

NetSDK Programming Manual

Version 1.0.0.1 (Built in 20180223)

2018-02-23

All rights reserved

Foreword

Thank you for using our devices. We are going to provide best services for you. This manual may contain spelling grammar and punctuation errors. We will update this manual regularly.

Catalogue

Catalogue	1
1. Brief introduction.....	5
1.1 Summary.....	5
1.2 Applicability.....	5
1.3 The instruction of programming	6
1.4 Typical calling sequence	7
1.4.1 Basic flow chart of calling SDK	7
1.4.2 Related interface function of every module.....	8
1.4.3 Flow chart of real-time preview module.....	10
1.4.4 Playback and download module	12
1.4.5 Parameter configuration module.....	13
1.4.6 Device maintenance module	13
1.4.7 Voice intercom module	14
1.4.8 Alarming module	15
1.4.8.1 Protective alarm mode	15
1.4.8.2 Monitoring alarm mode	16
1.4.9 Transparent serial channel module.....	18
2 Definition of data structure.....	19
2.1 System time structure SDK_SYSTEM_TIME.....	19
2.2 Related structures of video files.....	19
2.2.1 Query condition structures H264_DVR_FINDINFO	19
2.2.2 Returned video information structures H264_DVR_FILE_DATA	19
2.2.3 Query the structures by the time period SDK_SearchByTime	20
2.2.4 Related structures of the query results SDK_SearchByTimeResult	20
2.3 Structure of configuration information SDK_CONFIG_TYPE	21
2.4 Related structures of hard disk storage control	28
2.4.1 Storage device control types SDK_StorageDeviceControlTypes	28
2.4.2 Storage device control SDK_StorageDeviceControl	29
2.5 Related structures of frame information	29
2.6 Local playaction control SDK_LoalPlayAction	31
2.7 Subconnection type SubConnType.....	31
2.8 Socket style SocketStyle.....	31
2.9 Related structures of cloud upgrade SDK_CloudUpgradeVersion.....	32
2.10 Related information of serial port.....	32
2.10.1 Serial port type SERIAL_TYPE.....	32
2.10.2 Related information of serial port TransComChannel	32
2.11 Related information of playback action.....	32
2.12 Related structures of callback data in active service.....	33
2.12.1 Device information H264_DVR_DEVICEINFO	33
2.12.2 Active service of callback data H264_DVR_ACTIVEREG_INFO	34
2.12.3 Related structures of active registration configuration SDK_DASSerInfo	34
2.13 Related structures of alarming center.....	35

2.13.1 Related configuration of alarming center SDK_NetAlarmServerConfigAll...	35
2.13.2 Alarming center message content SDK_NetAlarmCenterMsg.....	36
2.14 Related structures of work state SDK_DVR_WORKSTATE	37
2.15 Related structures of network alarm SDK_NetAlarmInfo	38
3 Interface definition.....	38
3.1 SDK initialization.....	38
3.1.1 Initialization SDK H264_DVR_Init	38
3.1.2 Release SDK resources H264_DVR_Cleanup	39
3.2 SDK local function	39
3.2.1 Set wait time and try times H264_DVR_SetConnectTime	39
3.2.2 Bind local IP H264_DVR_SetLocalBindAddress	40
3.2.3 Return to error code of the final operation H264_DVR_GetLastError	40
3.3 User registration.....	40
3.3.1 User login device H264_DVR_Login	40
3.3.2 User logout device H264_DVR_Logout	41
3.3.3 Active registration H264_DVR_StartActiveRegister.....	41
3.4 Real-time monitoring.....	42
3.4.1 Real-time preview H264_DVR_RealPlay	42
3.4.2 Stop preview H264_DVR_StopRealPlay	43
3.4.3 Set data callback H264_DVR_SetRealDataCallBack	44
3.4.4 Cleanup callback function H264_DVR_DelRealDataCallBack.....	45
3.5 Forced I frame H264_DVR_MakeKeyFrame	46
3.6 Playback and download.....	46
3.6.1 Find video by file name H264_DVR_FindFile	46
3.6.2 Search video files by time H264_DVR_FindFileByTime	47
3.6.3 Playback video by name	47
3.6.4 Playback video by time H264_DVR_PlayBackByTime	49
3.6.5 Stop playback video H264_DVR_StopPlayBack	50
3.6.6 Playback Control H264_DVR_PlayBackControl.....	51
3.6.7 Downloading Video Files by File Name H264_DVR_GetFileByName	51
3.6.8 Downloading Video Files by Time H264_DVR_GetFileByTime	53
3.6.9 Stop Downloading Video Files H264_DVR_StopGetFile	54
3.6.10 Download Control H264_DVR_GetFileControl	55
3.6.11 Getting Download Progress H264_DVR_GetDownloadPos.....	55
3.7 PTZ Control H264_DVR_PTZControl.....	56
3.8 Parameter Configuration	56
3.8.1 Get Device Configuration H264_DVR_GetDevConfig	56
3.8.2 Setting Device Configuration H264_DVR_SetDevConfig	58
3.8.3 Setting Device Configurations Across Network Segments H264_DVR_Set ConfigOverNet.....	58
3.9 Log Management H264_DVR_FindDVRLog	59
3.10 Equipment Control H264_DVR_ControlDVR	60
3.11 Upgrading Device Programs	60
3.11.1 Local Upgrade H264_DVR_Upgrade.....	60
3.11.2 Obtaining Upgrade Status H264_DVR_GetUpgradeState	62

3.11.3 Close Upgrade Handle	H264_DVR_CloseUpgradeHandle	62
3.11.4 Close Upgrade	H264_DVR_Upgrade_Cloud	63
3.11.5 Stopping Cloud Upgrade	H264_DVR_StopUpgrade_Cloud	64
3.12 Voice Intercom.....		64
3.12.1 Start Intercom	H264_DVR_StartVoiceCom_MR.....	65
3.12.2 Send Talkback Data	H264_DVR_VoiceComSendData.....	65
3.12.3 Stop Intercom	H264_DVR_StopVoiceCom	66
3.12.4 Setting Intercom Audio Coding Mode	H264_DVR_SetTalkMode	66
3.13 Recording Mode Settings		66
3.13.1 Manual Recording	H264_DVR_StartDVRRecord.....	67
3.13.2 Close Recording	H264_DVR_StopDVRRecord	67
3.14 Setting System Time H264_DVR_SetSystemDateTime		68
3.15 Armed Alarm.....		68
3.15.1 Alarm Status Acquisition	H264_DVR_SetDVRMessCallBack	68
3.15.2 Setting Alarm Callback Upload Channe	H264_DVR_SetupAlarmChan	69
3.15.3 Turning Off the Alarm Callback Upload Path	H264_DVR_CloseAlarm Chan.....	70
3.16 Monitor alarm		70
3.16.1 Starting Alarm Center Monitoring	H264_DVR_StartAlarmCenterListen	70
3.16.2 Turning off Alarm Center Monitoring	H264_DVR_StopAlarmCenter Listen.....	71
3.17 Get device operating status information H264_DVR_GetDVR WorkState..		71
3.18 Network alarm H264_DVR_SendNetAlarmMsg.....		72
3.19 Disk Management H264_DVR_StorageManage.....		72
3.20 Device capture H264_DVR_CatchPic.....		73
3.21 Transparent serial port.....		73
3.21.1 Creating a Transparent Serial Port Channel	H264_DVR_OpenTrans ComChannel	73
3.21.2 Writing Data to the Device Through the Serial Port	H264_DVR_SerialWrite	74
3.21.3 Reading Data from the Device Through the Serial Port	H264_DVR_SerialRead	75
3.21.4 Disabling Transparent Serial Port Channels	H264_DVR_CloseTransCom Channel	76
3.22 Client recording		76
3.22.1 Starting Local Recording	H264_DVR_StartLocalRecord.....	76
3.22.2 Turning Off Local Recording	H264_DVR_StopLocalPlay	77
3.23 Client audio.....		77
3.23.1 Turning on Audio on the Video Channel	H264_DVR_OpenSound	77
3.23.2 Turning Off Audio on the Video Channel	H264_DVR_CloseSound	77
3.24 Play positioning		78
3.24.1 Get Playback Position (Percentage)	H264_DVR_GetPlayPos.....	78
3.24.2 Setting Playback Position (Percentage)	H264_DVR_SetPlayPos	78
3.25 Setting Info Frame Callbacks H264_DVR_SetInfoFrameCall Back.....		79
3.26 Client video color		80

3.26.1 Get Play Color Information	H264_DVR_LocalGetColor	80
3.26.2 Setting Play Color Information	H264_DVR_LocalSetColor	81
3.27 Playing Client Local Files.....		82
3.27.1 Playing Local Files	H264_DVR_StartLocalPlay	82
3.27.2 Turning off Local Playback	H264_DVR_StopLocalPlay.....	82
3.27.3 Local File Playback Callback	H264_DVR_SetFileEndCallBack	83
3.27.4 Local File Playback Control	H264_DVR_LocalPlayCtrl.....	83
3.28 Disconnection of Detector Connection H264_DVR_SetSub		
DisconnectCallBack		84
3.29 Set keep-alive time and break detection time H264_DVR_		
SetKeepLifeTime.....		85
3.30 Searching for Local Area Network Settings H264_DVR_Search Device		85
4 Error code enumeration		86
5 Sample function implementation.....		92

1. Brief introduction

1.1 Summary

Thank you for using the NetSDK Programming Manual from our company, as we known, the NetSDK is development kit used by software developers for exploiting online surveillance apps from our company's NVR (Network Video Recorder). This document describes the details of each functions, ports and invoking relationships as well as examples based on different functions in this SDK.

Files included in this SDK.

Network Library	NetSDK	Head file
	NetSDK.lib	Lib file
	NetSDK.dll	Interface library
Auxiliary Library	DllDeinterlace.dll	Decoding auxiliary library
	H264Play.dll	Decoding auxiliary library
	Hi_H264dec_w.dll	Decoding auxiliary library

1.2 Applicability

- Support NVR surveillance, playback, alarming, remote configuration, log query etc.
- Support TCP transport mode, and could connect 10 TCP simultaneously in front-end.
- Develop server programs such as stream media transmission, playback, alarming, through SDK callback interface.
- Preview images through multiple resolutions in client side, QCIF, CIF, 2CIF, HalfD1, D1, VGA (640*480) are the supported definitions.
- When SDK is operating video playback or download, the same login ID cannot run playback as well as download in the same channel at the same time.
- The performance of SDK is closely related with device running condition and CPU capability of computer. Theoretically, it can support 2000 registered users at the same time, internet preview as well as playback in the 2000

channels and upload alarm in 2000 channels. It can also support 300 channels in graphical presentation.

The principle of design

1.3 The instruction of programming

■ Initialization and clearance

1. When use Internet Client-side software packages, it should initialize system by calling [H264 DVR Init\(\)](#), and call [H264 DVR Cleanup\(\)](#) to release resources been taken as quit.
2. The most functions invoked should be put after [H264 DVR Init\(\)](#) and before [H264 DVR Cleanup\(\)](#), but the [H264 DVR GetLastError](#) could be called at any time.

■ Login and logout

Before visiting front-end devices, Users must invoke [H264 DVR Login\(\)](#) function to login devices. This handle just like a conversation channel, and then users can visit front-end equipment by the handle. If you want to quit this conversation, the handle can be ended in front-end devices with [H264 DVR Logout\(\)](#) function to expire this conversation channel. Finally, connection and login are synchronous.

■ Heartbeat function

This SDK provides auto heartbeat function(20 second per heartbeat), when device was cutoff, it could callback client-side in time.

■ Synchronization and asynchronization

Asynchronization is achieved by setting callback functions, and network data could be conveyed to application programs with callback functions. Some asynchronization returns to the required handle after setting, when requests finished, it will provide required handle to SDK to logout corresponding resources.

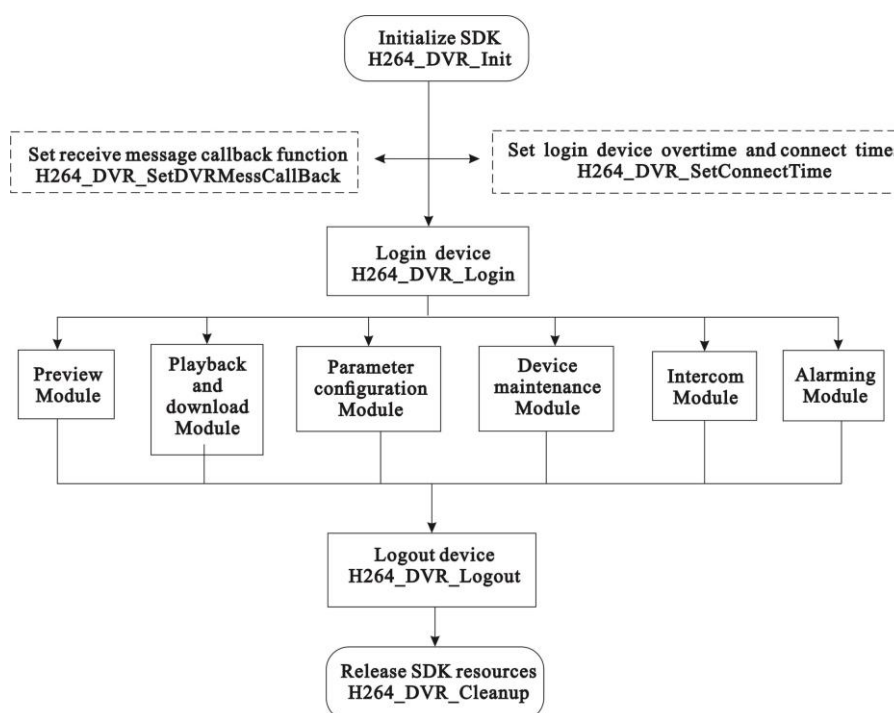
■ Callback functions

Generally, it has *dwUser* parameter on its own, users can define what data they need. Usually, this parameter could be used to introduce class object pointer to callback

achieved conveniently in class, and both callback applications could adapt this method.

1.4 Typical calling sequence

1.4.1 Basic flow chart of calling SDK



The flow charts in dashed boxes are optional, and they will not influence the functions of other charts and modules. According to difference of function achievement, it can be divided into 10 modules. If wanted to achieve each module's function, it must have four steps: initialized SDK, registered devices, logout devices and release SDK resources.

- Initialize SDK ([H264_DVR_Init\(\)](#)): Initialize the whole NetSDK system, pre-allocate the memory and so on.
- Set connection timeout ([H264_DVR_SetConnectTime](#)): This part is optional, and is used to set network connection timeout of SDK, users can set values according to their needs. System will adopt the default value if not calling this function.
- Set callback function of receiving abnormal message ([H264_DVR_SetDVRMessCallBack](#)): this port is used to receive abnormal message from alarming module. Users could set the callback function after initializing the SDK.

- User login devices ([H264 DVR Login](#)): Used to achieve login function, after login success, backward users' ID act as unique identification for other functional operations.
- Preview module: Pick up real time code stream from front-end devices, decode and display, play control and other functions. At the same time, support soft decoding as well as with decoding card to decode. Detailed process is shown in [Real-time preview module](#).
- Playback and download module: Remote playback or download video files by time and filename, decode or storage could enable in the future. Specific flowchart can be seen from [Playback and download module](#).
- Parameter configuration module: Set and obtain parameters from front-end devices, including parameters of devices, network, channels, ports, alarming, abnormal and users configuration. Detailed information can be seen from [Parameter configuration module](#).
- Remote device maintenance module: Shut down, reboot and remote upgrade and so on maintenance work can be fulfilled. See the detailed process from [Device maintenance module](#).
- Voice intercom transmit module: Voice data intercom and acquisition with front-end devices, audio coding format can be customized. Details refer to [Voice intercom module](#).
- Alarming module: Deal with several alarming signals uploaded from devices. Alarming can be divided into two patterns, disarming and monitoring. In the condition of adopting the pattern of monitoring as well as no need of users' ID, alarming module can skip "user login" step. Details refer to [Alarming module](#).
- Transparent channel module: The transparent channel is the technology of sending IP data to serial port after analyzing. SDK provides 485 and 232 serial port types separately. Details refer to [Transparent serial port channel module](#).

