopDecChanInfo_V30

API: BOOL NET_DVR_MatrixSetLoopDecChanInfo_V30(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_MATRIX_LOOP_DECINFO_V30 lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[in] lpInter	The configuration of the circle decoding channel, please kindly refer to the structure: NET_DVR_MATRIX_LOOP_DECINFO_V30

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: It is used to set circle decoding parameter of one decode channel in

multi-channel decoder, and every decoding channel can connect to 16 front-end channels for circle decoding. The cycle period supports to be set. After set successfully, if the starting flag of connection information is enabled, the channel go into circle status and start circle decoding; If disabled, then not start loop decoding. We can use with the switch API of circle decoding to achieve the circle decoding control, details refer to the part of the circle decoding switch (NET_DVR_MatrixGetLoopDecChanEnable and NET_DVR_MatrixSetLoopDecChanEnable).

Return to index

4.8.5 Get the circle switch of the decoding channel:

NET_DVR_MatrixGetLoopDecChanEnable

API: BOOL NET_DVR_MatrixGetLoopDecChanEnable(LONG lUserID, DWORD dwDecChanNum, LPDWORD lpdwEnable)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[out] lpdwEnable	0- close, 1- open

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The circle switch is used to control the starting and stopping of circle decoing, not to control the starting and stopping of decoding. When set the circle switch off, the decoding channel will stop the circle decoding and continue to decode the stream of the currently connected channel, that is, turn to dynamic decoding. When set the switch on, it will resume to circle decoding.

Return to index

4.8.6 Set the circle switch of the decoding channel:

NET_DVR_MatrixSetLoopDecChanEnable

API: BOOL NET_DVR_MatrixSetLoopDecChanEnable(LONG lUserID, DWORD dwDecChanNum, DWORD dwEnable)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[in] dwEnable	0- close, 1- open

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The circle switch is used to control the starting and stopping of circle decoing, not to control the starting and stopping of decoding. When set the circle switch off, the decoding channel will stop the circle decoding and continue to decode the stream of the currently connected channel, that is, turn to

dynamic decoding. When set the switch on, it will resume to circle decoding.

Return to index

4.8.7 Set the circle switch of all decoding channels:

NET_DVR_MatrixGetLoopDecEnable

API: BOOL NET_DVR_MatrixGetLoopDecEnable(LONG lUserID, LPDWORD lpdwEnable)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[out] lpdwEnable	indicated by bit: 0- close, 1- open

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: lpdwEnable is indicated by bit, for example, if lpdwEnable&0x1=1 and lpdwEnable&0x4=1, other bits are 0, it means the circle switch of no.1 and no.3 decoding channel are turned on, and that of the other channels are off.

Return to index

4.8.8 Get the information of current decoding channel:

NET_DVR_MatrixGetDecChanInfo_V30

API: BOOL NET_DVR_MatrixGetDecChanInfo_V30(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_MATRIX_DEC_CHAN_INFO_V30 lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[out] lpInter	Decoding channel information, please kindly refer to the structure: NET_DVR_MATRIX_DEC_CHAN_INFO_V30

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: It is used to get the information of current decoding channel, including the information of front-end device, stream mode, and so on.

Return to index

4.8.9 Configure the playback of remote files:

NET_DVR_MatrixSetRemotePlay

API: BOOL NET_DVR_MatrixSetRemotePlay(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_MATRIX_DEC_REMOTE_PLAY lpInter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30

	[in] dwDecChanNum	Decoding channel number
	[in] lpInter	Playback parameters, please kindly refer to the structure: NET_DVR_MATRIX_DEC_REMOTE_PLAY

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: After calling this API to configure the parameters, please call NET_DVR_MatrixSetRemotePlayControl (NET_DVR_PLAYSTART) to start play.

