Device Network SDK Programming Manual

(For Android)

V5.2

Preface

Thank you for purchasing our product. If there is any question or request, please feel free to contact us.

This manual may contain several technically incorrect places or printing errors, and the content is subject to change without notice. The updates will be added into the new version of this manual, and we will readily improve or update the product or procedure described in the manual.

Content

1 SDK Overview

The device network SDK is developed based on private network communication protocol, and it is designed for the remote connection and configuration of embedded DVR, NVR, video server, encoder, IPC, IP dome, security control panel and the other IP devices.

The functions supported by the SDK

Live view, capturing image, PTZ control, video file searching and playback.

Model (not limited to)

- Encoding and Decoding Device

NVR: DS-9600, DS-8600, DS-9500, DS-7700, DS-7600.
HDVR: DS-9000, DS-8000-ST, DS-7600.
DVR: DS-9100, DS-8100, DS-8000-S, DS-8800, DS-7800, DS-7300, DS-7200, DS-7100.
Encoder: DS-6401HFH, DS-6600, DS-6500(-JX), DS-6100.
Note: It includes -ST, -SH, -SE, -SN, -RT, -RH, -XT models.

- Network Camera and Network Dome

Network Camera: Standard definition, high definition, Infrared, thermal, fisheye, etc. E.g., DS-2CD7xx, DS-2CD71xx, DS-2CD72xx, DS-2CD8xx, DS-2CD81xx, DS-2CD82xx, DS-2CD84xx, DS-2CD83xx, DS-2CD20xx, DS-2CD21xx, DS-2CD22xx, DS-2CD23xx, DS-2CD26xx, DS-2CD30xx, DS-2CD31xx, DS-2CD32xx, DS-2CD33xx, DS-2CD40xx, DS-2CD41xx, DS-2CD42xx, DS-2CD62xx, DS-2CD63xx, etc.
Network Dome: Standard definition, high definition, Infrared, etc. E.g., DS-2DE71xx, DS-2DM72xx, DS-2DF72xx, DS-2DF1-7xx, DS-2DF1-6xx, DS-2DE51xx, DS-2DM52xx, DS-2DF52xx, DS-2DF1-5xx, DS-2DF1-4xx, DS-2DM1-7xx, DS-2DM1-6xx, DS-2DM1-5xx, etc.
Zoom Network Camera: DS-2DZ216MF, DS-2DZ2116, DS-2ZCN2006, DS-2ZCN2007, DS-2ZMN2007, DS-2ZMN2006, etc.
Traffic Cmaera (Capture Camera): (i)DS-2CD93xx, (i)DS-2CD92xx, (i)DS-2CD91xx, DS-2CD9xx, etc.

Running Environment

Android V4.0 or above

2 SDK Version Update

Version 5.2.5.2 (build20160715)

- Using JNA mode to call C++ DLL interface. The corresponding API and structures are based on HCNetSDKByJNA.
- Updating the alarm arming API to JNA API (The original JNI API is still supported):
  NET_DVR_SetDVRMessageCallBack_V30,
  NET_DVR_SetupAlarmChan_V41,
  NET_DVR_CloseAlarmChan_V30.
- Newly added alarm type:
  Alarm information including motion detection, video loss, video tempering, IO sensors, and so on: COMM_ALARM_V40 (the size of alarm data is variable);
  Behavior analysis alarm including intrution detection, line crossing detection, and so on: COMM_ALARM_RULE;
  License plate recognition, capture and uploading: COMM_UPLOAD_PLATE_RESULT, COMM_ITS_PLATE_RESULT;
- Newly added multi-streams encoding parameter configuration (corresponding API: NET_DVR_GetDeviceConfig, NET_DVR_SetDeviceConfig):
  NET_DVR_GET_MULTI_STREAM_COMPRESSIONCFG, NET_DVR_SET_MULTI_STREAM_COMPRESSIONCFG;