1.4.2 Related interface function of every module

A. Initialization

SDK initialization	H264 DVR Init ()
--------------------	----------------------------------

B. SDK functional information acquisition

Set message callback	H264_DVR_SetDVRMessCallBack ()
----------------------	--

C. Login devices

Login devices	H264_DVR_Login ()
Push channel of alarming message	H264_DVR_SetupAlarmChan ()

D. Real-time preview

Turn on the monitoring channel	H264_DVR_RealPlay ()
	H264_DVR_StopRealPlay ()
Callback and save of monitoring data	H264_DVR_SetRealDataCallBack ()

E. Device parameter configuration and control

Parameter configuration	H264_DVR_GetDevConfig ()
	H264_DVR_SetDevConfig ()
Find log	H264_DVR_FindDVRLog ()
PTZ control	H264_DVR_PTZControl ()
Transparent serial port control	H264_DVR_OpenTransComChannel ()
	H264_DVR_CloseTransComChannel ()

F. Playback/download channel

Find video	H264_DVR_FindFile ()
	H264_DVR_FindFileByTime()
Playback and control	H264_DVR_PlayBackByName()
	H264_DVR_PlayBackByTime()
	H264_DVR_PlayBackControl()
	H264_DVR_StopPlayBack()
Download	H264_DVR_GetFileByName ()
	H264_DVR_GetFileByTime()

<u>H264_DVR_GetDownloadPos()</u>
--

<u>H264_DVR_StopGetFile ()</u>
--

G. Remote control

Remote upgrade

<u>H264_DVR_Upgrade()</u>

<u>H264_DVR_GetUpgradeState()</u>

<u>H264_DVR_CloseUpgradeHandle()</u>
--

Reboot/Clear log

<u>H264_DVR_ControlDVR ()</u>

H. Logout device

Stop pushing alarming message

<u>H264_DVR_CloseAlarmChan ()</u>

Logout device

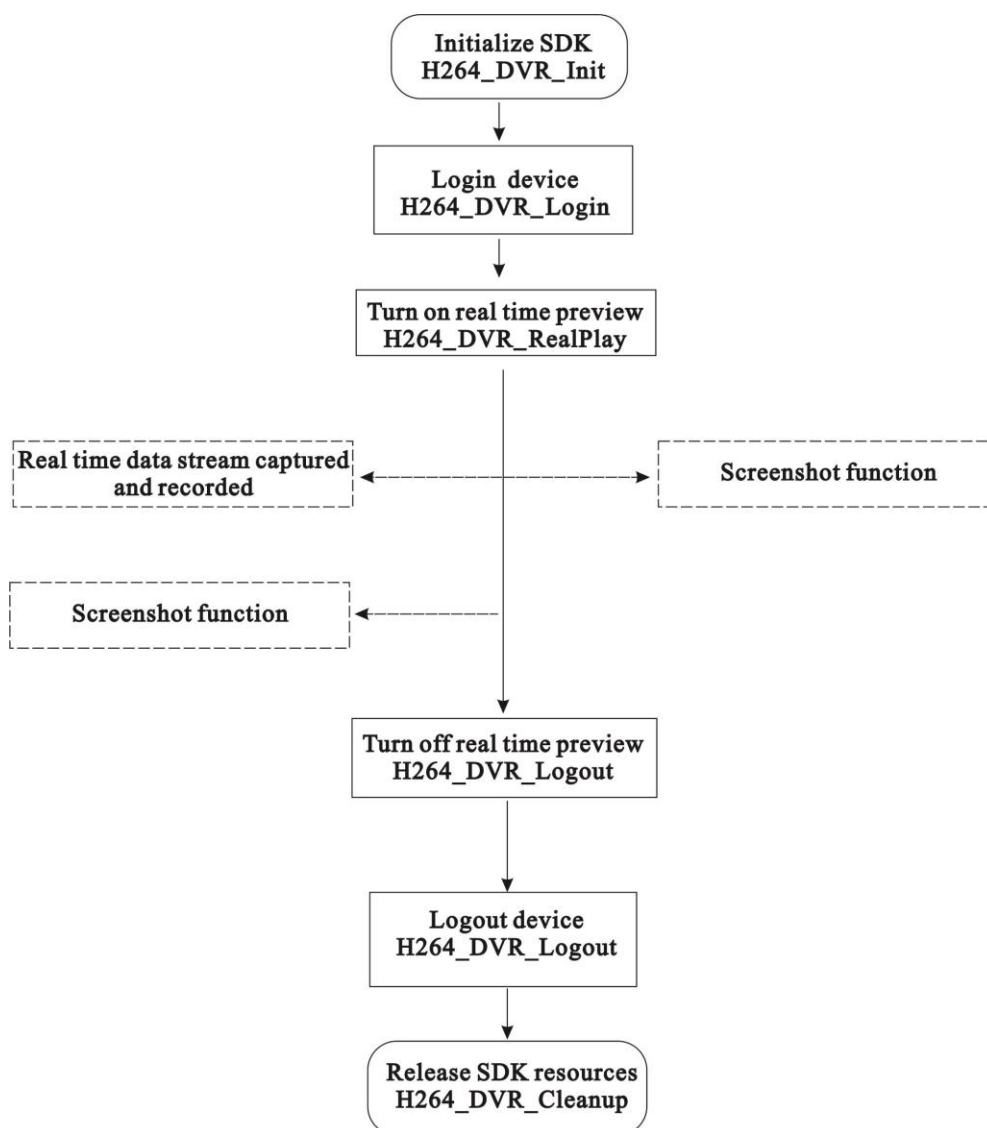
<u>H264_DVR_Logout ()</u>

I. Release SDK resources

Quit SDK

<u>H264_DVR_Cleanup ()</u>
--

1.4.3 Flow chart of real-time preview module



Modules shown in above picture in dashed boxes are related with preview module, and can be invoked after initiating preview. These modules are parallel to each other to complete the corresponding functions.

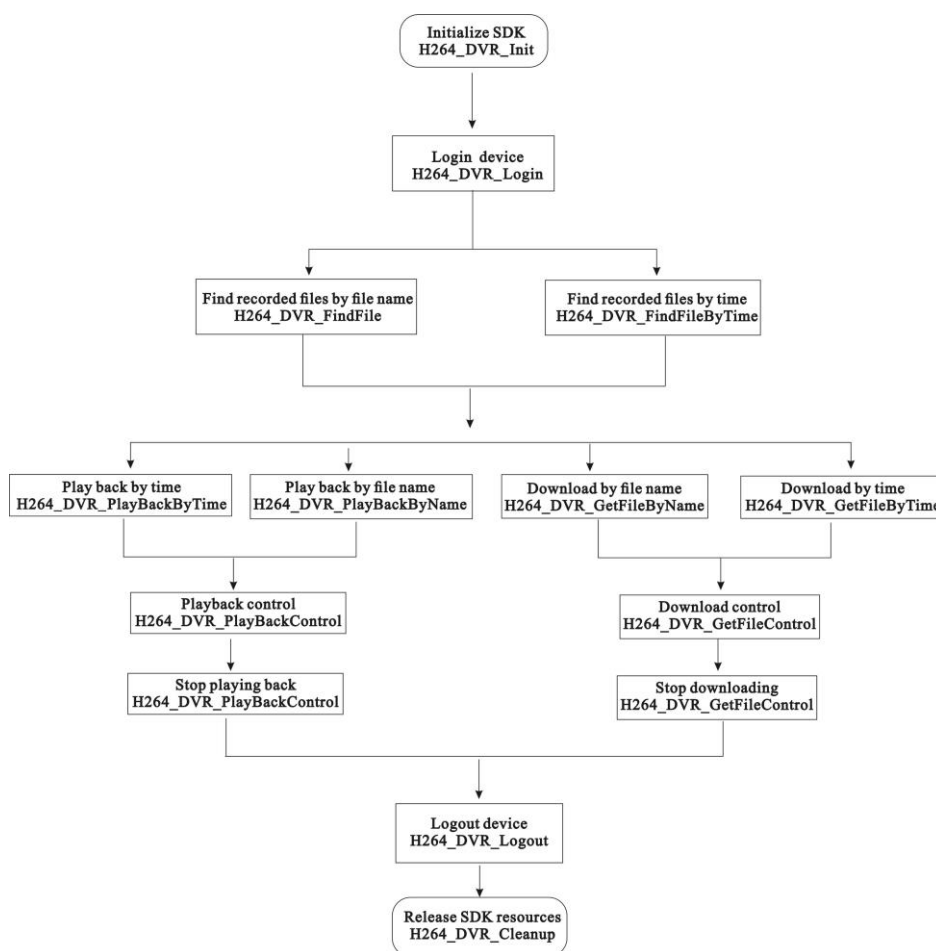
Corresponding functions:

- Real-time data capturing mainly enable real-time callback and local video functions. Related ports include: [H264_DVR_SetRealDataCallBack](#), [H264_DVR_DelRealDataCallBack](#), [H264_DVR_StartDVRRecord](#), [H264_DVR_StopDVRRecord](#) and so on.
- Screenshot function mainly enables capture current decoded pictures. Corrected ports include: [H264_DVR_CatchPic](#) (Caution: this screenshot port is not able to

conduct screenshot until the video configuration of device supports this function.)
or call H264_Play_CatchPic on screenshot port from PlaySDK (Note: only apply to preview) .

- PTZ control module mainly put control function on PTZ control, on the condition of turning on preview. Related ports include: [H264_DVR_PTZControl](#) etc.

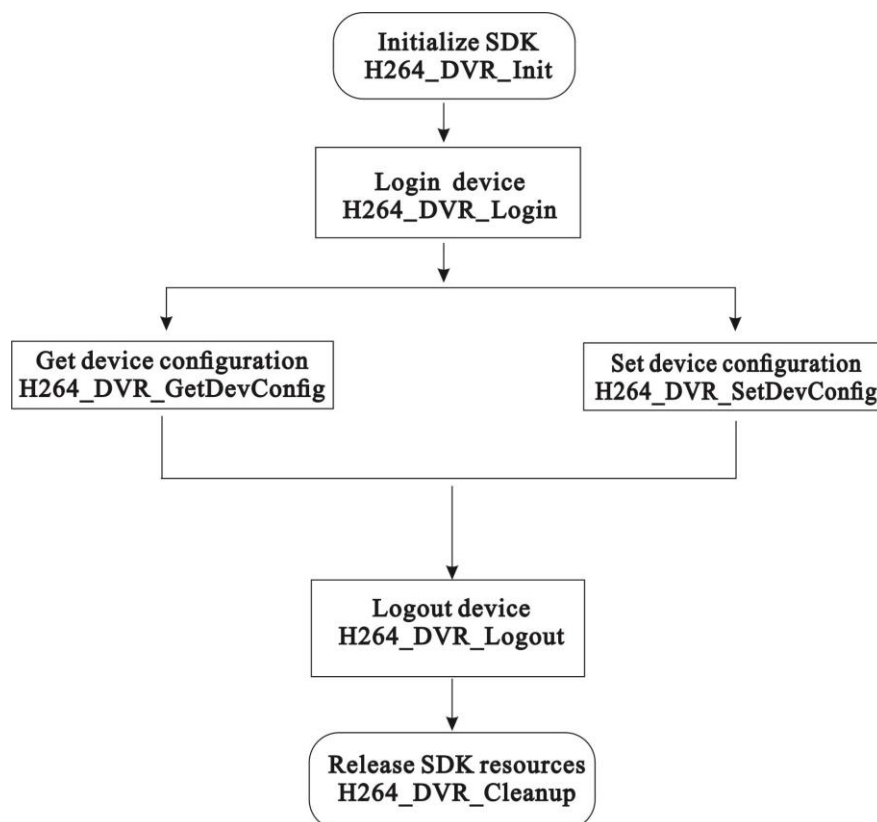
1.4.4 Playback and download module



- Playback or download according to files need to acquire file information by function of searching video files (related ports include [H264_DVR_FindFile](#)、[H264_DVR_FindFileByTime](#)), and then start playback or download refers to obtained file names (related ports include [H264_DVR_PlayBackByName](#)、[H264_DVR_GetFileBy Name](#)). After calling playback or download port, it needs to call control port ([H264_DVR_PlayBackControl](#)、[H264_DVR_GetFileControl](#)).
- Playback or download files according to time, users do not need to call related ports for finding video files, and only need to set the time of start and finish in the

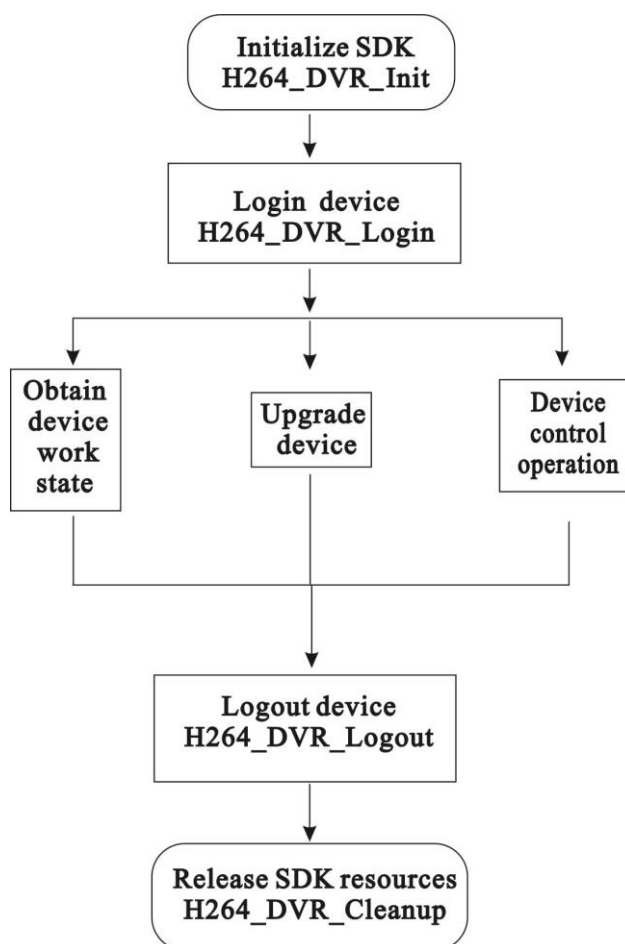
port, and call playback or download port (related ports include [H264 DVR PlayBackByTime](#)、[H264 DVR GetFileByTime](#)). Control port can be called if other operations needed to be done ([H264 DVR PlayBackControl](#)、[H264 DVR GetFileControl](#)).

1.4.5 Parameter configuration module



- In order to fulfill parameter configuration, SDK initialization and user registration must be done, and set the ID returned by the user registration interface as the first parameter of configuration interface. It is advised that users should call the port to get parameters each time before setting some kinds of parameters ([H264 DVR GetDevConfig](#)) to obtain whole parameter structure. Modify parameters need to be altered, and choose them as the input parameters of setting parameter interface. Finally, call the settings parameter interface ([H264 DVR SetDevConfig](#)), and if it is returned success, then sets successfully.

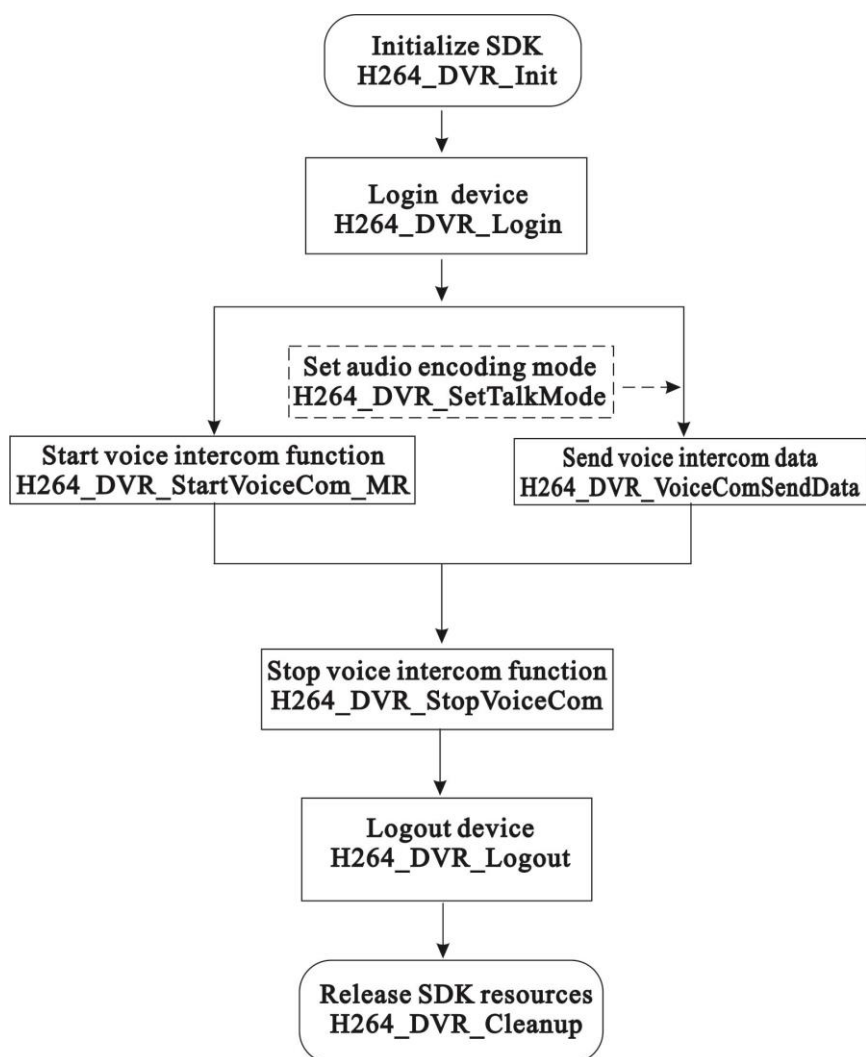
1.4.6 Device maintenance module



Some functions such as obtain device working condition, remote and local upgrades, reboot devices are included in remote device module.