Return to index

4.8.10 Control the playback of remote files:

NET_DVR_MatrixSetRemotePlayControl

API: BOOL NET_DVR_MatrixSetRemotePlayControl(LONG lUserID, DWORD dwDecChanNum, DWORD dwControlCode, DWORD dwInValue, DWORD *lpOutValue)

Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDecChanNum	Decoding channel number
	[in] dwControlCode	Control commands, see to the following list
	[in] dwInValue	The input value, related to the command
	[in] lpOutValue	The output parameter, related to the command

dwControlCode Macro Definition	Value	Implication
NET_DVR_PLAYSTART	**1**	Start playing
NET_DVR_PLAYSTOP	**2**	Stop playing
NET_DVR_PLAYPAUSE	**3**	Pause
NET_DVR_PLAYRESTART	**4**	Resume
NET_DVR_PLAYFAST	**5**	Fast
NET_DVR_PLAYSLOW	**6**	Slow
NET_DVR_PLAYNORMAL	**7**	Normal speed
NET_DVR_PLAYSTARTAUDIO	**9**	Open sound
NET_DVR_PLAYSTOPAUDIO	**10**	Close sound
NET_DVR_PLAYSETPOS	**12**	Change progress of playback by file

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: dwInValue and lpOutValue are related to the control command. For some commands, such as NET_DVR_PLAYSTART, it does not require to set both the two parameters; For some commands, such as NET_DVR_PLAYSETPOS, it requires to set the value of dwInValue. The playback by time does not support the control command **NET_DVR_PLAYSETPOS**

Return to index

4.8.11 Get the status of playback:

NET_DVR_MatrixGetRemotePlayStatus

API: BOOL NET_DVR_MatrixGetRemotePlayStatus(LONG lUserID, DWORD dwDecChanNum,LPNET_DVR_MATRIX_DEC_REMOTE_PLAY_STATUS lpOuter)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[out] lpOuter	Playback status, please kindly refer to the structure: NET_DVR_MATRIX_DEC_REMOTE_PLAY_STATUS

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: The decoder connects to the front-end device and playback the files by file name or by time. This API is used to get the status of the playback. There is certain delay for the command is transferred by the client, so we cannot call NET_DVR_MatrixSetRemotePlayControl frequently. If we get the status of playback and handle the playback according to the status, we should consider the network delay.

Return to index

4.9 Passive Decoding

4.9.1 Start passive decoding: NET_DVR_MatrixStartPassiveDecode

API: LONG NET_DVR_MatrixStartPassiveDecode(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_MATRIX_PASSIVEMODE lpPassiveMode)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] dwDecChanNum	Decoding channel number
[in] lpPassiveMode	Passive decoding parameter, please kindly refer to the structure: NET_DVR_MATRIX_PASSIVEMODE

Return: -1 means false, and other values could be used as the parameters of other interfaces, such as NET_DVR_MatrixSendData. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.9.2 Send data to the passive decoding channel:

NET_DVR_MatrixSendData

API: BOOL NET_DVR_MatrixSendData(LONG lPassiveHandle, char *pSendBuf, DWORD dwBufSize)

Parameters:

[in]	lPassiveHandle	The return value of NET_DVR_MatrixStartPassiveDecode
[in]	pSendBuf	The buffer that saves the data to be sent
[in]	dwBufSize	Size of the buffer, should be less than 30K bytes

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.9.3 Stop the passive decoding: NET_DVR_MatrixStopPassiveDecode

API: BOOL NET_DVR_MatrixStopPassiveDecode(LONG lPassiveHandle)

Parameters:

[in]	lPassiveHandle	The return value of NET_DVR_MatrixStartPassiveDecode

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.9.4 Get status of the passive decoding:

NET_DVR_MatrixGetPassiveDecodeStatus

API: LONG NET_DVR_MatrixGetPassiveDecodeStatus(LONG lPassiveHandle)

Parameters:

[in]	lPassiveHandle	The return value of NET_DVR_MatrixStartPassiveDecode

Return: -1- failed, 1- send the data successfully, 2- sending is suspended, 3- sending is resumed, 4- error, 5- heartbeat messages. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.9.5 Control of the passive decoding:

NET_DVR_MatrixPassiveDecodeControl

API:	BOOL NET_DVR_MatrixPassiveDecodeControl(LONG lUserID, DWORD dwDecChanNum, LPNET_DVR_PASSIVEDECODE_CONTROL lpInter)	
Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDecChanNum	Decoding channel number
	[in] lpInter	The control parameters, please kindly refer to the structure: NET_DVR_PASSIVEDECODE_CONTROL
Return:	Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.	
Remarks:		

Return to index

4.10 Upload the LOGO and Control Its Display

4.10.1 Upload the LOGO: NET_DVR_UploadLogo

API:	BOOL NET_DVR_UploadLogo(LONG lUserID,DWORD dwDecChanNum, LPNET_DVR_DISP_LOGOCFG lpDispLogoCfg, char *sLogoBuffer)	
Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDecChanNum	The number of decoding channel
	[in] lpDispLogoCfg	The LOGO parameters, please kindly refer to the structure: NET_DVR_DISP_LOGOCFG
	[in] sLogoBuffer	LOGO data buffer, at most 100k, width and height must be multiples of 32
Return:	Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.	
Remarks:		

Return to index

4.10.2 Display control of the LOGO: NET_DVR_LogoSwitch

API:	BOOL NET_DVR_LogoSwitch(LONG lUserID, DWORD dwDecChan, DWORD dwLogoSwitch)	
Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[in] dwDecChan	The number of decoding channel
	[in] dwLogoSwitch	Switch command, please kindly see to the following list

Macro Definition	Value	Implication
NET_DVR_SHOWLOGO	1	Display LOGO
NET_DVR_HIDELOGO	2	Hide LOGO

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks:

Return to index

4.11 Transparent Channel

4.11.1 Get the information of transparent channel:

NET_DVR_MatrixGetTranInfo_V30

API: BOOL NET_DVR_MatrixGetTranInfo_V30(LONG lUserID, LPNET_DVR_MATRIX_TRAN_CHAN_CONFIG_V30 lpTranInfo)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[out] lpTranInfo	The parameter of transparent channel, please kindly refer to the structure: NET_DVR_MATRIX_TRAN_CHAN_CONFIG_V30

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: Here transparent channel configuration is to build network transparent channel between the decoder and the front-end device, not between client and the decoder. Because most multi-channel decoders don't support building 232/485 transparent channel between with the PC client, local serial port can only be the access port of the serial console (via RS 232) or device like control keyboard (via RS232/RS485).

Return to index

4.11.2 Set the transparent channel: NET_DVR_MatrixSetTranInfo_V30

API: BOOL NET_DVR_MatrixSetTranInfo_V30(LONG lUserID, LPNET_DVR_MATRIX_TRAN_CHAN_CONFIG_V30 lpTranInfo)

Parameters:

[in] lUserID	User ID, the return value of NET_DVR_Login_V30
[in] lpTranInfo	The parameter of transparent channel, please kindly refer to the structure: NET_DVR_MATRIX_TRAN_CHAN_CONFIG_V30

Return: Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.

Remarks: Currently, one multi-channel decoder supports 64 transparent channels at

most, including 232 and 485 transparent channels. It supports only one 232 full-duplex transparent channel and one 485 full-duplex transparent channel, and it supports not to set full-duple transparent channel.

Return to index

4.12 Device Status

4.12.1 Get the status of the device:

NET_DVR_MatrixGetDeviceStatus_V41

API:	BOOL NET_DVR_MatrixGetDeviceStatus_V41(LONG lUserID, LPNET_DVR_DECODER_WORK_STATUS_V41 lpDecoderCfg)	
Parameters:	[in] lUserID	User ID, the return value of NET_DVR_Login_V30
	[out] lpDecoderCfg	The status of the decoder, please kindly refer to the structure: NET_DVR_DECODER_WORK_STATUS_V41
Return:	Return TRUE on success, FALSE on failure. Please call NET_DVR_GetLastError to get the error code.	
Remarks:		