Version 5.1.3.2 (build20150605)

- Newly added dynamically IP and port resolution API:
  NET_DVR_GetAddrInfoByServer.
- Newly added activate device API:
  NET_DVR_ActivateDevice.
- Newly added parameter configuration (corresponding API: NET_DVR_GetDVRConfig):
  NET_DVR_GET_DIGITAL_CHANNEL_STATE, NET_DVR_GET_PRESET_NAME.
- Newly added instantly update record index API:
  NET_DVR_UpdateRecordIndex.
- Newly added search record by event API:
  NET_DVR_FindFileByEvent, NET_DVR_FindNextEvent.
- Newly added expand transparent channel API:
  NET_DVR_SerialStart_V40.
- Newly added remote control panel API:
  NET_DVR_ClickKey.

3 Function Definition

3.1 SDK Initialization

3.1.1 Initialize SDK NET_DVR_Init

Function: public boolean NET_DVR_Init()
Parameter: NULL
Returned Value: Return TRUE on success, FALSE on failure.
Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.The premise of calling device network SDK other functions.

Back

3.1.2 Release SDK Resource NET_DVR_Cleanup

Function: public boolean NET_DVR_Cleanup()
Parameter: NULL
Returned Value: Return TRUE on success, FALSE on failure.
Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Call before finish. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Back

3.2 SDK Local Function

SDK Local Parameter Settings

3.2.1 Get SDK Local Parameter NET_DVR_GetSDKLocalConfig

Function: public boolean NET_DVR_GetSDKLocalConfig(NET_DVR_SDKLOCAL_CFG lpSdkCfg)
Parameter: [in] lpSdkCfg Local configuration parameter. See: NET_DVR_SDKLOCAL_CFG.
Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.
Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.2.2 Set SDK Local Parameter NET_DVR_SetSDKLocalConfig

Function: public boolean NET_DVR_SetSDKLocalConfig(NET_DVR_SDKLOCAL_CFG lpSdkCfg)

Parameter: [in] lpSdkCfg Local configuration parameter. See: NET_DVR_SDKLOCAL_CFG.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Connecting and Receiving Timeout and Reconnecting Settings

3.2.3 Set Network Connection Timeout NET_DVR_SetConnectTime

Function: public boolean NET_DVR_SetConnectTime(int iWaitTime)

Parameter: [in] iWaitTime Timeout time (ms), value range [300,60000]. The actual max timeout time varies because of different systems connection timeout.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Default timeout of SDK to establish a connection is 3 seconds. API will not return FALSE when the set timeout value is greater or less than the limit in SDK 4.0 and future versions. It will take the nearest upper and lower limit value as the actual timeout.

Back

3.2.4 Set Reconnection Function NET_DVR_SetReconnect

Function: public boolean NET_DVR_SetReconnect(int iInterval, boolean bEnableRecon)

Parameter: [in] iInterval Reconnection interval (millisecond).

[in] bEnableRecon Enable or disable reconnection function, false-disable, true-enable (default).

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This interface can control the reconnection function for live view, transparent channel and arming at the same time. By default, SDK enables the reconnection function for live view, transparent channel and arming if not calling this API. The reconnection interval is 5s.

Back

3.2.5 Set Receiving Timeout NET_DVR_SetRecvTimeOut

Function: public boolean NET_DVR_SetRecvTimeOut(int nRecvTimeOut)

Parameter: [in] nRecvTimeOut Receiving timeout (millisecond). Default: 5000ms. Min.: 3000ms.

Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. This API is used to set receiving timeout. E.g., receiving real-time stream data of live view, receiving record data of playback and downloading, receiving alarm message, etc.

Back

SDK Version and Logs

3.2.6 Get SDK Version Information NET_DVR_GetSDKVersion

Function:	public int NET_DVR_GetSDKVersion()
Parameter:	NULL
Returned Value:	Get SDK version information
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. SDK version information. 2 higher bytes indicate the main version number and 2 lower bytes mean sub version number. E.g. 0x00030000: the version is 3.0.