- Acquire device operation condition: acquire device current channel condition, alarming input and output condition and so on. Related ports include [H264_DVR_Get DVRWorkState](#) etc.
- Device local and remote upgrades: upgrade devices, obtain current upgrade schedule and conditions. Related ports include [H264 DVR Upgrade](#) 、[H264 DVR Upgrade Cloud](#) etc.
- Device control operation: reboot device, cleanup logs and logout functions etc. Related ports include [H264_DVR_ControlDVR](#) etc.

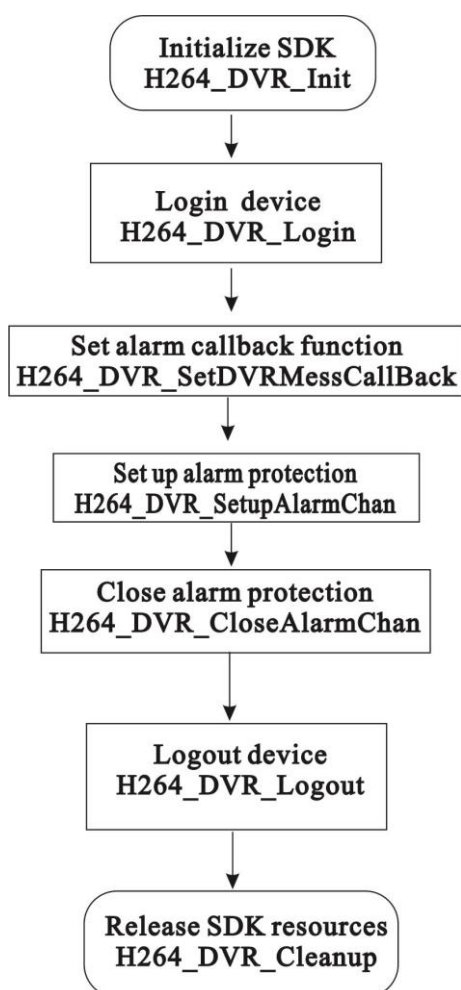
1.4.7 Voice intercom module



- Enable audio sending and receiving between PC and devices through voice intercom functions. After login device successfully, call [H264 DVR StartVoiceCom_MR](#) and complete interface, while users can acquire data from current devices through set callback function in this port (select the post-callback data as needed).
- Sending intercom data to devices, the encoding format can be set, the default is G711A. Related ports include [H264 DVR SetTalkMode](#)、[H264 DVR VoiceComSendData](#).

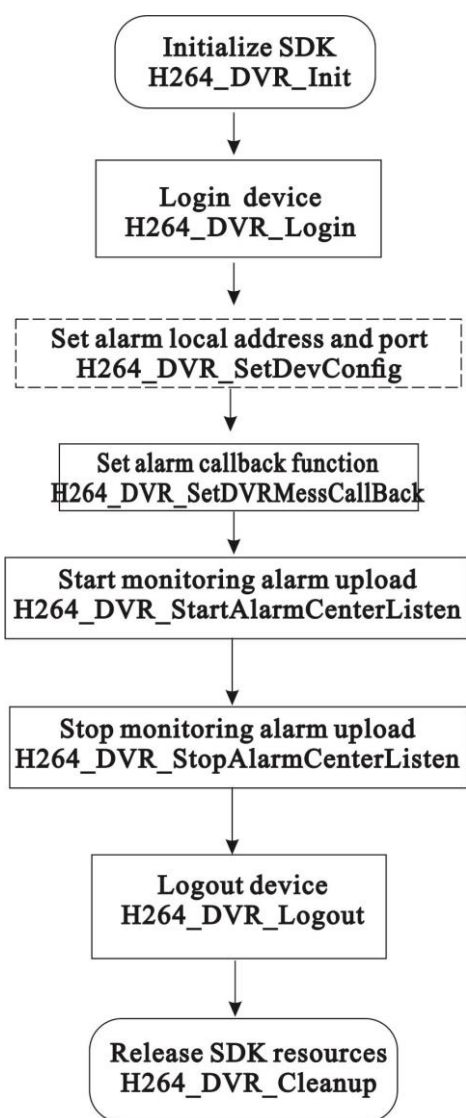
1.4.8 Alarming module

1.4.8.1 Protective alarm mode



- Type of alarming: SDK initiatively connects to the device and initiates the alarm uploading command, which will be sent to SDK immediately when the device alarms.
- From above flow chart of alarming (protection) module, we can see that protection method needs users to register firstly ([H264 DVR Login](#)). And then, set alarm callback function ([H264 DVR SetDVRMessCallBack](#)), and also needs to set protection after calling successfully ([H264 DVR SetupAlarmChan](#)). After the whole alarming processes uploaded successfully, it is necessary to call the disarming interface and other operations.

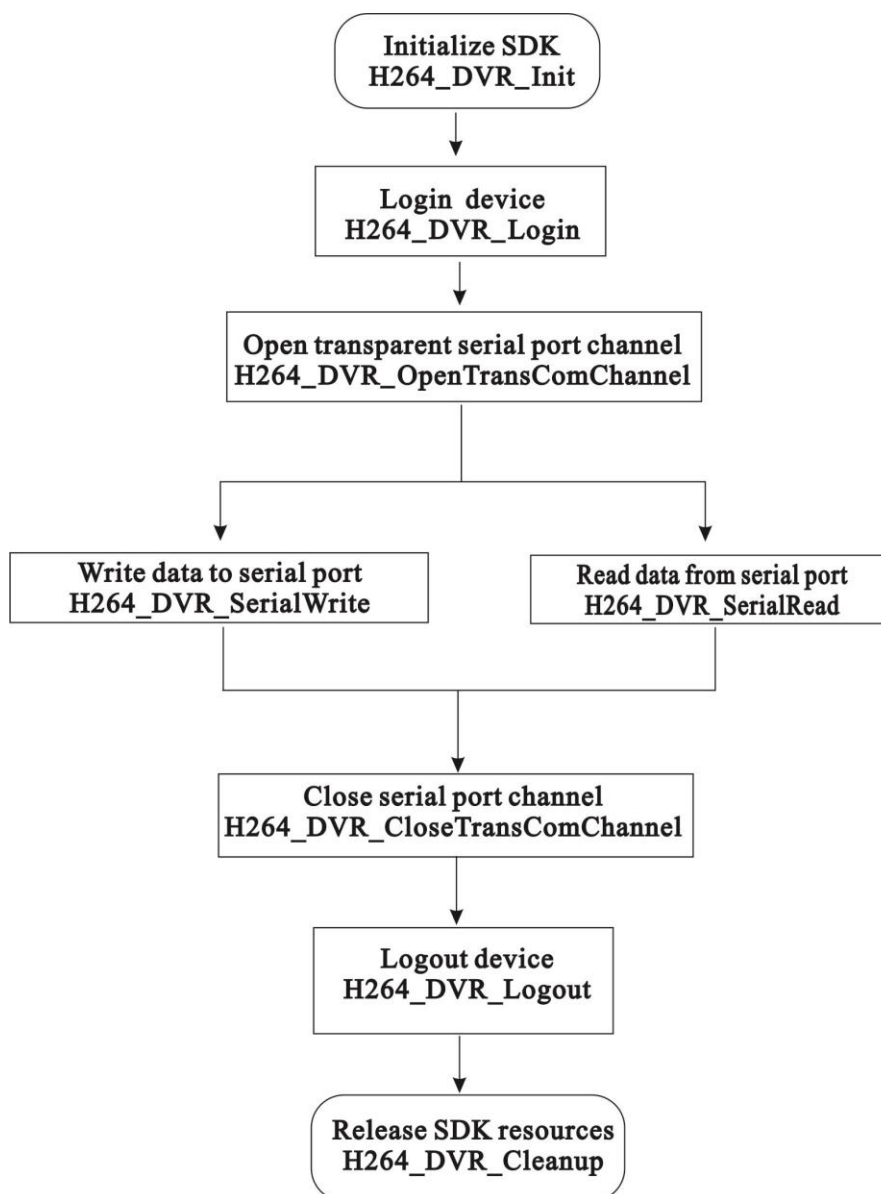
1.4.8.2 Monitoring alarm mode



- Alarming mode: the SDK does not connect devices automatically, but only monitor the alarming message uploaded by the receiving devices on the set port.
- This process needs to configure alarming local address (PC address) connected with devices and alarming host port (PC monitoring port) in remote, and the alarming host is monitoring in this port on alarming message auto-uploaded by receiving devices. If the configuration of alarming host address and port has been finished, then the user registration as well as alarming host address and port configuration in dashed boxes in the flow chart of monitoring module can be omitted, but if no configuration in advance, we must call parameters configuration interface ([H264 DVR SetDevConfig](#)) to deploy the device network parameters. After all the above parameters needed to be configured are set, the [H264 DVR StartAlarmCenterListen](#) function is called to open the SDK

listening port and prepare to receive the alarming message uploaded by the devices.

1.4.9 Transparent serial channel module



- SDK provides serial port types, such as 485 and 232. Call the [H264_DVR_OpenTrans ComChannel](#) to build transparent channels, write data to the serial port based on [H264_DVR_SerialWrite](#) function and read data from the serial port through [H264_DVR_SerialRead](#) function. After finishing all of these processes, you also need to disconnect transparent channels and so on ([H264_DVR_CloseTransComChannel](#)).

2 Definition of data structure

2.1 System time structure **SDK_SYSTEM_TIME**

```
typedef struct SDK_SYSTEM_TIME
{
    int year;           ///< Year。
    int month;          ///< Month, January = 1, February = 2, and so on.
    int day;            ///< Day。
    int wday;           ///< Weekday, Sunday = 0, Monday = 1, and so on
    int hour;           ///< Hour。
    int minute;         ///< Minute。
    int second;         ///< Second。
    int isdst;          ///< Summer time identifier。
}SDK_SYSTEM_TIME;
```

2.2 Related structures of video files

2.2.1 Query condition structures **H264_DVR_FINDINFO**

```
typedef struct
{
    int nChannelNO;      ///

```

2.2.2 Returned video information structures **H264_DVR_FILE_DATA**

```
typedef struct
{
    int ch;               ///

```

```

int size;                //Size of file
char sFileName[108];     ///< File name
SDK_SYSTEM_TIME stBeginTime; ///< File start time
SDK_SYSTEM_TIME stEndTime;    ///< File end time
void *hWnd;              // Output window handle (If it was NULL, Network
                          data is processed separately from decoding player.)
}H264_DVR_FILE_DATA;

```

2.2.3 Query the structures by the time period **SDK_SearchByTime**

```

typedef struct SDK_SearchByTime
{
    int nHighChannel;      ///< 33~64 video channel masked code

    int nLowChannel;       ///< 1~32 video channel masked code
    int nFileType;         ///< File type, see SDK_File_Type
    SDK_SYSTEM_TIME stBeginTime; ///< Query begin time
    SDK_SYSTEM_TIME stEndTime;   ///< Query end time
    int iSync;             ///< Whether needs to be synchronized
    unsigned int nHighStreamType; ///< 33~64 video code stream type, the binary bit
    represents main code stream and auxiliary code stream
    unsigned int nLowStreamType;  ///< 1~32 video code stream type, the binary bit
    represents main code stream and auxiliary code stream
}SDK_SearchByTime;

```

2.2.4 Related structures of the query results **SDK_SearchByTimeResult**

```

//Video information for each channel
typedef struct SDK_SearchByTimeInfo
{
    int iChannel;          ///< Video channel number
    ///< Using bytes represent minutes of the day in video recording
    ///< 0000: no video 0001:F_COMMON 0002:F_ALERT 0003:F_DYNAMIC
    0004:F_CARD 0005:F_HAND
    unsigned char cRecordBitMap[720];
}SDK_SearchByTimeInfo;

```

```
typedef struct SDK_SearchByTimeResult
{
    int nInfoNum;          ///< Video in channel records the number of information
    SDK_SearchByTimeInfo ByTimeInfo[NET_MAX_CHANNUM];  ///< The
    information of recorded video in channel
}SDK_SearchByTimeResult;
```

2.3 Structure of configuration information SDK_CONFIG_TYPE

Definition of the order of [H264 DVR GetDevConfig](#)、[H264 DVR SetDevConfig](#)

Definition of dwCommand	Functions	Related structures
E_SDK_CONFIG_USER	User information, includes permission list, users list and group list	USER_MANAGE_INFO
E_SDK_CONFIG_ADD_USER	Add user	USER_INFO
E_SDK_CONFIG_MODIFY_USER	Modify user	CONF_MODIFYUSER
E_SDK_CONFIG_DELETE_USER	Delete user	USER_INFO
E_SDK_CONFIG_ADD_GROUP	Add group	USER_GROUP_INFO
E_SDK_CONFIG_MODIFY_GROUP	Modify group	CONF_MODIFYGROUP
E_SDK_CONFIG_DELETE_GROUP	Delete group	USER_GROUP_INFO
E_SDK_CONFIG_MODIFY_PSW	Modify password	CONF_MODIFY_PSW
E_SDK_CONFIG_ABILITY_SYSFUNC	Network capabilities	SDK_SystemFunction
E_SDK_CONFIG_ABILITY_ENCODE	First encode ability	CONFIG_EncodeAbility
E_SDK_CONFIG_ABILITY_PTZPRO	PTX protocol	SDK_PTZPROTOCOLFUNCTION
E_SDK_CONFIG_ABILITY_COMMPRO	Serial port protocol	SDK_COMMFUNC

E_SDK_CONFIG_ABILITY_MOTION_FUNC	Fast dynamic detection	SDK_MotionDetectFunction
E_SDK_CONFIG_ABILITY_BLIND_FUNC	Fast video block	SDK_BlindDetectFunction
E_SDK_CONFIG_ABILITY_DDNS_SERVER	DDNS service function	SDK_DDNSServiceFunction
E_SDK_CONFIG_ABILITY_TALK	Intercom decode type	SDK_DDNSServiceFunction
E_SDK_CONFIG_SYSTEM_INFO	System information	H264_DVR_DEVICEINFO
E_SDK_CONFIG_SYSNORMAL	General configuration	SDK_CONFIG_NORMAL
E_SDK_CONFIG_SYSENCODE	Encode configuration	SDK_EncodeConfigAll
E_SDK_CONFIG_SYSNET	Network configuration	SDK_CONFIG_NET_COMMON
E_SDK_CONFIG_PTZ	PTZ page	SDK_STR_PTZCONFIG_ALL
E_SDK_CONFIG_COMM	Serial port page	SDK_CommConfigAll
E_SDK_CONFIG_RECORD	Video set interface	SDK_RECORDCONFIG
E_SDK_CONFIG_MOTION	Dynamic detection page	SDK_MOTIONCONFIG
E_SDK_CONFIG_SHELLTER	Video blocked	SDK_BLINDDETECTCONFIG
E_SDK_CONFIG_VIDEO_LOSS	Video lost	SDK_VIDEOLOSSCONFIG
E_SDK_CONFIG_ALARM_IN	Alarm input	SDK_ALARM_INPUTCONFIG
E_SDK_CONFIG_ALARM_OUT	Alarm output	SDK_AlarmOutConfigAll
E_SDK_CONFIG_DISK_MANAGER	Disk management interface	SDK_StorageDeviceControl
E_SDK_CONFIG_OUT_MODE	Out mode interface	SDK_VideoWidgetConfigAll
E_SDK_CONFIG_CHANNEL_NAME	Channel name	SDK_ChannelNameConfigAll
E_SDK_CONFIG_AUTO	Auto maintain configuration	SDK_AutoMaintainConfig