Return to index

5 Macro Definition of Error Code

5.1 Error code of network communication library

Error	Value	Message
NET_DVR_NOERROR	0	No error.
NET_DVR_PASSWORD_ERROR	1	User name or password error.
NET_DVR_NOENOUGHPRI	2	Not authorized to do this operation.
NET_DVR_NOINIT	3	SDK is not initialized.
NET_DVR_CHANNEL_ERROR	4	Channel number error. There is no corresponding channel number on the device.
NET_DVR_OVER_MAXLINK	5	The number of clients connected to the device has exceeded the max limit.
NET_DVR_VERSIONNOMATCH	6	Version mismatch. SDK version is not matching with the device.
NET_DVR_NETWORK_FAIL_CONNECT	7	Failed to connect to the device. The device is off-line, or connection timeout caused by network.
NET_DVR_NETWORK_SEND_ERROR	8	Failed to send data to the device.
NET_DVR_NETWORK_RECV_ERROR	9	Failed to receive data from the device.
NET_DVR_NETWORK_RECV_TIMEOUT	10	Timeout when receiving the data from the device.
NET_DVR_NETWORK_ERRORDATA	11	The data sent to the device is illegal, or the data received from the device error. E.g. The input data is not supported by the device for remote configuration.
NET_DVR_ORDER_ERROR	12	API calling order error.
NET_DVR_OPERNOPERMIT	13	Not authorized for this operation.
NET_DVR_COMMANDTIMEOUT	14	Executing command on the device is timeout.
NET_DVR_ERRORSERIALPORT	15	Serial port number error. The assigned serial port does not exist on the device.
NET_DVR_ERRORALARMPORT	16	Alarm port number error.
NET_DVR_PARAMETER_ERROR	17	Parameter error. Input or output parameter in the SDK API is NULL.
NET_DVR_CHAN_EXCEPTION	18	Device channel is in exception status.
NET_DVR_NODISK	19	No hard disk on the device, and the operation of recording and hard disk configuration will fail.
NET_DVR_ERRORDISKNUM	20	Hard disk number error. The assigned hard disk number does not exist during hard disk management.
NET_DVR_DISK_FULL	21	Device hark disk is full.
NET_DVR_DISK_ERROR	22	Device hard disk error.
NET_DVR_NOSUPPORT	23	Device does not support this function.
NET_DVR_BUSY	24	Device is busy.
NET_DVR_MODIFY_FAIL	25	Failed to modify device parameters.

NET_DVR_PASSWORD_FORMAT_ERROR	26	The inputting password format is not correct.
NET_DVR_DISK_FORMATING	27	Hard disk is formatting, and the operation cannot be done.
NET_DVR_DVRNORESOURCE	28	Not enough resource on the device.
NET_DVR_DVROPRATEFAILED	29	Device operation failed.
NET_DVR_OPENHOSTSOUND_FAIL	30	Failed to collect local audio data or to open audio output during voice talk / broadcasting.
NET_DVR_DVRVOICEOPENED	31	Voice talk channel on the device has been occupied.
NET_DVR_TIMEINPUTERROR	32	Time input is not correct.
NET_DVR_NOSPECFILE	33	There is no selected file for playback.
NET_DVR_CREATEFILE_ERROR	34	Failed to create a file, during local recording, saving picture, getting configuration file or downloading record file.
NET_DVR_FILEOPENFAIL	35	Failed to open a file, when importing configuration file, upgrading device or uploading inquest file.
NET_DVR_OPERNOTFINISH	36	The last operation has not been completed.
NET_DVR_GETPLAYTIMEFAIL	37	Failed to get the current played time.
NET_DVR_PLAYFAIL	38	Failed to start playback.
NET_DVR_FILEFORMAT_ERROR	39	The file format is not correct.
NET_DVR_DIR_ERROR	40	File directory error.
NET_DVR_ALLOC_RESOURCE_ERROR	41	Resource allocation error.
NET_DVR_AUDIO_MODE_ERROR	42	Sound adapter mode error. Currently opened sound playing mode does not match with the set mode.
NET_DVR_NOENOUGH_BUF	43	Buffer is not enough.
NET_DVR_CREATESOCKET_ERROR	44	Create SOCKET error.
NET_DVR_SETSOCKET_ERROR	45	Set SOCKET error.
NET_DVR_MAX_NUM	46	The number of login or preview connections has exceeded the SDK limitation.
NET_DVR_USERNOTEXIST	47	User doest not exist. The user ID has been logged out or unavailable.
NET_DVR_WRITEFLASHERROR	48	Writing FLASH error. Failed to write FLASH during device upgrade.
NET_DVR_UPGRADEFAIL	49	Failed to upgrade device. It is caused by network problem or the language mismatch between the device and the upgrade file.
NET_DVR_CARDHAVEINIT	50	The decode card has already been initialed.
NET_DVR_PLAYERFAILED	51	Failed to call API of player SDK.
NET_DVR_MAX_USERNUM	52	The number of login user has reached the maximum limit.
NET_DVR_GETLOCALIPANDMACFAIL	53	Failed to get the IP address or physical address of local PC.
NET_DVR_NOENCODEING	54	This channel hasn't started encoding.
NET_DVR_IPMISMATCH	55	IP address not match.
NET_DVR_MACMISMATCH	56	MAC address not match.
NET_DVR_UPGRADELANGMISMATCH	57	The language of upgrading file does not match the language of the device.