Back

3.2.7 Get SDK Version and Build Information NET_DVR_GetSDKBuildVersion

Function:	public int NET_DVR_GetSDKBuildVersion()
Parameter:	NULL
Returned Value:	Get SDK version and build information.
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. The API is used to get the SDK version and build number. 2 higher bytes indicate the version number: the bits from 25 to 32 mean major version number, and bits from 17 to 24 mean minor version number. 2 lower bytes mean build number, E.g. 0x03000101: the version is 3.0, build number is 0101.

Back

3.2.8 Start Writing Log File NET_DVR_SetLogToFile

Function:	public boolean NET_DVR_SetLogToFile(int bLogEnable, java.lang.String strLogDir, boolean bAutoDel)	
Parameter:	[in] bLogEnable	Log level: 0-close log(default); 1-output ERROR log only; 2-output ERROR and DEBUG log; 3-output all log, including ERROR, DEBUG and INFO log.
	[in] strLogDir	Log file saving path.
	[in] bAutoDel	Whether to delete the exceeded files. Default: TRUE.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. The log file path must be absolute path, and should be ended with **/**.

Back

Exception Message Callback

3.2.9 Register Callback Function of Receiving Exception and Reconnection Message NET_DVR_SetExceptionCallBack

Function: public boolean NET_DVR_SetExceptionCallBack (ExceptionCallBack CallBack)

Parameter: [in] CallBack Callback function to receive exception message. Callback current exception relevant information.

```
public interface ExceptionCallBack {
    public void fExceptionCallBack(int iType, int iUserID, int iHandle);
}
```

[out] iType Message types of exception or reconnection. See Table 3.1.

[out] iUserID Login ID

[out] iHandle Handle of exception relevant types.

Table 3.1 Exception Message Type

dwType Macro Definition	Value	Implication
EXCEPTION_EXCHANGE	0x8000	User interaction exception (Heartbeat timeout when registering, 2 minutes interval.)
EXCEPTION_AUDIOEXCHANGE	0x8001	Two-way audio exception
EXCEPTION_ALARM	0x8002	Alarm exception
EXCEPTION_PREVIEW	0x8003	Network live view exception
EXCEPTION_SERIAL	0x8004	Transparent channel exception
EXCEPTION_RECONNECT	0x8005	Reconnection when live view
EXCEPTION_ALARMRECONNECT	0x8006	Reconnection when alarm triggered
EXCEPTION_SERIALRECONNECT	0x8007	Transparent channel recorrection
SERIAL_RECONNECTSUCCESS	0x8008	Transparent channel recorrection succeeded
EXCEPTION_PLAYBACK	0x8010	Playback exception
EXCEPTION_DISKFMT	0x8011	HDD format
EXCEPTION_EMAILTEST	0x8013	Mail test exception
EXCEPTION_BACKUP	0x8014	Backup exception
PREVIEW_RECONNECTSUCCESS	0x8015	Reconnection when live view succeeded
ALARM_RECONNECTSUCCESS	0x8016	Reconnection when alarm triggered succeeded

RESUME_EXCHANGE	0x8017	User interaction recovery

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Get Error Message

3.2.10 Return Error Code of the Last Operation NET_DVR_GetLastError

Function: public int NET_DVR_GetLastError()

Parameter: NULL

Returned Value: Return error code of the last operation. See Error Code Definition.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Return the error code.

Back

3.2.11 Return Error Message of the Last Operation NET_DVR_GetErrorMsg

Function: public String NET_DVR_GetErrorMsg(INT_PTR pErrNo);

Parameter: [out] pErrNo Error code.