E_SDK_CONFIG_DEFAULT	Renew default interface configuration	SDK_SetDefaultConfigTypes
E_SDK_CONFIG_DISK_INFO	Disk information	SDK_StorageDeviceInformationAll
E_SDK_CONFIG_LOG_INFO	Query log	SDK_LogList
E_SDK_CONFIG_NET_IPFILTER	Blacklist and whitelist configuration	SDK_NetIPFilterConfig
E_SDK_CONFIG_NET_DHCP	DHCP configuration	SDK_NetDHCPConfigAll
E_SDK_CONFIG_NET_DDNS	DDNS information	SDK_NetDDNSConfigALL
E_SDK_CONFIG_NET_EMAIL	EMAIL	SDK_NetEmailConfig
E_SDK_CONFIG_NET_MULTICAST	Multicast	SDK_NetMultiCastConfig
E_SDK_CONFIG_NET_NTP	NTP	SDK_NetNTPConfig
E_SDK_CONFIG_NET_PPPOE	PPOE	SDK_NetPPPoEConfig
E_SDK_CONFIG_NET_DNS	DNS	SDK_NetDNSConfig
E_SDK_CONFIG_NET_FTPSERVER	FTP	SDK_FtpServerConfig
E_SDK_CONFIG_SYS_TIME	System time	SDK_SYSTEM_TIME
E_SDK_CONFIG_ABILITY_LANG	Support languages	SDK_MultiLangFunction
E_SDK_CONFIG_COMBINEENCODE	Combined encode	SDK_CombineEncodeConfigAll
E_SDK_CONFIG_COMBINEENCODEMODE	Combined encode mode	SDK_CombEncodeModeAll
E_SDK_WORK_STATE	Workstate	SDK_DVR_WORKSTATE
E_SDK_ABILITY_LANGLIST	Actual supported language set	SDK_MultiLangFunction
E_SDK_CONFIG_NET_ARSP	ARSP	SDK_NetARSPConfigAll
E_SDK_CONFIG_SNAPSHOT_STORAGE	Snapshot set	SDK_SnapshotConfig

E_SDK_CONFIG_NET_3G	3G dial	SDK_Net3GConfig
E_SDK_CONFIG_NET_MOBILE	Mobile monitoring	SDK_NetMoblieConfig
E_SDK_CONFIG_UPGRADEINFO	Obtain upgrade information	SDK_UpgradeInfo
E_SDK_ABILITY_VSTD	Actual supported video standard	SDK_MultiVstd
E_SDK_CONFIG_NET_UPNP	UPNP set	SDK_NetUPNPConfig
E_SDK_CONFIG_NET_WIFI	WIFI	SDK_NetWifiConfig
E_SDK_CONFIG_NET_WIFI_AP_LIST	Searched WIFI list	SDK_NetWifiDeviceAll
E_SDK_CONFIG_SYSENCODE_SIMPLIFY	Simplified encoding configuration	SDK_EncodeConfigAll_SIMPLIFY
E_SDK_CONFIG_ALARM_CENTER	Alarming center	SDK_NetAlarmServerConfigAll
E_SDK_CONFIG_NET_ALARM	Network alarming	SDK_NETALARMCONFIG_ALL
E_SDK_CONFIG_NET_PHONMSG	Short Message	SDK_NetShortMsgCfg
E_SDK_CONFIG_NET_PHONMEDIAMSG	Multimedia message	SDK_NetMultimediaMsgCfg
E_SDK_CONFIG_NET_RTSP	RTSP	SDK_NetRTSPConfig
E_SDK_CONFIG_COM485	Serial port 485 protocol configuration	SDK_STR_RS485CONFIG_ALL
E_SDK_CONFIG_ABILITY_COMMPRO485	Serial port 485 protocol	SDK_COMMFUNC
E_SDK_CONFIG_SYSTEM_TIME_NORTC	Set system time noRTC	SDK_SYSTEM_TIME
E_SDK_CONFIG_CHANNEL_TILE_DOT	The dot matrix information to modify IPC channel name	SDK_TitleDot
E_SDK_CONFIG_CAMERA	Camera parameter	SDK_CameraParam
E_SDK_CONFIG_ABILITY_CAMERA	Camera capability level	SDK_CameraAbility
E_SDK_CONFIG_STORAGE	Hard disk not found	SDK_VIDEOLOSSCON

AGENOTEXIST		FIG
E_SDK_CONFIG_STORAGE_LOWSPACE	Hard disk storage low space	SDK_StorageLowSpaceConfig
E_SDK_CONFIG_STORAGE_FAILURE	Hard disk failed	SDK_StorageFailConfig
E_SDK_CFG_NET_IP_CONFLICT	IP Conflict	SDK_VIDEOLOSSCONFIG
E_SDK_CFG_NETWORK_ABNORMAL	Network abnormal	SDK_VIDEOLOSSCONFIG
E_SDK_CONFIG_CHANNEL_STATUS	Channel status	SDK_NetDecoderChnStatusAll
E_SDK_CONFIG_CHANNEL_MODE	Channel mode	SDK_NetDecoderChnModeConfig
E_SDK_CONFIG_NETWORK_DAS	Active registration	SDK_DASSerInfo
E_SDK_CONFIG_CAR_INPUT_EXCHANGE	Correspondence relationship between external information input and vehicle status	SDK_CarStatusExchangeAll
E_SDK_CONFIG_DELAY_TIME	On-board system delay configuration	SDK_CarDelayTimeConfig
E_SDK_CONFIG_NETWORK_ORDER	Network priority	SDK_NetOrderConfig
E_SDK_CONFIG_ABILITY_NETWORK_ORDER	Ability to set network priority	SDK_NetOrderFunction
E_SDK_CONFIG_GPS_TIMING	GPS timing configuration	SDK_GPSTimingConfig
E_SDK_CONFIG_VIDEO_ANALYZE	Video analysis	SDK_ANALYSECONFIG
E_SDK_CONFIG_NAT_STATUS_INFO	Nat status information	SDK_NatStatusInfo
E_SDK_CONFIG_MEDIA_WATERMARK	Watermark configuration	SDK_WaterMarkConfigAll
E_SDK_CONFIG_ENCODE_STATICPARAM	Encode static parameter	SDK_EncodeStaticParamAll
E_SDK_CONFIG_DIG_MANAGER_SHOW	Channel management display configuration	SDK_DigManagerShowStatus
E_SDK_CONFIG_ABILITY_ANALYZEABILITY	Intelligent analysis ability	SDK_ANALYZEABILITY

E_SDK_CONFIG_NAT	NAT function, MTU value configuration	SDK_NatConfig
E_SDK_CONFIG_CPCINFO	Intelligent CPC enumeration data information	SDK_CPCDataAll
E_SDK_CONFIG_STORAGE_POSITION	Device type of video storage	SDK_RecordStorageType
E_SDK_CONFIG_ABILITY_CARSTATUSNUM	Car status number	SDK_CarStatusNum
E_SDK_CFG_VPN	VPN	SDK_VPNConfig
E_SDK_CFG_VIDEOOUTPUT	VGA video resolution	SDK_VGAresolution
E_SDK_CFG_ABILITY_VGARESOLUTION	Supported VGA video resolution list	SDK_VGAResolutionAbility
E_SDK_CFG_NET_LOCAL_SEARCH	Search device, LAN devices on the device side	SDK_NetDevList
E_SDK_CFG_ENCODE_STATICPARAM_V2	Static parameters of DVR encoder	SDK_EncodeStaticParamV2
E_SDK_ABILITY_ENCODE_STATICPARAM	Static encoding capability set	SDK_EncStaticParamAbility
E_SDK_CFG_MAIL_TEST	Email test	SDK_NetEmailConfig
E_SDK_CFG_SPVMN_PLATFORM	28181 protocol configuration	SDK_ASB_NET_VSP_CONFIG
E_SDK_CFG_PMS	Mobile phone service	SDK_PMSConfig
E_SDK_CFG_OSD_INFO	Screen prompt message	SDK_OSDInfoConfigAll
E_SDK_CFG_DIGITAL_REAL	Actually supported channel modes	SDK_VideoChannelManage
E_SDK_ABILITY_PTZ_CONTROL	PTZ control level	SDK_PTZControlAbility
E_SDK_CFG_PARAM_EX	Camera extended parameters	SDK_CameraParamEx
E_SDK_GPS_STATUS	GPS linkage information	SDK_GPSStatusInfo
E_SDK_WIFI_STATUS	WIFI linkage information	SDK_WifiStatusInfo
E_SDK_3G_STATUS	3G linkage information	SDK_WirelessStatusInfo
E_SDK_DAS_STATUS	Active registration status	SDK_DASStatusInfo
E_SDK_ABILITY_DECODE_DELEY	Capability of decoding strategy	SDK_DecodeDeleyTimePrame

E_SDK_CFG_DECODE_PARAM	Decoding maximum delay	SDK_DecodeParam
E_SDK_ABILITY_ONVIF_SUB_PROTOCOL	Onvif sub-protocol	SDK_AbilityMask
E_SDK_CFG_CAR_BOOT_TYPE	On-board switch mode	SDK_CarBootTypeConfig
E_SDK_CFG_IPC_ALARM	IPC network alarming	SDK_IPCArmConfigAll
E_SDK_CFG_TIME_ZONE	Time zone configuration	SDK_TimeZone
E_SDK_ABILITY_MAX_PRE_RECORD	Maximum pre-recording time can be set to 30	SDK_AbilityMask
E_SDK_CFG_DIGITAL_TIME_SYNC	Digital channel time synchronization configuration	SDK_TimeSynParam
E_SDK_CFG_DIGITAL_ENCODE	Digital channel compact coding configuration	SDK_EncodeConfigAll_SIMPLIFY
E_SDK_CFG_DIGITAL_ABILITY	Coding ability of digital channels	SDK_DigitDevInfo
E_SDK_CFG_ENCODE_CH_DISPLAY	The IE side encoding configuration shows the front-end channel number	SDK_EncodeChDisplay
E_SDK_CFG_RESUME_PTZ_STATE	Turn on the platform status	SDK_ResumePtzState
E_SDK_ABILITY_AHD_ENCODE_L	AHDL capability set	SDK_AHDEncodeLMask
E_SDK_CFG_SPEED_ALARM	Speed alarm	SDK_SpeedAlarmConfigAll
E_SDK_CFG_CORRESPONDENT_INFO	Customized configuration	SDK_CorrespondentOwnInfo
E_SDK_SET_OSDINFO	OSD message settings (this function only supports analogue channels)	SDK_OSDInfo
E_SDK_SET_OSDINFO_V2	OSD information overlays, unsaved configuration (no superposition effect after power off and reboot) Note: The IPC devices transmit characters in a lattice.	SDK_OSDInfoConfigAll

E_SDK_ABILITY_SUPPORT_EXTSTREAM	Support auxiliary code stream video recording	SDK_AbilityMask
E_SDK_CFG_EXT_RECORD	Auxiliary code flow configuration	SDK_RECORDCONFIG_ALL/ SDK_RECORDCONFIG
E_SDK_CFG_UPGRADE_VERSION_LIST	List of cloud upgrade files	SDK_CloudUpgradeList
E_SDK_OPERATION_SET_LOGO	Video is superimposed with the manufacturer's LOGO	SDK_SetLogo
E_SDK_OPERATION_SPLIT_CONTROL	Image split mode	SDK_SplitControl
E_SDK_OPERATION_UTC_TIME_SETTING	Set UTC time	SDK_SYSTEM_TIME
E_SDK_CFG_ENCODE_SmartH264	SmartH264+ configuration	SDK_SmartH264ParamAll
E_SDK_CFG_WIFI_INFO	Wireless WIFI information	SDK_WifiInfo
E_SDK_CFG_NET_RTMP	RTMP protocol	SDK_NetRTMPConfig
E_SDK_CFG_SNAPSHOT_SCHEDULE	Timed screensnap configuration	SDK_SnapConfigAll
E_SDK_CFG_PTZPRESET	Preset point configuration	SDK_PtzPreset
E_SDK_CFG_PTZTOUR	Tour configuration	SDK_PtzTour
E_SDK_CFG_PWD_SAFETY	Configuration of security issue (used to reset password)	SDK_PasswordSafety
E_SDK_ABILITY_QUESTION_DELIVERY	Obtain retrieval problem of password	SDK_QuestionDelivery

2.4 Related structures of hard disk storage control

2.4.1 Storage device control types SDK_StorageDeviceControlTypes

enum SDK_StorageDeviceControlTypes

```
{
    SDK_STORAGE_DEVICE_CONTROL_SETTYPE,    ///< Set type
    SDK_STORAGE_DEVICE_CONTROL_RECOVER,    ///< Recover error
```

```

    SDK_STORAGE_DEVICE_CONTROL_PARTITIONS,    ///< Partitions
    SDK_STORAGE_DEVICE_CONTROL_CLEAR,          ///< Clear

    SDK_STORAGE_DEVICE_CONTROL_ADDNAS,         ///

```

2.4.2 Storage device control **SDK_StorageDeviceControl**

```

typedef struct SDK_StorageDeviceControl
{
    int iAction;           ///< Refer to enum SDK_StorageDeviceControlTypes
    int iSerialNo;         ///< Disk series number
    int iPartNo;           ///< Partition number
    int iType;             ///< enum SDK_StorageDeviceClearTypes or
                           SDK_FileSystemDriverTypes
    int iPartSize[4/*MAX_DRIVER_PER_DISK*/];  ///< Size of each part
}SDK_StorageDeviceControl;

```

2.5 Related structures of frame information

```

enum MEDIA_PACK_TYPE
{
    FILE_HEAD = 0,        // File head
    VIDEO_I_FRAME = 1,    // Video I frame
    VIDEO_B_FRAME = 2,    // Video B frame
    VIDEO_P_FRAME = 3,    // Video P frame
    VIDEO_BP_FRAME = 4,   // Video BP frame
    VIDEO_BBP_FRAME = 5,  // Video B and BP frames
    VIDEO_J_FRAME = 6,    // Picture frame
    AUDIO_PACKET = 10,    // Audio packet
};

```

```

enum SDK_ENCODE_TYPE
{
    SDK_StreamTypeEmpty = 0,
    SDK_StreamTypeH264 = 2,
    SDK_StreamTypeJpeg = 3,
    SDK_StreamTypeGeneral = 4,
    SDK_StreamTypeH265 = 5,
    SDK_StreamTypePCM8 = 7,
    SDK_StreamTypeStd = 8
};

typedef struct
{
    int            nPacketType;           //Packet type, refer to MEDIA_PACK_TYPE
    char*          pPacketBuffer;        //Buffer address
    unsigned int   dwPacketSize;         //Packet size

    unsigned int   nEncodeType;          //Data format type refers to SDK_ENCODE
    _TYPE

    // Absolute timestamp
    int            nYear;                //Timestamp: year
    int            nMonth;               //Timestamp: month
    int            nDay;                 //Timestamp: day
    int            nHour;                //Timestamp: hour
    int            nMinute;              //Timestamp: minute
    int            nSecond;              //Timestamp: second
    unsigned int   dwTimeStamp;          //Relative timestamp low level, unit in
    millisecond
    unsigned int   dwTimeStampHigh;      //Relative timestamp high level, unit in
    millisecond
    unsigned int   dwFrameNum;           //Frame number
    unsigned int   dwFrameRate;          //Frame rate

```

```

unsigned short uWidth;           //Width of picture
unsigned short uHeight;         //Height of picture
unsigned int   Reserved[6];      //Reserved

} PACKET_INFO_EX;

```

2.6 Local playaction control

SDK_LoalPlayAction

//Local playaction control

```
enum SDK_LoalPlayAction
```

```

{
    SDK_Local_PLAY_PAUSE,      /*<! Pause playing*/
    SDK_Local_PLAY_CONTINUE,   /*<! Continue playing*/
    SDK_Local_PLAY_FAST,       /*<! Play fast*/
    SDK_Local_PLAY_SLOW,       /*<! Play slow*/
};

```

2.7 Subconnection type

SubConnType

```
typedef enum SubConnType
```

```

{
    conn_realTimePlay=1,
    conn_talk,
    conn_playback,
    conn_push

}SubConnType;

```

2.8 Socket style

SocketStyle

```
enum SocketStyle
```

```

{
    TCPSOCKET=0,
    UDPSOCKET,
    PLUGLANSOCKET=4,           //Socket LAN login
    PLUGOUTERSOCKET,          //Socket out network login
    P2P_TUTKSOCKET,           //TUTK P2P

```



```

    SOCKETNR
};

```

2.9 Related structures of cloud upgrade SDK_CloudUpgradeVersion

```

typedef struct SDK_CloudUpgradeVersion
{
    char name[128];           // Version name
    char date[12];           //Date of version, format: "2014-08-26"
    unsigned int length;     // Length of upgrade file
}SDK_CloudUpgradeVersion;