NET_DVR_MAX_PLAYERPORT	58	The number of player ports has reached the maximum limit.
NET_DVR_NOSPACEBACKUP	59	No enough space to backup file in backup device.
NET_DVR_NODEVICEBACKUP	60	No backup device.
NET_DVR_PICTURE_BITS_ERROR	61	The color quality setting of the picture does not match the requirement, and it should be limited to 24.
NET_DVR_PICTURE_DIMENSION_ERROR	62	The dimension is over 128x256.
NET_DVR_PICTURE_SIZ_ERROR	63	The size of picture is over 100K.
NET_DVR_LOADPLAYERSDKFAILED	64	Failed to load the player SDK.
NET_DVR_LOADPLAYERSDKPROC_ERROR	65	Can not find the function in player SDK.
NET_DVR_LOADDSSDKFAILED	66	Failed to load the library file-"DsSdk".
NET_DVR_LOADDSSDKPROC_ERROR	67	Can not find the API in "DsSdk".
NET_DVR_DSSDK_ERROR	68	Failed to call the API in "DsSdk".
NET_DVR_VOICEMONOPOLIZE	69	Sound adapter has been monopolized.
NET_DVR_JOINMULTICASTFAILED	70	Failed to join to multicast group.
NET_DVR_CREATEDIR_ERROR	71	Failed to create log file directory.
NET_DVR_BINDSOCKET_ERROR	72	Failed to bind socket.
NET_DVR_SOCKETCLOSE_ERROR	73	Socket disconnected. It is caused by network disconnection or destination unreachable.
NET_DVR_USERID_ISUSING	74	The user ID is operating when logout.
NET_DVR_SOCKETLISTEN_ERROR	75	Failed to listen.
NET_DVR_PROGRAM_EXCEPTION	76	SDK program exception.
NET_DVR_WRITEFILE_FAILED	77	Failed to write file, during local recording, saving picture or downloading record file.
NET_DVR_FORMAT_READONLY	78	Failed to format read-only HD.
NET_DVR_WITHSAMEUSERNAME	79	This user name already exists in the user configuration structure.
NET_DVR_DEVICETYPE_ERROR	80	Device type does not match when import configuration.
NET_DVR_LANGUAGE_ERROR	81	Language does not match when import configuration.
NET_DVR_PARAVERSION_ERROR	82	Software version does not match when import configuration.
NET_DVR_IPCHAN_NOTALIVE	83	IP channel is not on-line when previewing.
NET_DVR_RTSP_SDK_ERROR	84	Load StreamTransClient.dll failed.
NET_DVR_CONVERT_SDK_ERROR	85	Load SystemTransform.dll failed.
NET_DVR_IPC_COUNT_OVERFLOW	86	Exceeds maximum number of connected IP channels.
NET_DVR_MAX_ADD_NUM	87	Exceeds maximum number of supported record labels or other operations.
NET_DVR_PARAMMODE_ERROR	88	Image intensifier, parameter mode error. This error may occur when client sets software or hardware parameters.
NET_DVR_CODESPITTER_OFFLINE	89	Code splitter is offline.
NET_DVR_BACKUP_COPYING	90	Device is backing up.
NET_DVR_CHAN_NOTSUPPORT	91	Channel not support.
NET_DVR_CALLINEINVALID	92	The height line location is too concentrated, or the length line is not inclined enough.