```

2.10 Related information of serial port

2.10.1 Serial port type SERIAL_TYPE

```

typedef enum SERIAL_TYPE
{
    RS232 = 0,
    RS485 = 1,
}SERIAL_TYPE;

```

2.10.2 Related information of serial port TransComChannel

```

typedef struct __TransComChannel    //Transparent serial port
{
    SERIAL_TYPE TransComType; //SERIAL_TYPE
    unsigned int baudrate;
    unsigned int databits;
    unsigned int stopbits;
    unsigned int parity;
} TransComChannel;

```

2.11 Related information of playback action

```

// Playback action
enum SDK_PlayBackAction
{

```

```

    SDK_PLAY_BACK_PAUSE,           /*<! Playback pause*/
    SDK_PLAY_BACK_CONTINUE,        /*<! Playback continue*/
    SDK_PLAY_BACK_SEEK,            /*<! Playback positioning, time unit in s*/
    SDK_PLAY_BACK_FAST,            /*<! Playback fast*/
    SDK_PLAY_BACK_SLOW,            /*<! Playback slow*/
    SDK_PLAY_BACK_SEEK_PERCENT,    /*<! Playback positioning percent*/
    SDK_PLAY_SET_TYPE,             /*<! Playback intelligent positioning*/
};

```

2.12 Related structures of callback data in active service

2.12.1 Device information H264_DVR_DEVICEINFO

```

typedef struct _H264_DVR_DEVICEINFO
{
    char sSoftWareVersion[64];    ///< Software version information
    char sHardWareVersion[64];    ///< Hardware version information
    char sEncryptVersion[64];     ///< Encrypt version information
    SDK_SYSTEM_TIME tmBuildTime;  ///< Software built time
    char sSerialNumber[64];       ///< Device serial number
    int byChanNum;                 ///< Channel number of video in
    int iVideoOutChannel;          ///< Channel number of video out
    int byAlarmInPortNum;          ///< Channel number of alarm in
    int byAlarmOutPortNum;         ///< Channel number of alarm out
    int iTalkInChannel;            ///< Channel number of intercom in
    int iTalkOutChannel;           ///< Channel number of intercom out
    int iExtraChannel;             ///< Extended channel number
    int iAudioInChannel;           ///< Channel number of audio in
    int iCombineSwitch;            ///< Whether partition mode of the combined
encoding channel supports switching
    int iDigChannel;               ///< Channle number of digital
    unsigned int uiDeviceRunTime;  ///< System run time
    SDK_DeviceType deviceTye;      ///< Device type
    char sHardWare[64];           ///< Device type

```

```

char uUpdataTime[20];    ///< Updata time, such as 2013-09-03 14:15:13
unsigned int uUpdataType; ///< Updata content
char sDeviceModel[16];    // Device model (underlying library obtained
from encryption, sHardWare is indistinguishable from using the same program for
multiple devices)
int nLanguage;    // Language ID of country, 0 refers to English, 1 refers to
Simplified Chinese, 2 refers to traditional Chinese, 3 refers to Korean, 4 refers to
German, 5 refers to Portuguese, 6 refers to Russian
char sCloudErrCode[NET_MAX_PATH_LENGTH];    // Specific error content
of cloud login
int status[32];    //Judge the new coming connection is whether transmited by
agent, if it was ,then limite it according to the restrictions returned by the server
//status[0] Channel number limited:0 refers to no limit, n refers to limiting n
channel
//status[1] Code stream limitation. : no limits. Watch the main code stream
limited
//status[2] Limited time. : no limits. n: limit n minutes。
//status[3] Limited code rate, there are four levels at present. :no limits. :limited to
CIF 6 frame 100K, the follow-up needs to be determined
//status[4] Reserved bits, subsequent expansion
// Among them, status[0] and status[1] reflected here; Status[2] and status[3]
reflected in the process of transmitting code stream
}H264_DVR_DEVICEINFO,*LPH264_DVR_DEVICEINFO;

```

2.12.2 Active service of callback data H264_DVR_ACTIVEREG_INFO

```

typedef struct H264_DVR_ACTIVEREG_INFO
{
    char deviceSarialID[64];    //Device serial ID, assigne a value if it is
greater than a bit
    H264_DVR_DEVICEINFO deviceInfo;    //Device information
    char IP[IP_SIZE];    //Outer network IP
}H264_DVR_ACTIVEREG_INFO;

```

2.12.3 Related structures of active registration configuration SDK_DASSerInfo

```
typedef struct SDK_DASSerInfo
{
    bool enable;
    char serAddr[NET_NAME_PASSWORD_LEN];
    int port;
    char userName[NET_NAME_PASSWORD_LEN];
    char passwd[NET_NAME_PASSWORD_LEN];
    char devID[NET_NAME_PASSWORD_LEN];
}SDK_DASSerInfo;
```

2.13 Related structures of alarming center

2.13.1 Related configuration of alarming center SDK_NetAlarmServerConfigAll

//IP addr

```
typedef union
```

```
{
    unsigned char    c[4];
    unsigned short   s[2];
    unsigned int      l;
}CONFIG_IPAddress;
```

///< Definition of server structure

```
typedef struct SDK_RemoteServerConfig
{
    char ServerName[NET_NAME_PASSWORD_LEN];    ///< Server name
    CONFIG_IPAddress ip;                       ///< IP address
    int Port;                                  ///< Port number
    char UserName[NET_NAME_PASSWORD_LEN];      ///< User name
    char Password[NET_NAME_PASSWORD_LEN];      ///< Password
    bool Anonymity;                            ///< Whether login in anonymity
}SDK_RemoteServerConfig;
```

///< Alarming center setting

```
typedef struct SDK_NetAlarmCenterConfig
```

```
{
    bool bEnable;                             ///< Wheter enabled
}
```

```

    char sAlarmServerKey[NET_NAME_PASSWORD_LEN]; ///< The name of
protocol type in alarming center
    SDK_RemoteServerConfig Server;          ///< Alarming center server
    bool bAlarm;
    bool bLog;
}SDK_NetAlarmCenterConfig;

```

```

typedef struct SDK_NetAlarmServerConfigAll
{
    SDK_NetAlarmCenterConfig
vAlarmServerConfigAll[NET_MAX_ALARMSEVER_TYPE];

}SDK_NetAlarmServerConfigAll;

```

2.13.2 Alarming center message content SDK_NetAlarmCenterMsg

```

//IP addr
typedef union
{
    unsigned char    c[4];
    unsigned short   s[2];
    unsigned int      l;
}CONFIG_IPAddress;

// Alarming center message content
typedef struct SDK_NetAlarmCenterMsg
{
    CONFIG_IPAddress HostIP;          ///< Device IP
    int  nChannel;                    ///< Channel
    int  nType;                       ///< Types in larmCenterMsgType
    int  nStatus;                     ///< Status in AlarmCenterStatus
    SDK_SYSTEM_TIME Time;              ///< Time happen
    char sEvent[NET_MAX_INFO_LEN];    ///< Event
    char sSerialID[NET_MAX_MAC_LEN];  ///< Device serial number
    char sDescrip[NET_MAX_INFO_LEN];  ///< Description

```

```
}SDK_NetAlarmCenterMsg;
```

2.14 Related structures of work state SDK_DVR_WORKSTATE

// Alarm state

```
typedef struct SDK_DVR_ALARMSTATE
{
    char iVideoMotion[NET_MAX_MSK_SIZE];    ///< Video motion state, use mask
    to represent channel number, byte0 refers to channel one, and so on. 1 means alarm,
    0 refers to no alarm

    char iVideoBlind[NET_MAX_MSK_SIZE];      ///< Video blind state, use to
    represent channel number, byte0 represent channel one, and so on. 1 refers to
    alarm, 0 refers to no alarm

    char iVideoLoss[NET_MAX_MSK_SIZE];       ///< Video loss state, use to
    represent channel number, byte0 represent channel one, and so on. 1 refers to
    alarm, 0 refers to no alarm

    char iAlarmIn[NET_MAX_MSK_SIZE];        ///< Alarm in state, use to
    represent channel number, byte0 represent channel one, and so on. 1 refers to
    alarm, 0 refers to no alarm

    char iAlarmOut[NET_MAX_MSK_SIZE];       ///< Alarm out state, use to
    represent channel number, byte0 represent channel one, and so on. 1 refers to
    alarm, 0 refers to no alarm
}SDK_DVR_ALARMSTATE;
```

// Channel state

```
typedef struct SDK_DVR_CHANNELSTATE
{
    bool bRecord; ///< Whether normally recorded
    int iBitrate;  ///< Current code rate
}SDK_DVR_CHANNELSTATE;
```

// DVR working state

```
typedef struct SDK_DVR_WORKSTATE
{
```

```
typedef struct SDK_NetAlarmInfo
{
    int iEvent;    //Currently not used
    int iState;    //Each bit means a channel, bit0: the first channel, 0-no alarm, 1-
alarm, and so on
}SDK_NetAlarmInfo;
```

```
Function:      H264_DVR_API long H264_DVR_Init(
               fDisConnect cbDisConnect,
               unsigned long dwUser
               );
```

Parameter:	[in]cbDisConnect	Off line callback function, callback current off line devices, and not callback devices active switch off for calling SDK H264 DVR Logout function, and forbid calling back when set as NULL
------------	------------------	--

```
[in]dwUser           Users data
typedef void (CALL_METHOD *fDisconnect)(
long lLoginID,
char *pchDVRIP,
```

```

    long nDVRPort,
    unsigned long dwUser
);
[out]lLoginID          H264_DVR_Login returned value

[out]pchDVRIP          Device IP

[out]nDVRPort          Device port number

[out]dwUser            Users data

```

Returned value: TRUE refers to be successful, FALSE refers to be disabled

Declaration: Initialized SDK, before every SDK function has been called back

3.1.2 Release SDK resources **H264_DVR_Cleanup**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_Cleanup();

Parameter: None

Returned value: TRUE refers to success, FALSE refers to failure

Declaration: Empty SDK, release occupied resources, before every SDK function has been called back. If return failed, please call [H264_DVR_GetLastError](#) function to get error code, and find reason of error by code.

3.2 SDK local function

3.2.1 Set wait time and try times **H264_DVR_SetConnectTime**

```

Function: H264_DVR_API bool CALL_METHOD H264_DVR_SetConnectTime(
    long nWaitTime,
    long nTryTimes
);

```

Parameter: [in]nWaitTime Wait time (unit in ms, and default value is 5000ms)

[in]nTryTimes Try times (unit in times, and default value is three times)

Returned value: True refers to success.

Declaration: SDK's default wait time is 5000ms, try times is three times.

3.2.2 Bind local IP **H264_DVR_SetLocalBindAddress**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_SetLocalBindAddress(
 char* szIP
);

Parameter: [in]szIP Need bounded IP address

Returned value: TRUE refers to success, FALSE refers to failure

Declaration: Set bounded IP address (In multi-NIC, users can set bounded IP address)

3.2.3 Return to error code of the final operation **H264_DVR_GetLastError**

Function: H264_DVR_API **long** CALL_METHOD H264_DVR_GetLastError();

Parameter: None.

Returned value: Return to error code of the final operation. Details refer to [Error Code Enumeration](#).

Declaration: The returned value is the error code of device returned.

3.3 User registration

3.3.1 User login device **H264_DVR_Login**

Function: H264_DVR_API **long** H264_DVR_Login (

char *sDVRIP,
 unsigned short wDVRPort,
 char *sUserName,
 char *sPassword,
 LPH264_DVR_DEVICEINFO lpDeviceInfo,
 int *error,
 int socketType DEF_PARAM(0)
);

Parameter: [in]sDVRIP Device IP address

[in]wDVRPort	Device port
[in]sUserName	User name
[in]sPassword	User password
[out]lpDeviceInfo	Device information, belong to output parameter
[out]error	Return to login error code
[in]socketTyle	Login style refers to SocketStyle (Default is TCPSOCKET, and could be set NULL)

Returned value: Returned failure 0, return device ID successfully. After login, users can operate device through the value (device handle) referring to related devices. Obtain specific error code by calling [H264_DVR_GetLastError of port](#).

Declaration: Register user to device, when set user as reuse through authenticator system (The default device user, such as admin, and reuse cannot be set), user can register device several times with this account.

3.3.2 User logout device **H264_DVR_Logout**

Function: `H264_DVR_API long CALL_METHOD H264_DVR_Logout(
long lLoginID
);`

Parameter: [in]lLoginID Login handle

Returned value: 1 refers to success, 0 refers to failure.

Declaration:

3.3.3 Active registration **H264_DVR_StartActiveRigister**

Function: `H264_DVR_API bool CALL_METHOD
H264_DVR_StartActiveRigister(
int nPort,
fMessCallBack cbFunc,
unsigned long dwDataUser
);`

Parameter: [in]nPort Monitor port number, $0 \leq nPort \leq 65535$
[out]cbFunc[out] Register online callback function
[in]dwDataUser[in] Callback function parameter
`typedef bool (CALL_METHOD *fMessCallBack)(`

Returned value: true means success, false means failure

Declaration: Need to call parameter configuration port ([H264_DVR_SetDevConfig](#)) to configure network parameter of device, corresponding to configuration command is E_SDK_CONFIG_NET_DAS, and structure set is [SDK_DASSerInfo](#).

3.4.1 Real-time preview H264_DVR_RealPlay

Parameter:	[in]lLoginID	Returned value of H264_DVR_Login
	[in]lpClientInfo	Play handle

Analysis of some error code:

42

establish video sub-connection, device may not be online or in rebooting. Handling method: login after receiving disconnected callback.

(2)H264_DVR_SUB_CONNECT_SEND_ERROR = -11203:

a. LAN access: Sub-connection communication failed, that is the sub-connection was established successfully, but the communication failed, and the device was disconnected after sub-connection was established successfully. Handling method: the return value of (1) will appear when H264_DVR_RealPlay is called again.

b. Active registration access: Main connection communication failed, the device was disconnected, and interior SDK has received disconnected callback. Handling method: Logout after receiving the disconnected callback.

(3)H264_DVR_NOTVALID=-11206: Illegal error, main connection disconnected, the device has been disconnected, and the reboot succeed, but no disconnected callback has been received, and the previous login handle is still in use. How to deal with it: logout after receiving the wire break callback.

Declaration: Call this interface, according to the obtained device information when login, and then you can turn on any effective real-time monitoring in one channel, and obtain raw data by [H264_DVR_SetRealDataCallBack](#) of device callback (note: if playing can be completed by assigning value to hWnd in lpClientInfo, assignment can be completed, without the need for decoding broadcast to the callback data). Successfully return to real-time monitoring ID, in order to monitor the following video channels.

3.4.2 Stop preview **H264_DVR_StopRealPlay**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_StopRealPlay(
long lRealHandle,
void*hWnd DEF_PARAM(0)
);

Parameter: [in]lRealHandle Returned value of H264_DVR_RealPlay
[in]hWnd Used to stop corresponding window decoding play; In

adopting default value NULL, stop every windows decoding play.

Returned value: 1 refers to success, 0 refers to failure

Declaration:

3.4.3 Set data callback **H264_DVR_SetRealDataCallBack**

Function: H264_DVR_API **bool** CALL_METHOD

```
H264_DVR_SetRealDataCallBack(
    long lRealHandle,
    fRealDataCallBack cbRealData,
    long dwUser
);
```

Parameters: [in]lRealHandle Preview play handle

[out]cbRealData Real data callback

[in]dwUser Callback function parameters

```
typedef int(CALL_METHOD *fRealDataCallBack) (
    long lRealHandle,
    long dwDataType,
    unsigned char *pBuffer,
    long lbufsize,
    long dwUser
);
```

[out]lRealHandle Play handle

[out]dwDataType Data type – temporary no need to judge

[out]pBuffer Callback data

[out]lbufsize Callback data length

[out]dwUser User callback parameter

Returned value: TRUE refers to success, FALSE refers to failure. Detailed error code can be obtained from [H264_DVR_GetLastError](#).

Declaration: Set real-time monitoring data callback, provide users data from device.

Note: Besides the general port, there is extended port H264_DVR_SetRealData Callback_V2, which can obtain frame information. Details can be referred to instruction of netsdk.h and utilization of Clientdemo.

3.4.4 Cleanup callback function H264_DVR_DelRealDataCallBack

Function: H264_DVR_API **bool** CALL_METHOD

```
H264_DVR_DelRealDataCallBack(
    long IRealHandle,
    fRealDataCallBack cbRealData,
    long dwUser
);
```

Parameters: [in]IRealHandle Play handle
 [out]cbRealData Real data callback
 [in]dwUser Users parameters
 typedef int(CALL_METHOD *fRealDataCallBack) (
 long IRealHandle,
 long dwDataType,
 unsigned char *pBuffer,
 long lbufsize,
 long dwUser
);

[out]IRealHandle Play handle
 [out]dwDataType Data type – temporary no need to judge
 [out]pBuffer Callback data
 [out]lbufsize Callback data length
 [out]dwUser Users callback parameter

Returned value: TRUE refers to success, FALSE refers to failure. Detailed error code can be obtain from [H264_DVR_GetLastError](#).

Declarartion: Cleanup callback function, and this function should be ahead of H264_DVR_StopRealPlay.

Note: H264_DVR_DelRealDataCallBack_V2 corresponds to H264_DVR_

SetRealDataCallBack_V2.

3.5 Forced I frame H264_DVR_MakeKeyFrame

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_MakeKeyFrame(
 long lLoginID,
 int nChannel,
 int nStream
);

Parameters: [in]lLoginID Login handle
 [in]nChannel Channel number
 [in]nStream Type of code stream – 0 refers to main code stream,
 refers to sub-code stream

Returned value: TRUE refers to success, FALSE refers to failure. Detailed error code
 can be obtained from [H264_DVR_GetLastError](#).

Declaration: Support forced I frame

3.6 Playback and download

Video files finding

3.6.1 Find video by file name H264_DVR_FindFile

Function: H264_DVR_API **long** CALL_METHOD H264_DVR_FindFile(
 long lLoginID,
 H264_DVR_FINDINFO* lpFindInfo,
 H264_DVR_FILE_DATA *lpFileData,
 int lMaxCount,
 int *findcount,
 int waittime DEF_PARAM(5000)
);

Parameters: [in]lLoginID Login handle
 [in]lpFindInfo Find information -- [H264_DVR_FINDINFO](#)
 [out]lpFileData Find result-- [H264_DVR_FILE_DATA](#)

[in]lMaxCount	Maximum video number for searching
[out]findcount	Video number of searching
[in]waittime	Wait time

Returned value: 1 refers to success, 0 refers to failure. Detailed error code can be obtained from [H264_DVR_GetLastError](#).

Declaration: Before playback, you should call the interface to search video record on ahead, when finded video record information according to the period of inputing is larger than the defined buffer size, it only return video record buffer can stored, and further search can be done according to needs.

3.6.2 Search video files by time **H264_DVR_FindFileByTime**

Function: `H264_DVR_API long CALL_METHOD H264_DVR_FindFileByTime(
long lLoginID,
SDK_SearchByTime* lpFindInfo,
SDK_SearchByTimeResult *lpFileData,
int waittime DEF_PARAM(10000)
);`

Parameters: [in]lLoginID Login handle
[in]lpFindInfo Find information-- [SDK_SearchByTime](#)
[out]lpFileData Find result-- [SDK_SearchByTimeResult](#)
[in] waittime Wait time

Returned value: 1 refers to success, 0 refers to failure. Detailed error code can be obtained by [H264_DVR_GetLastError](#).

Declaration: Search video files by time.

Playback video files

3.6.3 Playback video by name

Function: `H264_DVR_API long CALL_METHOD H264_DVR_PlayBackByName(
long lLoginID,
H264_DVR_FILE_DATA *sPlayBackFile,
fDownloadPosCallBack cbDownloadPos,
fRealDataCallBack fDownloadDataCallBack,`


```
long dwDataUser
```

```
);
```

Parameters: [in]lLoginID Login handle

[in]sPlayBackFile Playback file parameters-- [H264 DVR FILE DATA](#)

[out]cbDownLoadPos Schdule callback, User notifies device used that whether data has been sent, and lDownLoadSize=-1 in the callback indicates that the data has been sent. If you want to display real-time progress, you should acquire time from code flow to calculate the network part without analyzing code flow. It is not accurate to calculate the schedule according to the ratio between current data size and total size. It should choose the start time and end time to calculate the schedule.

[out]fDownLoadDataCallBack Playback data callback

[in]dwDataUser Data callback parameter

```
typedef void(CALL_METHOD *fDownLoadPosCallBack) (
```

```
long lPlayHandle,
```

```
long lTotalSize,
```

```
long lDownLoadSize,
```

```
long dwUser
```

```
);
```

[out]lPlayHandle Playback handle

[out]lTotalSize Total length of data

[out]lDownLoadSize Downloaded data length

[out]dwUser User callback parameters

```
typedef int(CALL_METHOD *fRealDataCallBack) (
```

```
long lRealHandle,
```

```
long dwDataType,
```

```
unsigned char *pBuffer,
```

```
long lbufsize,
```

```
long dwUser);
```

[out]IRealHandle	Play handle
[out]dwDataType	Data type - temporarily do not need to judge
[out]pBuffer	Callback data
[out]lbufsize	Callback data length
[out]dwUser	User callback parameters

Return value: Non-zero indicates success, 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: For network playback, it should be noted that after a user logs in to a device, only one video can be played at the same time in each channel, and multiple records in the same channel cannot be played at the same time.

3.6.4 Playback video by time **H264_DVR_PlayBackByTime**

Function: H264_DVR_API [long](#) CALL_METHOD

```
H264_DVR_PlayBackByTime(
    long lLoginID,
    H264_DVR_FINDINFO* lpFindInfo,
    fDownloadPosCallBack cbDownloadPos,
    fRealDataCallBack fDownloadDataCallBack,
    long dwDataUser);
```

Parameter: [in]lLoginID	Login handle
[in]lpFindInfo	Query recording conditions --

[H264_DVR_FINDINFO](#)

[out]cbDownloadPos	Progress callback, the user informs the user device whether the data has been sent, lDownloadSize = -1 in the callback indicates that the data is sent.
--------------------	---

[out]fDownloadDataCallBack	Playback data callback
[in]dwDataUser	Data callback parameters

```
typedef void(CALL_METHOD *fDownloadPosCallBack) (
long lPlayHandle,
long lTotalSize,
long lDownloadSize,
long dwUser);
```

[out]lPlayHandle	Playback handle
[out]lTotalSize	Total data length
[out]lDownloadSize	Downloaded data length
[out]dwUser	User callback parameters

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser);
```

[out]lRealHandle	Play handle
[out]dwDataType	Data type - temporarily do not need to judge
[out]pBuffer	Callback data
[out]lbufsize	Callback data length
[out]dwUser	User callback parameters

Return value: Non-zero indicates success, 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Playback video files by time.

3.6.5 Stop playback video **H264_DVR_StopPlayBack**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_StopPlayBack(
long lPlayHandle);

Parameter: [in]IPlayHandle Playback handle

Return value: true for success, false for failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Enter the playback ID returned by the previous interface. Call this interface to stop the control.

3.6.6 Playback Control H264_DVR_PlayBackControl

Function: H264_DVR_API [bool](#) CALL_METHOD

```
H264_DVR_PlayBackControl(
    long IPlayHandle,
    long IControlCode,
    long ICtrlValue,
    int itype DEF_PARAM(0));
```

Parameter: [in]IPlayHandle Login handle
 [in]IControlCode Control commands, see enum

[SDK_PlayBackAction](#)

 [in]ICtrlValue Control value
 [in]itype Type, see enum

SDK_PLAY_BACK_SETTYPE

Return value: true for success, false for failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Playback control is only valid for smart playback positioning.

Download video files

3.6.7 Downloading Video Files by File Name H264_DVR_GetFileByName

Function: H264_DVR_API [long](#) CALL_METHOD H264_DVR_GetFileByName(
 [long](#) ILoginID,
 [H264_DVR_FILE_DATA](#) *sPlayBackFile,

```

char *sSavedFileName,
fDownloadPosCallBack cbDownloadPos DEF_0_PARAM,
long dwDataUser DEF_0_PARAM,
fRealDataCallBack fDownloadDataCallBack DEF_0_PARAM);

```

Parameter: [in]lLoginID Login handle

[in]sPlayBackFile Downloaded video information-
[H264 DVR FILE DATA](#)

[out]sSavedFileName Saved file path

[out]cbDownloadPos Download progress callback (can be empty, download
progress via [H264 DVR GetDownloadPos](#))

[in]dwDataUser Callback function parameters

[out]fDownloadDataCallBack Data callback

```

typedef void(CALL_METHOD *fDownloadPosCallBack) (

```

```

long lPlayHandle,

```

```

long lTotalSize,

```

```

long lDownloadSize,

```

```

long dwUser);

```

[out]lPlayHandle Playback handle

[out]lTotalSize Total data length

[out]lDownloadSize Downloaded data length

[out]dwUser User callback parameters

```

typedef int(CALL_METHOD *fRealDataCallBack) (

```

```

long lRealHandle,

```

```

long dwDataType,

```

```

unsigned char *pBuffer,

```

```

long lbufsize,

```

```

long dwUser);

```

[out]lRealHandle Play handle

[out]dwDataType Data type - temporarily do not need to

judge

[out]pBuffer	Callback data
[out]lbufsize	Callback data length
[out]dwUser	User callback parameters

Return value: Non-zero indicates success, 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Download the video file according to the file and download the file information through the query. According to the above query records, you can save the video to the specified file. The download progress callback is similar to the playback progress.

3.6.8 Downloading Video Files by Time **H264_DVR_GetFileByTime**

Function: `H264_DVR_API long CALL_METHOD H264_DVR_GetFileByTime(
long lLoginID,
H264_DVR_FINDINFO* lpFindInfo,
char *sSavedFileDIR,
bool bMerge DEF_PARAM(0),
fDownloadPosCallBack cbDownloadPos DEF_0_PARAM,
long dwDataUser DEF_0_PARAM,
fRealDataCallBack fDownloadDataCallBack DEF_0_PARAM);`

Parameter: [in]lLoginID Login handle
 [in]lpFindInfo Video search conditions --
[H264_DVR_FINDINFO](#)
 [in]sSavedFileDIR Video file save path
 [in]bMerge Does the file merge
 [out]cbDownloadPos Download progress callback
 [in]dwDataUser Callback function parameters
 [in]fDownloadDataCallBack Data callback
[typedef void](#)(CALL_METHOD *fDownloadPosCallBack) (

```

long lPlayHandle,
long lTotalSize,
long lDownloadSize,
long dwUser);

```

[out]lPlayHandle	Playback handle
[out]lTotalSize	Total data length
[out]lDownloadSize	Downloaded data length
[out]dwUser	User callback parameters

```

typedef int(CALL_METHOD *fRealDataCallBack) (

```

```

long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser
);

```

[out]lRealHandle	Play handle
[out]dwDataType	Data type - temporarily do not need to
[out]pBuffer	Callback data
[out]lbufsize	Callback data length
[out]dwUser	User callback parameters

judge

Return value: Non-zero indicates success, 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Downloads video files by time and downloads them by querying the file information.

3.6.9 Stop Downloading Video Files **H264_DVR_StopGetFile**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_StopGetFile(
long lFileHandle);

Parameter: [in]IFileHandle Download file handle

Return value: true for success, false for failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: According to need, you can wait for the files to be downloaded and closed, or you can download them to stop downloading.

3.6.10 Download Control H264_DVR_GetFileControl

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_GetFileControl(
 long IPlayHandle,
 long IControlCode,
 bool bDown DEF_PARAM(1));

Parameter: [in]IPlayHandle Login handle
 [in]IControlCode Control commands,see enum [SDK_PlayBack
Action](#)
 [in]bDown Is bit download, the default is 1

Return value: true for success, false for failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Pause and resume control of already opened playback.

3.6.11 Getting Download Progress H264_DVR_GetDownloadPos

Fuction: H264_DVR_API **int** CALL_METHOD H264_DVR_GetDownloadPos(
 long IFileHandle);

Parameter: [in]IFileHandle Download handle

Return value: Greater than or equal to 0 for download progress, less than 0 for failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Obtaining the current position of the downloaded video can be used for an interface that does not need to display download progress in real time,

similar to the function of downloading a callback function. It is not intended to calculate progress through callbacks. It can be called periodically to obtain the current progress.

3.7 PTZ Control **H264_DVR_PTZControl**

Function: H264_DVR_API bool CALL_METHOD H264_DVR_PTZControl(

```

    long lLoginID,

    int nChannelNo,

    long lPTZCommand,

    bool bStop   DEF_PARAM(0),

    long lSpeed DEF_PARAM(4));

```

Parameter:[in]lLoginID	Login handle
[in]nChannelNo	Controlled device channel number
[out]lPTZCommand	Control type :PTZ_ControlType
[in]bStop	Is it stop
[out]lSpeed	Speed, default 4

Return value :Return TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Controls the PTZ, but it must be used with the current channel open.

3.8 Parameter Configuration

3.8.1 Get Device Configuration **H264_DVR_GetDevConfig**

Function: H264_DVR_API long CALL_METHOD H264_DVR_GetDevConfig(

```

    long lLoginID,

    unsigned long dwCommand,

```

```

int nChannelNO,
char * lpOutBuffer,
unsigned long dwOutBufferSize,
unsigned long* lpBytesReturned,
int waittime DEF_PARAM(1000));

```

Parameter:

[in]lLoginID	Login handle
[in]dwCommand	The configuration type is specifically defined in the SDK_CONFIG_TYPE in the data structure definition
[in]nChannelNO	Configure the channel number, -1 means all channels
[out]lpOutBuffer	Store buffers for output parameters. According to different types, output different configuration structures. For details, see the configuration structure in the data structure definition.
[in]dwOutBufferSize	The size of the input buffer, in bytes.
[out]lpBytesReturned	The actual returned buffer size corresponds to the size of the configuration structure (in bytes)
[in]waittime	Waiting time

Return value: Greater than 0 success, less than 0 failure (can be found based on the type of error). The specific error code can be obtained by [H264 DVR GetLastError](#).

Instructions: Different acquisition functions correspond to different structures and command numbers. Specific commands in the configuration information structure [SDK_CONFIG_TYPE](#).

3.8.2 Setting Device Configuration **H264_DVR_SetDevConfig**

Function: `H264_DVR_API long CALL_METHOD H264_DVR_SetDevConfig(
 long lLoginID,
 unsigned long dwCommand,
 int nChannelNO,
 char * lpInBuffer,
 unsigned long dwInBufferSize,
 int waittime DEF_PARAM(1000));`

Parameter:	[in]lLoginID	Login handle
	[in]dwCommand	Configuration type, as defined in the SDK_CONFIG_TYPE in the data structure definition
	[in]nChannelNO	Configure the channel number, -1 means all channels
	[in]lpInBuffer	Store buffers for input parameters, input different configuration structures according to different types. For details, see the configuration structures in the data structure definition.
	[in]dwInBufferSize	The size of the input buffer (in bytes).
	[in]waittime	Waiting time

Return value: Greater than 0 success, less than 0 failure (can be found based on the type of error). The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions : Different setup functions correspond to different structures and command numbers. Specific commands in the configuration information structure [_SDK_CONFIG_TYPE](#)。

3.8.3 Setting Device Configurations Across Network Segments **H264_DVR_Set**

ConfigOverNet

Function: H264_DVR_API [long](#) CALL_METHOD

```
H264_DVR_SetConfigOverNet(
    unsigned long dwCommand,
    int nChannelNO,
    char * lpInBuffer,
    unsigned long dwInBufferSize,
    int waittime DEF_PARAM(1000)
);
```

Parameter: [in]dwCommand Configuration type,

E_SDK_CONFIG_SYSNET

[in]nChannelNO Configure the channel number, 1 temporarily save, other for permanent preservation

[in]lpInBuffer Buffer, structure pointer or address for input parameters—[SDK_CONFIG_NET_COM MON_V3](#)

[in]dwInBufferSize The size of the input buffer (in bytes).

[in]waittime Waiting time

Return value: Equal to 0 indicates success, less than 0 indicates failure. The specific error code can be obtained by [H264 DVR_GetLastError](#).

Instructions: Set the device configuration across network segments. Only network configuration settings are currently supported.

3.9 Log Management H264_DVR_FindDVRLog

Function: H264_DVR_API [bool](#) CALL_METHOD H264_DVR_FindDVRLog(
[long](#) lLoginID,
[SDK_LogSearchCondition](#) *pFindParam,
[SDK_LogList](#) *pRetBuffer,