NET_DVR_CALCANCELCONFLICT	93	Cancel calibration conflict, if the rule and overall actual size filter have been set.
NET_DVR_CALPOINTOUTRANGE	94	Calibration point exceeds the range.
NET_DVR_FILTERRECTINVALID	95	The size filter does not meet the requirement.
NET_DVR_DDNS_DEVOFFLINE	96	Device has not registered to DDNS.
NET_DVR_DDNS_INTER_ERROR	97	DDNS inner error.
NET_DVR_ALIAS_DUPLICATE	150	Alias is duplicate (for EasyDDNS)
NET_DVR_DEV_NET_OVERFLOW	800	Network traffic is over device ability limit.
NET_DVR_STATUS_RECORDFILE_WRITING_NOT_LOCK	801	The video file is recording and can't be locked.
NET_DVR_STATUS_CANT_FORMAT_LITTLE_DISK	802	The hard disk capacity is too small and can not be formatted.
Error code of RAID		
NET_DVR_NAME_NOT_ONLY	200	This user name already exists.
NET_DVR_OVER_MAX_ARRAY	201	The array exceeds the limitation.
NET_DVR_OVER_MAX_VD	202	The virtual disk exceeds the limitation.
NET_DVR_VD_SLOT_EXCEED	203	The virtual disk slots are full.
NET_DVR_PD_STATUS_INVALID	204	Physical disk used to rebuild RAID is in error state.
NET_DVR_PD_BE_DEDICATE_SPARE	205	Physical disk used to rebuild RAID is assigned as spare disk.
NET_DVR_PD_NOT_FREE	206	Physical disk used to rebuild RAID is not free.
NET_DVR_CANNOT_MIG2NEWMODE	207	Can not migrate from current RAID type to the new type.
NET_DVR_MIG_PAUSE	208	Migration has been paused.
NET_DVR_MIG_ABOUTED	209	Migration has been aborted.
NET_DVR_EXIST_VD	210	There is virtual disk in the array, and the array can not been deleted.
NET_DVR_TARGET_IN_LD_FUNCTIONAL	211	Target physical disk is part of the virtual disk and is functional.
NET_DVR_HD_IS_ASSIGNED_ALREADY	212	Specified physical disk is assigned as a virtual disk.
NET_DVR_INVALID_HD_COUNT	213	Number of physical disks doesn't fit the specified RAID level.
NET_DVR_LD_IS_FUNCTIONAL	214	Specified virtual disk is functional and it can not be rebuilt.
NET_DVR_BGA_RUNNING	215	BGA is running.
NET_DVR_LD_NO_ATAPI	216	Can not create virtual disk with ATAPI drive.
NET_DVR_MIGRATION_NOT_NEED	217	Migration is not necessary.
NET_DVR_HD_TYPE_MISMATCH	218	Physical disks are not of the same type.
NET_DVR_NO_LD_IN_DG	219	No virtual disk exists on the specified array.
NET_DVR_NO_ROOM_FOR_SPARE	220	Disk space is too small to be assigned as spare drive.
NET_DVR_SPARE_IS_IN_MULTI_DG	221	Disk is already assigned as a spare drive for an array.
NET_DVR_DG_HAS_MISSING_PD	222	Disk is missing from an array.

Error code of intelligent device		
NET_DVR_ID_ERROR	300	Configuration ID is illegal.
NET_DVR_POLYGON_ERROR	301	Polygon does not match requirement.
NET_DVR_RULE_PARAM_ERROR	302	Rule parameter is illegal.
NET_DVR_RULE_CFG_CONFLICT	303	Configuration conflict.
NET_DVR_CALIBRATE_NOT_READY	304	Calibration not ready.
NET_DVR_CAMERA_DATA_ERROR	305	Camera parameter is illegal.
NET_DVR_CALIBRATE_DATA_UNFIT	306	Not inclined enough, not fit to calibrate.
NET_DVR_CALIBRATE_DATA_CONFILICT	307	Calibration error.
NET_DVR_CALIBRATE_CALC_FAIL	308	Failed to calculate camera calibration parameter.
NET_DVR_CALIBRATE_LINE_OUT_RECT	309	The input calibrating line exceeds the external rectangle sample.
NET_DVR_ENTER_RULE_NOT_READY	310	Enter rule not ready.
NET_DVR_AID_RULE_NO_INCLUDE_LANE	311	It does not include lane in the traffic event rule (especial for traffic jam or driving against the traffic).
NET_DVR_LANE_NOT_READY	312	Lane not ready.
NET_DVR_RULE_INCLUDE_TWO_WAY	313	There are two different directions in event rule.
NET_DVR_LANE_TPS_RULE_CONFLICT	314	The lane conflicts with the data rule.
NET_DVR_NOT_SUPPORT_EVENT_TYPE	315	The event type is not supported by the device.
NET_DVR_LANE_NO_WAY	316	The lane has no direction.
NET_DVR_SIZE_FILTER_ERROR	317	The size of filter is illegal.
NET_DVR_LIB_FFL_NO_FACE	318	There is no face when feature point positioning.
NET_DVR_LIB_FFL_IMG_TOO_SMALL	319	The input image is too small when feature point positioning.
NET_DVR_LIB_FD_IMG_NO_FACE	320	The input image has no face when detecting face in single image.
NET_DVR_LIB_FACE_TOO_SMALL	321	Face is too small when building model.
NET_DVR_LIB_FACE_QUALITY_TOO_BAD	322	Face image is of poor quality when building model.
NET_DVR_KEY_PARAM_ERR	323	Advanced parameter setting error.
NET_DVR_CALIBRATE_DATA_ERR	324	Calibration sample size error, or data value error, or sample points beyond the horizon
NET_DVR_CALIBRATE_DISABLE_FAIL	325	The configured rules do not allow to cancel calibration.

5.2 Error code of RTSP communication library

Error	Value	Message
NET_DVR_RTSP_GETPORTFAILED	407	RTSP port getting error.
NET_DVR_RTSP_DESCRIBESENDTIMEOUT	411	Sending "RTSP DECRIBE" is timeout.
NET_DVR_RTSP_DESCRIBESENDERROR	412	Failed to send "RTSP DECRIBE".
NET_DVR_RTSP_DESCRIBERECVTIMEOUT	413	Receiving "RTSP DECRIBE" is timeout.
NET_DVR_RTSP_DESCRIBERECVDATALOST	414	Receiving data of "RTSP DECRIBE" error.
NET_DVR_RTSP_DESCRIBERECVERROR	415	Failed to receive "RTSP DECRIBE".