```

long lBufSize,
int waittime DEF_PARAM(2000));

```

Parameter:	[in]lLoginID	Login handle
	[in]pFindParam	Log query conditions
	[in]pRetBuffer	Return log information
	[in]lBufSize	Log return length
	[in]waittime	Waiting time

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Query log.

3.10 Equipment Control **H264_DVR_ControlDVR**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_ControlDVR(
long lLoginID,
int type,
int waittime DEF_PARAM(2000));

Parameter:	[in]lLoginID	Login handle
	[in]type	0 means restart the device, 1 means clear log, 2 means shutdown, 3 means log recovery, 4 means stop logging, and 5 means temporarily opened audio before handset intercom restore

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Restart/Clear Log/Shutdown/Resume Log/Stop Log/Intercom Off State Restore.

3.11 Upgrading Device Programs

3.11.1 Local Upgrade **H264_DVR_Upgrade**

Function: H264_DVR_API **long** CALL_METHOD H264_DVR_Upgrade(
long lLoginID,
char *sFileName,
int nType DEF_0_PARAM,
fUpgradeCallBack cbUpgrade DEF_0_PARAM,
long dwUser DEF_0_PARAM);

Parameter: [in]lLoginID	Login handle
[in]sFileName	File name to upgrade
[in]nType	File type to upgrade
[in]cbUpgrade	Callback upgrade progress
[in]dwUser	User data

typedef void(CALL_METHOD *fUpgradeCallBack) (
long lLoginID,
long lUpgradechannel,
int nTotalSize,
int nSendSize,
long dwUser);

[out]lLoginID	Login handle
[out]lUpgradechannel	Upgrade channel
[out]nTotalSize	Total data length, description: When nTotalSize = -1, nSendSize:1-99 returns the upgrade progress nTotalSize = 0, nSendSize = H264_DVR_NOENOUGH_ME MORY- H264_DVR_INVALID_WIFI_D RIVE upgrade error code, the other is the progress of sending

[out]nSendSize	Send data length, description: nSendSize = -1 Description Upgrade completed
	nSendSize = -2 Description upgrade error
[out]dwUser	User ddata

Return value: Non-zero indicates success, 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Set up a front-end device network upgrade program. Set up the upgrade of the remote program and return the program upgrade handle.

3.11.2 Obtaining Upgrade Status **H264_DVR_GetUpgradeState**

Function: H264_DVR_API **int** CALL_METHOD H264_DVR_GetUpgradeState(
long lUpgradeHandle);

Parameter: [in]lUpgradeHandle	Upgrade handle, return value: 1 successful, 2 failing to upgrade 3
-------------------------------	--

Return value: 1 means success, 0 means failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: null

3.11.3 Close Upgrade Handle **H264_DVR_CloseUpgradeHandle**

Function: H264_DVR_API **long** CALL_METHOD
H264_DVR_CloseUpgradeHandle(
long lUpgradeHandle);

Parameter: [in]lUpgradeHandle	Upgrade handle
-------------------------------	----------------

Return value: 1 means success, 0 means failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Stop the upgrade.

3.11.4 Close Upgrade H264_DVR_Upgrade_Cloud

Function: H264_DVR_API [long](#) CALL_METHOD H264_DVR_Upgrade_Cloud(
[long](#) lLoginID,
[SDK_CloudUpgradeVersion](#) *sUpgradeVer,
[int](#) nType DEF_0_PARAM,
[fUpgradeCallBack](#) cbUpgrade DEF_0_PARAM,
[long](#) dwUser DEF_0_PARAM);

Parameter:	[in]lLoginID	Login handle
	[in]sUpgradeVer	Upgraded file information
	[in]nType	Type
	[out]cbUpgrade	Callback upgrade information
	[in]dwUser	User parameters

```
typedef void(CALL_METHOD *fUpgradeCallBack) (
long lLoginID,
long lUpgradechannel,
int nTotalSize,
int nSendSize,
long dwUser);
```

[out]lLoginID	Login handle
[out]lUpgradechannel	Upgrade channel
[out]nTotalSize	Total data length, description: nTotalSize = -1时, nSendSize:1-99 return upgrade progress nTotalSize =0时,nSendSize = H264_DVR_NOENOUGH_ME

MORY-

H264_DVR_INVALID_WIFI_D

RIVE Upgrade the error code, the other is the sending progress.

When the cloud upgrade

increases nTotalSize=-2,

nSendSize:0 - 100=download

progress, no progress is sent

[out]nSendSize

Send data length, description:

nSendSize = -1 Description Upgrade

completed

nSendSize = -2 Description upgrade

error

[out]dwUser

User data

Return value: Equal to 0 indicates success, less than 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Perform an online upgrade.

3.11.5 Stopping Cloud Upgrade **H264_DVR_StopUpgrade_Cloud**

Function: H264_DVR_API [long](#) CALL_METHOD

H264_DVR_StopUpgrade_Cloud(

[long](#) lHandle);

Parameter: [in]lHandle

Cloud upgrade handle

Return value: Equal to 0 indicates success, less than 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Stop the cloud upgrade.

3.12 Voice Intercom

3.12.1 Start Intercom **H264_DVR_StartVoiceCom_MR**

Function: H264_DVR_API **long** CALL_METHOD

H264_DVR_StartVoiceCom_MR(

long ILoginID,
 pfAudioDataCallBack pVcb,
 long dwDataUser);

Parameter:	[in]ILoginID	Login handle
	[out]pVcb	Speech callback data received from the device
	[in]dwDataUser	Callback function parameters

```

typedef void (CALL_METHOD *pfAudioDataCallBack)(
long IVoiceHandle,
char *pDataBuf,
long dwBufSize,
char byAudioFlag,
long dwUser);

```

	[out]IVoiceHandle	Intercom handle
	[out]pDataBuf	Callback data
	[out]dwBufSize	Data size
	[out]byAudioFlag	Flag
	[out]dwUser	User parameters

Return value: Non-zero indicates success, 0 indicates failure. The specific error code can be obtained by [H264_DVR_GetLastError](#).

Instructions: Start the intercom process.

3.12.2 Send Talkback Data **H264_DVR_VoiceComSendData**

Function: H264_DVR_API **bool** CALL_METHOD

H264_DVR_VoiceComSendData(

long IVoiceHandle,
 char *pSendBuf,

`long lBufSize);`

Parameter:	[in]lVoiceHandle	Intercom handle
	[in]pSendBuf	Transmitted voice data
	[in]lBufSize	The size of the data sent

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Send talkback data.

3.12.3 Stop Intercom **H264_DVR_StopVoiceCom**

Function: H264_DVR_API `bool` CALL_METHOD H264_DVR_StopVoiceCom(
`long` lVoiceHandle);

Parameter:	[in]lVoiceHandle	Intercom handle
------------	------------------	-----------------

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Stop the intercom.

3.12.4 Setting Intercom Audio Coding Mode **H264_DVR_SetTalkMode**

Function: H264_DVR_API `bool` CALL_METHOD H264_DVR_SetTalkMode(
`long` lLoginID,
`SDK_AudioInFormatConfig*` pTalkMode);

Parameter:	[in]lLoginID	Login handle
	[in]pTalkMode	Talkback mode structure

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Set intercom audio encoding mode, users can not set, the default is G711A encoding.

3.13 Recording Mode Settings

3.13.1 Manual Recording H264_DVR_StartDVRRecord

Function: H264_DVR_API [bool](#) CALL_METHOD

```
H264_DVR_StartDVRRecord(
    long lLoginID,
    int nChannelNo,
    long lRecordType);
```

Parameter: [in]lLoginID	Login handle
[in]nChannelNo	Channel number, -1 for full channel, 0-n for single channel
[in]lRecordType	Video type See: SDK_RecordModeTypes

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: This interface is added to facilitate manual recording. You can also set recording to manual mode through the system configuration setting interface ([H264_DVR_SetDevConfig](#)).

3.13.2 Close Recording H264_DVR_StopDVRRecord

Function: H264_DVR_API [bool](#) CALL_METHOD

```
H264_DVR_StopDVRRecord(
    long lLoginID,
    int nChannelNo);
```

Parameter: [in]lLoginID	Login handle
[in]nChannelNo	Channel number, -1 for full channel, 0-n for single channel

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: This interface is added for the convenience of manually closing the video, and it can also be set to the shutdown mode through the system configuration setting interface ([H264_DVR_SetDevConfig](#)).

3.14 Setting System Time H264_DVR_SetSystemDateTime

Function: H264_DVR_API **bool** CALL_METHOD

```
H264_DVR_SetSystemDateTime(
    long lLoginID,
    SDK_SYSTEM_TIME *pSysTime,
    bool nType DEF_0_PARAM);
```

Parameter: [in]lLoginID Login handle
 [in]pSysTime System time ,See:
[SDK_SYSTEM_TIME](#)
 [in]nType System time type (true - new system time)

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Set the device system time.

3.15 Armed Alarm

3.15.1 Alarm Status Acquisition H264_DVR_SetDVRMessCallBack

Function: H264_DVR_API **bool** CALL_METHOD

```
H264_DVR_SetDVRMessCallBack(
    fMessCallBack cbAlarmcallback,
    unsigned long lUser);
```

Parameter: [out]cbAlarmcallback The message callback function can call back the status of the device. For example, the alarm status can be obtained

through this callback; when set to 0, the callback is prohibited.

[in]lUser User data

```
typedef bool (CALL_METHOD *fMessCallBack)(
```

```
long lLoginID,
```

```
char *pBuf,
```

```
unsigned long dwBufLen,
```

```
long dwUser,
```

```
int nType);
```

[out]lLoginID Login handle

[out]pBuf Callback data--

SDK_AlarmInfo

[out]dwBufLen Callback data length

[out]dwUser Callback function parameters

[out]nType Type-- ALARM_TYPE

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Set the device message callback function, used to get the current status information of the device. Regardless of the calling sequence, the SDK does not callback by default. This callback function must first call the alarm callback upload channel interface [H264_DVR_SetupAlarmChan](#) to be effective, and it needs to explain the current defined alarm. Is the callback device's current alarm message per second.

3.15.2 Setting Alarm Callback Upload Channe H264_DVR_SetupAlarmChan

Function: H264_DVR_API long CALL_METHOD

```
H264_DVR_SetupAlarmChan(
```

```
long lLoginID);
```

Parameter: [in]lLoginID Login handle

Return value: 1 means success, 0 means failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Begin to subscribe to a device message, used to set whether the device message callback, the message obtained from the [H264_DVR_SetDVRMessCallBack](#) set callback.

3.15.3 Turning Off the Alarm Callback Upload Path H264_DVR_CloseAlarmChan

Function: H264_DVR_API **bool** CALL_METHOD

H264_DVR_CloseAlarmChan(
 long lLoginID);

Parameter: [in]lLoginID Login handle

Return value : 1 indicates success, less than or equal to 0 indicates failure. The specific error code is obtained by [H264_DVR_GetLastError](#) .

Instructions: Stop listening for messages on a device.

3.16 Monitor alarm

3.16.1 Starting Alarm Center Monitoring H264_DVR_StartAlarmCenterListen

Function: H264_DVR_API **bool** CALL_METHOD

H264_DVR_StartAlarmCenterListen(
 int nPort,
 fMessCallBack cbAlarmCenter,
 unsigned long dwDataUser);

Parameter: [in]nPort Listening port number

[out]cbAlarmCenter Data callback

[in]dwDataUser Callback parameters

typedef bool (CALL_METHOD *fMessCallBack)(

long lLoginID,

```

char *pBuf,
unsigned long dwBufLen,
long dwUser,
int nType
);

```

[out]lLoginID	Login handle
[out]pBuf	Callback data -
SDK_AlarmInfo	
[out]dwBufLen	Callback data length
[out]dwUser	Callback function parameters
[out]nType	Type-- ALARM_TYPE

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Before opening the alarm center, you need to call the interface ([H264_DVR_SetDevConfig](#)) to set the computer address for monitoring the device alarm. Corresponding configuration command E_SDK_CONFIG_ALARM_CENTER, structure [SDK_NetAlarmServer ConfigAll](#).

3.16.2 Turning off Alarm Center Monitoring H264_DVR_StopAlarmCenter Listen

Function: H264_DVR_API **bool** CALL_METHOD

H264_DVR_StopAlarmCenterListen();

Parameter: null

Return value: null

Instructions: Turn off alarm center monitoring.

3.17 Get device operating status information H264_DVR_GetDVR WorkState

Function: H264_DVR_API bool CALL_METHOD

```
H264_DVR_GetDVRWorkState(
    long ILoginID,
    SDK_DVR_WORKSTATE *pWorkState);
```

Parameter: [in]ILoginID Login handle
 [out]pWorkState The working status of the equipment-[SDK_DVR_WORKSTATE](#)

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Get device working status information.

3.18 Network alarm H264_DVR_SendNetAlarmMsg

Function: H264_DVR_API bool CALL_METHOD

```
H264_DVR_SendNetAlarmMsg(
    long ILoginID,
    SDK_NetAlarmInfo *pAlarmInfo);
```

Parameter: [in]ILoginID Login handle
 [in]pAlarmInfo Network alarm parameters -- [SDK_NetAlarmInfo](#)

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Send network alarm information.

3.19 Disk Management H264_DVR_StorageManage

```
Function: H264_DVR_API int CALL_METHOD H264_DVR_StorageManage(
    long ILoginID,
    SDK_StorageDeviceControl *pStorageCtl);
```

Parameter: [in]ILoginID Login handle
 [in]pStorageCtl Operating parameters- [SDK_StorageDeviceControl](#)

Return value : 1 indicates success, less than or equal to 0 indicates failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Disk management.

3.20 Device capture **H264_DVR_CatchPic**

Function: H264_DVR_API bool CALL_METHOD H264_DVR_CatchPic(
 long lLoginID,
 int nChannel,
 char *sFileName,
 int nType DEF_0_PARAM);

Parameter:	[in] lLoginID	Login handle
	[in]nChannel	Controlled device channel
	number	
	[in] sFileName	The name of the picture to be saved, full
	path	

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions :

1. The device configuration is required to have a screenshot configuration option for this interface to be effective;
2. If it satisfies 1, the default resolution is D1. If you need to capture the same resolution as the video resolution, you need to modify the resolution in the encoding settings. If the encoding setting does not have a resolution option, you need custom support. The program of this item.

3.21 Transparent serial port

3.21.1 Creating a Transparent Serial Port Channel **H264_DVR_OpenTransComChannel**

Function: H264_DVR_API bool CALL_METHOD
 H264_DVR_OpenTransComChannel(

```

long lLoginID,
TransComChannel *TransInfo,
fTransComCallBack cbTransCom,
unsigned long lUser);

```

Parameter:	[in]lLoginID	Login handle
	[in]TransInfo	Serial port parameters, refer to TransComChannel
	[in]cbTransCom	Callback

```

typedef void (CALL_METHOD *fTransComCallBack) (
long lLoginID,
long lTransComType,
char *pBuffer,
unsigned long dwBufSize,
unsigned long dwUser
);

```

	[out]lLoginID	Login handle
	[out]lTransComType	Serial port type, see SERIAL_TYPE
	[out]pBuffer	Callback data buffer
	[out]dwBufSize	Callback data length
	[out]dwUser	User data

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions : Create a transparent serial channel.

3.21.2 Writing Data to the Device Through the Serial Port **H264_DVR_SerialWrite**

Function: H264_DVR_API [bool](#) CALL_METHOD H264_DVR_SerialWrite(

```
long ILoginID,  
SERIAL_TYPE nType,  
char *pBuffer,  
int nBufLen);
```

Parameter:	[in] ILoginID	Login handle
	[in] nType	See SERIAL_TYPE for details
	[int] pBuffer	Data buffer
	[in] nBufLen	The length of the data buffer

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Write data to the device through the serial port.

3.21.3 Reading Data from the Device Through the Serial Port H264_DVR_Serial Read

Function: H264_DVR_API [bool](#) CALL_METHOD H264_DVR_SerialRead(

long ILoginID,

SERIAL_TYPE nType,

char *pBuffer,

int nBufLen,

int *pReadLen);

Parameter:	[in] ILoginID	Login handle
	[in] nType	See SERIAL_TYPE for details
	[out] pBuffer	Read data after the buffer
	[in] nBufLen	Data length in the buffer
	[out] pReadLen	Actual received length

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Reading Data from the Device Through the Serial Port.

3.21.4 Disabling Transparent Serial Port Channels H264_DVR_CloseTransComChannel

Function: H264_DVR_API **bool** CALL_METHOD

```
H264_DVR_CloseTransComChannel(
    long lLoginID,
    SERIAL_TYPE nType);
```

Parameter: [in]lLoginID Login handle
 [in]nType See [SERIAL_TYPE](#) for details

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Close the transparent serial port channel.

3.22 Client recording

3.22.1 Starting Local Recording H264_DVR_StartLocalRecord

Function: H264_DVR_API **bool** CALL_METHOD

```
H264_DVR_StartLocalRecord(
    long lRealHandle,
    char* szSaveFileName,
    long type=0);
```

Parameter: [in]lRealHandle Play handle (H264_DVR_RealPlay return value)
 [in]szSaveFileName Save route
 [in]type Recording type: (0: file name suffix is .h264; 2: file name suffix .avi), the default is 0;

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Function: H264 DVR API **bool** CALL METHOD

Parameter:	[in]IRealHandle	Play handle (H264_DVR_RealPlay return value)
------------	-----------------	--

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Stop the PC video.

3.23 Client audio

```
Function:  H264_DVR_API bool CALL_METHOD H264_DVR_OpenSound(
long lHandle);
```

Parameter: [in]IHandle

H264_DVR_RealPlay or
H264_DVR_StartLocalPlay or
H264_DVR_PlayBackByName
or
H264_DVR_PlayBackByTimeE
x return value

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Turn on the audio of the video channel.