NET_DVR_RTSP_DESCRIBESERVERERR	416	"RTSP DECRIBE" device returns the error that values 401 or 501.
NET_DVR_RTSP_SETUPSENDTIMEOUT	421	Sending "RTSP SETUP" is timeout.
NET_DVR_RTSP_SETUPSENDERROR	422	Sending "RTSP SETUP" error.
NET_DVR_RTSP_SETUPRECVTIMEOUT	423	Receiving "RTSP SETUP" is timeout.
NET_DVR_RTSP_SETUPRECVDATALOST	424	Receiving data of "RTSP SETUP" error.
NET_DVR_RTSP_SETUPRECVERROR	425	Failed to receive "RTSP SETUP".
NET_DVR_RTSP_OVER_MAX_CHAN	426	"RTSP SETUP" device returns the error that values 401 or 501. It exceeds the max connection number.
NET_DVR_RTSP_PLAYSENDTIMEOUT	431	Sending "RTSP PLAY" is timeout.
NET_DVR_RTSP_PLAYSENDERROR	432	Sending "RTSP PLAY" error.
NET_DVR_RTSP_PLAYRECVTIMEOUT	433	Receiving "RTSP PLAY" is timeout.
NET_DVR_RTSP_PLAYRECVDATALOST	434	Receiving data of "RTSP PLAY" error.
NET_DVR_RTSP_PLAYRECVERROR	435	Failed to receive "RTSP PLAY".
NET_DVR_RTSP_PLAYSERVERERR	436	"RTSP PLAY" device returns the error that values 401 or 501.
NET_DVR_RTSP_TEARDOWNSENDTIMEOUT	441	Sending "RTSP TEARDOWN" is timeout.
NET_DVR_RTSP_TEARDOWNSENDERROR	442	Sending "RTSP TEARDOWN" error.
NET_DVR_RTSP_TEARDOWNRECVTIMEOUT	443	Receiving "RTSP TEARDOWN" is timeout.
NET_DVR_RTSP_TEARDOWNRECVDATALOST	444	Receiving data of "RTSP TEARDOWN" error.
NET_DVR_RTSP_TEARDOWNRECVERROR	445	Failed to receive "RTSP TEARDOWN".
NET_DVR_RTSP_TEARDOWNSERVERERR	446	"RTSP TEARDOWN" device returns the error that values 401 or 501.

5.3 Error code of software decoding library

Error	Value	Message
NET_PLAYM4_NOERROR	500	No error.
NET_PLAYM4_PARA_OVER	501	Input parameter is invalid.
NET_PLAYM4_ORDER_ERROR	502	API calling order error.
NET_PLAYM4_TIMER_ERROR	503	Failed to create multimedia clock.
NET_PLAYM4_DEC_VIDEO_ERROR	504	Failed to decode video data.
NET_PLAYM4_DEC_AUDIO_ERROR	505	Failed to decode audio data.
NET_PLAYM4_ALLOC_MEMORY_ERROR	506	Failed to allocate memory.
NET_PLAYM4_OPEN_FILE_ERROR	507	Failed to open the file.
NET_PLAYM4_CREATE_OBJ_ERROR	508	Failed to create thread event.
NET_PLAYM4_CREATE_DDRAW_ERROR	509	Failed to create DirectDraw object.
NET_PLAYM4_CREATE_OFFSCREEN_ERROR	510	Failed to create backstage cache for OFFSCREEN

		mode.
NET_PLAYM4_BUF_OVER	511	Buffer overflow, failed to input stream.
NET_PLAYM4_CREATE_SOUND_ERROR	512	Failed to create audio equipment.
NET_PLAYM4_SET_VOLUME_ERROR	513	Failed to set the volume.
NET_PLAYM4_SUPPORT_FILE_ONLY	514	This API can be called only for file playback mode.
NET_PLAYM4_SUPPORT_STREAM_ONLY	515	This API can be called only when playing stream.
NET_PLAYM4_SYS_NOT_SUPPORT	516	Not support by the system. Decoder can only work on the system above Pentium 3.
NET_PLAYM4_FILEHEADER_UNKNOWN	517	There is no file header.
NET_PLAYM4_VERSION_INCORRECT	518	The version mismatch between decoder and encoder.
NET_PLAYM4_INIT_DECODER_ERROR	519	Failed to initialize the decoder.
NET_PLAYM4_CHECK_FILE_ERROR	520	The file is too short, or the stream data is unknown.
NET_PLAYM4_INIT_TIMER_ERROR	521	Failed to initialize multimedia clock.
NET_PLAYM4_BLT_ERROR	522	BLT failure.
NET_PLAYM4_UPDATE_ERROR	523	Failed to update overlay surface
NET_PLAYM4_OPEN_FILE_ERROR_MULTI	524	Failed to open video & audio stream file.
NET_PLAYM4_OPEN_FILE_ERROR_VIDEO	525	Failed to open video stream file.
NET_PLAYM4_JPEG_COMPRESS_ERROR	526	JPEG compression error.
NET_PLAYM4_EXTRACT_NOT_SUPPORT	527	Don't support the version of this file.
NET_PLAYM4_EXTRACT_DATA_ERROR	528	Extract video data failed.