```
Function: H264_DVR_API bool CALL_METHOD H264_DVR_CloseSound(
    long lHandle);
```

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instruction: Turn off the audio of the video channel.

3.24.1 Get Playback Position (Percentage) **H264_DVR_GetPlayPos**

Parameter:	[in] IPlayHandle	H264_DVR_StartLocalPlay	or
		H264_DVR_PlayBackByName	
		or	
		H264_DVR_PlayBackByTimeE	
		x	return value

Instructions: Get the playback progress of playback or local playback. The interface handle is valid when playback.

```
Function:  H264_DVR_API bool CALL_METHOD H264_DVR_SetPlayPos(
           long lPlayHandle,
           float fRelativPos);
```

78

H264_DVR_PlayBackByName

or

H264_DVR_PlayBackByTimeE

x return value

[in]fRelativPos

Play percentage

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Sets the playback progress of playback or local playback. This interface is valid only when transmitting the window handle during playback.

3.25 Setting Info Frame Callbacks H264_DVR_SetInfoFrameCall

Back

Function: H264_DVR_API [bool](#) CALL_METHOD

H264_DVR_SetInfoFrameCallBack(

[long](#) lPlayHandle,

[InfoFramCallBack](#) callback,

[long](#) user);

Parameter: [in] lPlayHandle

H264_DVR_RealPlay or

H264_DVR_StartLocalPlay or

H264_DVR_PlayBackByName

or

H264_DVR_PlayBackByTimeE

x return value

[in]callback

Callback

[in] user

User-defined data

[typedef void](#) (CALL_METHOD *InfoFramCallBack)(

[long](#) lPlayHand,

[long](#) nType,


```

        LPCSTR pBuf,
        long nSize,
        long nUser);

[out]IPlayHand          Play handle
[out]nType               Data type - temporarily do not need to
judge
[out]pBuf               Callback data
[out]nSize               Callback data length
[out]nUser               User callback parameters

```

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Set information frame callbacks.

3.26 Client video color

3.26.1 Get Play Color Information **H264_DVR_LocalGetColor**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_LocalGetColor(
 long lHandle,
 DWORD nRegionNum,
 LONG *pBrightness,
 LONG *pContrast,
 LONG *pSaturation,
 LONG *pHue);

Parameter: [in] lHandle H264_DVR_RealPlay or
 H264_DVR_StartLocalPlay or
 H264_DVR_PlayBackByName
 or
 H264_DVR_PlayBackByTimeE
 x return value

[in]nRegionNum	Area (temporary: can be set to 0)
[out]pBrightness	Brightness
[out]pContrast	Contrast
[out]pSaturation	Saturation
[out]pHue	Chroma

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Get playback video color information.

3.26.2 Setting Play Color Information **H264_DVR_LocalSetColor**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_LocalSetColor(
long lHandle,
DWORD nRegionNum,
LONG nBrightness,
LONG nContrast,
LONG nSaturation,
LONG nHue);

Parameter: [in] lHandle H264_DVR_RealPlay or
 H264_DVR_StartLocalPlay or
 H264_DVR_PlayBackByName
 or
 H264_DVR_PlayBackByTimeE
 x return value

[in]nRegionNum	Area (temporary: can be set to 0)
[in]nBrightness	Brightness
[in]nContrast	Contrast
[in]nSaturation	Saturation
[in]nHue	Chroma

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Set playback video color information.

3.27 Playing Client Local Files

3.27.1 Playing Local Files **H264_DVR_StartLocalPlay**

Function: H264_DVR_API **long** CALL_METHOD H264_DVR_StartLocalPlay(
 char*pFileName,
 void* hWnd,
 fPlayDrawCallBack drawCallBack=0,
 long user=0);

Parameter: [in]pFileName	Play file name
[in]hWnd	Play window handle
[in]drawCallBack	Callback function (can not be set to NULL)
[in]user	User-defined data

typedef void (CALL_METHOD * fPlayDrawCallBack)(
 long lPlayHand,
 HDC hDc,
 long nUser);

[out]lPlayHand	Play handle
[out]hDc	
[out]nUser	Callback parameters

Return value: Failed to return 0, successfully returned to the playback ID (local playback handle), will be used as a parameter of the relevant function.

Instructions: Play the local .h264 video file.

3.27.2 Turning off Local Playback **H264_DVR_StopLocalPlay**

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_StopLocalPlay(

```
long lPlayHandle
```

```
);
```

Parameter: [in] lPlayHandle H264_DVR_StartLocalPlay return value

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Turn off local playback.

3.27.3 Local File Playback Callback H264_DVR_SetFileEndCallBack

Function: H264_DVR_API bool CALL_METHOD

```
H264_DVR_SetFileEndCallBack(
```

```
    long lPlayHandle,
```

```
    fLocalPlayFileCallBack callBack,
```

```
    long user);
```

Parameter: [in]lPlayHandle H264_DVR_StartLocalPlay return value

[in]callBack End callback

[in]user User-defined data

```
typedef void (CALL_METHOD * fLocalPlayFileCallBack)(
```

```
    long lPlayHand,
```

```
    long nUser);
```

[out]lPlayHand Play handle

[out]nUser Callback parameters

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: The end of the local file playback callback.

3.27.4 Local File Playback Control H264_DVR_LocalPlayCtrl

Function: H264_DVR_API bool CALL_METHOD H264_DVR_LocalPlayCtrl(

```
    long lPlayHandle,
```

```
    SDK_LoalPlayAction action,
```

Parameter: [in]IPlayHandle	H264_DVR_StartLocalPlay return value
[in]action	See the reference: SDK_LoalPlayAction
[in]ICtrlValue	Quick release (1, 2, 3, 4 levels), and slow playback (1, 2, 3, 4 levels)

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Play control (play, stop, resume, fast send, slow release).

3.28 Disconnection of Detector Connection H264 DVR SetSub

DisconnectCallback

Function: H264_DVR_API long CALL_METHOD

```
H264_DVR_SetSubDisconnectCallBack(  
    fSubDisConnectCallBack callBack,  
    DWORD userData);
```

Parameter: [in]callBack Subconnection disconnect callback (see [SubConnType](#))

[in]userData	User data
typedef void (CALL_METHOD *fSubDisConnectCallBack)(
long lLoginID,	
SubConnType type,	
long nChannel,	
long dwUser);	
[out]lLoginID	Login handle
[out]type	Type of data --
<u>SubConnType</u> [out]nChannel	Channel
number	
[out]dwUser	User parameter

Return value: 1 means success, 0 means failure.

Instructions: The detector connection is abnormally disconnected.

3.29 Set keep-alive time and break detection time H264_DVR_

SetKeepLifeTime

Function: H264_DVR_API **long** CALL_METHOD

```
H264_DVR_SetKeepLifeTime(
    long lLoginID,
    unsigned int perKeeplifeTime,
    unsigned int detectDisconTime
);
```

Parameter:	[in]lLoginID	Login handle
	[in]perKeeplifeTime	Heartbeat packet interval (in seconds)
	[in]detectDisconTime	Device disconnection time (in seconds)

Return value: 1 means success, 0 means failure.

Instructions: Set keep-alive time, perKeeplifeTime: Default 1 second,
detectDisconTime (break detection time): Default 60 seconds.

3.30 Searching for Local Area Network Settings H264_DVR_Search

Device

Function: H264_DVR_API **bool** CALL_METHOD H264_DVR_SearchDevice(
char* szBuf,
int nBufLen,
int* pRetLen,
int nSearchTime);

Parameter:	[out]szBuf	Receive buffer
	[in]nBufLen	Receive buffer size
		sizeof(SDK_CONFIG_NET_CO MMON_V2)*n
	[in]pRetLen	Return size
	[in]nSearchTime	Waiting time

Return value: Returns TRUE on success and FALSE on failure. The specific error code is obtained by [H264_DVR_GetLastError](#).

Instructions: Search for devices in the LAN.

4 Error code enumeration

Error code name	Error return value	Instructions
H264_DVR_NOERROR	0	No error
H264_DVR_SUCCESS	1	Return success
H264_DVR_SDK_NOTVALID	-10000	Illegal request
H264_DVR_NO_INIT	-10001	SDK is not initialized
H264_DVR_ILLEGAL_PARAM	-10002	User parameters are invalid
H264_DVR_INVALID_HANDLE	-10003	Invalid handle
H264_DVR_SDK_UNINIT_ERROR	-10004	SDK cleanup error
H264_DVR_SDK_TIMEOUT	-10005	Waiting for timeout
H264_DVR_SDK_MEMORY_ERR OR	-10006	Memory error, creating memory failed
H264_DVR_SDK_NET_ERROR	-10007	Network Error
H264_DVR_SDK_OPEN_FILE_ER ROR	-10008	fail to open the file
H264_DVR_SDK_UNKNOWNERR OR	-10009	unknown mistake
H264_DVR_DEV_VER_NOMATCH	-11000	Incorrect data received, possible version mismatch
H264_DVR_SDK_NOTSUPPORT	-11001	Version does not support

H264_DVR_ANAS_EXIST	-11130	NAS address already exists
H264_DVR_ANAS_ALIVE	-11131	Path is used and cannot be operated
H264_DVR_ANAS_FULL	-11132	NAS has reached the maximum supported
H264_DVR_OPEN_CHANNEL_ERROR	-11200	Failed to open the channel, it may be detected that the device is no longer online
H264_DVR_CLOSE_CHANNEL_ERROR	-11201	Failed to close the channel
H264_DVR_SUB_CONNECT_ERROR	-11202	Failed to establish media subconnection, network error or device may not be online
H264_DVR_SUB_CONNECT_SEND_ERROR	-11203	Media sub-connection communication failed, may detect that the device is no longer online
H264_DVR_NATCONNET_REACHED_MAX	-11204	Nat video link reaches maximum, no new Nat video link allowed
H264_DVR_NOTSUPPORT	-11205	Version does not support
H264_DVR_NOTVALID	-11206	The request was illegal and the primary connection may have been disconnected
H264_DVR_TCPCONNET_REACHED_MAX	-11207	Tcp video link reaches maximum, new Tcp video link is not allowed
H264_DVR_OPENEDPREVIEW	-11208	The channel has opened the preview (opening and closing of the channel needs one to one

		correspondence, and opening several times requires closing several times; the inconsistency will open the prompt for this error; prevent the client from developing the logically unreasonable design to increase the error value)
H264_DVR_NOPOWER	-11300	No permission
H264_DVR_PASSWORD_NOT_VALID	-11301	Account password is wrong
H264_DVR_LOGIN_USER_NOEXIST	-11302	User does not exist
H264_DVR_USER_LOCKED	-11303	The user is locked
H264_DVR_USER_IN_BLACKLIST	-11304	This user is not allowed to access (in the blacklist)
H264_DVR_USER_HAS_USED	-11305	This user is logged in
H264_DVR_USER_NOT_LOGIN	-11306	The user is not logged in
H264_DVR_CONNECT_DEVICE_ERROR	-11307	Possible device does not exist
H264_DVR_ACCOUNT_INPUT_NOT_VALID	-11308	User management input is illegal
H264_DVR_ACCOUNT_OVERLAP	-11309	Duplicate index
H264_DVR_ACCOUNT_OBJECT_NONE	-11310	No object exists for query
H264_DVR_ACCOUNT_OBJECT_NOT_VALID	-11311	There is no object
H264_DVR_ACCOUNT_OBJECT_INUSE	-11312	The object is in use

N_USE		
H264_DVR_ACCOUNT_SUBSET_OVERLAP	-11313	Subset over range (eg, group permissions exceed permission tables, user permissions exceed group permissions, etc.)
H264_DVR_ACCOUNT_PWD_NOT_VALID	-11314	Incorrect password
H264_DVR_ACCOUNT_PWD_NOT_MATCH	-11315	Passwords do not match
H264_DVR_ACCOUNT_RESERVED	-11316	Reserved account
H264_DVR_ACCOUNT_SYS_MAINTAIN	-11317	System maintenance cannot be logged in
H264_DVR_EE_DVR_PASSWORD_NOT_VALID2	-11318	Account password is wrong
H264_DVR_OPT_RESTART	-11400	Need to restart the application after saving the configuration
H264_DVR_OPT_REBOOT	-11401	Need to restart the system
H264_DVR_OPT_FILE_ERROR	-11402	Error writing file
H264_DVR_OPT_CAPS_ERROR	-11403	Configuration features are not supported
H264_DVR_OPT_VALIDATE_ERROR	-11404	Configuration check failed
H264_DVR_OPT_CONFIG_NOT_EXIST	-11405	The configuration requested or set does not exist
H264_DVR_CTRL_PAUSE_ERROR	-11500	Pause failed
H264_DVR_SDK_NOTFOUND	-11501	Failed to find the

		corresponding file
H264_DVR_CFG_NOT_ENABLE	-11502	Configuration is not enabled
H264_DVR_DECORD_FAIL	-11503	Failed to decode
H264_DVR_SOCKET_ERROR	-11600	Failed to create socket
H264_DVR_SOCKET_CONNECT	-11601	Failed to connect socket
H264_DVR_SOCKET_DOMAIN	-11602	Domain name resolution failed
H264_DVR_SOCKET_SEND	-11603	Failed to send data
H264_DVR_ARSP_NO_DEVICE	-11604	Failed to get device information, device should not be online
H264_DVR_ARSP_BUSING	-11605	ARSP service is busy
H264_DVR_ARSP_BUSING_SELECT	-11606	ARSP service is busy and select fails
H264_DVR_ARSP_BUSING_RECEIVE	-11607	ARSP service is busy and receive fails
H264_DVR_CONNECTSERVER_ERROR	-11608	Connection failure
H264_DVR_CONNECT_AGNET	-11609	Proxy
H264_DVR_CONNECT_NAT	-11610	Penetrate
H264_DVR_CONNECT_FAILED	-11611	Connection failed
H264_DVR_CONNECT_FULL	-11612	The server connection is full
H264_DVR_CLOUD_LOGIN_ERR	-11613	Cloud login specific error code, indicating: When login interface error=-11613, get error code through

		H264_DVR_DEVICEINFO member sCloudErrCode
H264_DVR_NO_CONNECT_FRONT	-11614	The front-end device is not connected or the resolution of the connected front-end device is unknown
H264_DVR_LOGIN_FULL	-11615	The login handle has reached its maximum value and can no longer log in
H264_DVR_ARSP_USER_NOEXIST	-11619	User does not exist
H264_DVR_ARSP_PASSWORD_ERROR	-11620	Account password is wrong
H264_DVR_ARSP_QUERY_ERROR	-11621	Query failed
H264_DVR_PIRATESOFTWARE	-11700	Equipment piracy
H264_DVR_AUTH_TIMEOUT	-11800	Authentication timeout
H264_DVR_AUTH_FILE_FAILED	-11801	Authentication file failed
H264_DVR_GAIN_LIST_TIMEOUT	-11802	Get server list timeout
H264_DVR_AUTH_CODE_ERR	-11803	Authentication code error
H264_DVR_NOENOUGH_MEMORY	-11804	Not enough storage
H264_DVR_INVALID_FORMAT	-11805	The upgrade file format is incorrect
H264_DVR_UPDATE_PART_FAIL	-11806	A partition failed to upgrade
H264_DVR_INVALID_HARDWARE	-11807	Hardware model does not

E		match
H264_DVR_INVALID_VENDOR	-11808	Customer information does not match
H264_DVR_INVALID_COMPATIBLE	-11809	The upgrade program's compatible version number is smaller than the device's existing one and does not allow the device to be upgraded back to the old program
H264_DVR_INVALID_VERSION	-11810	Illegal version
H264_DVR_INVALID_WIFI_DRIVER	-11811	The Wi-Fi driver that the Wi-Fi driver and device is currently using in the upgrade program does not match
H264_DVR_INVALID_CUR_FLASH	-11812	Upgrade program does not support Flash used by device
H264_DVR_NAT_INIT_TIMEOUT	-12000	Cloud login initialization timeout is used to distinguish general timeout conditions

5 Sample function implementation

Please see the ClientDemo program and DEMO instructions.doc.