Returned Value: Return error description information.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.3 User Register

3.3.1 Dynamic IP Address and Port Resolution NET_DVR_GetAddrInfoByServer

Function: public boolean NET_DVR_GetAddrInfoByServer(int dwQueryType, NET_DVR_ADDR_QUERY_COND pCond, NET_DVR_ADDR_QUERY_RET pRet)

Parameter: [in] dwQueryType Searching type. For the value, see ADDR_QUERY_TYPE; for the corresponding relation, see Table 3.2.

[in] pCond Searching condition. For the corresponding relation, see Table 3.2.

[out] pRet Searching result. For the corresponding relation, see Table 3.2.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Different searching types correspond to different structs and searching results. For the corresponding relation, see Table 3.2.

Table 3.2 Domain Resolution Type

dwQueryType Definition	Implication	pCond Corresponding Struct	pRet Corresponding Struct
QUERYSVR_BY_COUNTRYID	Search server address by	NET_DVR_QUERY_COUNTRYID_CO	NET_DVR_QUERY_COUNTRYID_R

	national No.	ND	ET
QUERYDEV_BY_NICKNAME_DDNS	Search device information by device name from hiddns	NET_DVR_QUERY_DDNS_COND	NET_DVR_QUERY_DDNS_RET
QUERYDEV_BY_SERIAL_DDNS	Search device information by device No. from hiddns	NET_DVR_QUERY_DDNS_COND	NET_DVR_QUERY_DDNS_RET
CHECKDEV_BY_NICKNAME_DDNS	Diagnose device by device name from hiddns	NET_DVR_QUERY_DDNS_COND	NET_DVR_CHECK_DDNS_RET
CHECKDEV_BY_SERIAL_DDNS	Diagnose device by device No. from hiddns	NET_DVR_QUERY_DDNS_COND	NET_DVR_CHECK_DDNS_RET
QUERYDEV_BY_NICKNAME_IPSERVER	Search device information by device name from IPServer	NET_DVR_QUERY_IPSERVER_COND	NET_DVR_QUERY_IPSERVER_RET
QUERYDEV_BY_SERIAL_IPSERVER	Search device information by device No. from IPServer	NET_DVR_QUERY_IPSERVER_COND	NET_DVR_QUERY_IPSERVER_RET

Back

3.3.2 Get Dynamic IP Address and Port No. by Resolution Server

NET_DVR_GetDVRIPByResolveSvr_EX

Function: public boolean NET_DVR_GetDVRIPByResolveSvr_EX(java.lang.String sServerIP, short wServerPort, java.lang.String sDVRName, short wDVRNameLen, java.lang.String sDVRSerialNumber, short wDVRSerialLen, NET_DVR_RESOLVE_DEVICEINFO lpDeviceInfo)

Parameter:

[in]sServerIP	IP address or domain name of resolution server.
[in]wServerPort	Resolution server port number. IP Server: 7071, EasyDDNS: 80.
[in]sDVRName	Device name.
[in]wDVRNameLen	Length of the device name.
[in]sDVRSerialNumber	The device serial number.
[in]wDVRSerialLen	The length of the device serial number.
[out] lpDeviceInfo	The device IP address, port information. See: NET_DVR_RESOLVE_DEVICEINFO

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Resolve the device IP address and port No. by device domain name or serial number, and then call NET_DVR_Login_V30 to login the device. The device name and serial number cannot be NULL at the same time. IPServer is domain resolution server software provided by HIKVISION.

Back

3.3.3 Activate Device NET_DVR_ActivateDevice

Function: public boolean NET_DVR_ActivateDevice(String sDvrIp, int iDvrPort, NET_DVR_ACTIVATECFG

	lpActivateCfg)	
Parameter:	[in] sDvrIp	Device address.
	[in] iDvrPort	Device port No.
	[in] lpActivateCfg	Activate parameter. See: NET_DVR_ACTIVATECFG.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

3.3.4 User Registers Device NET_DVR_Login_V30

Function:	public int NET_DVR_Login_V30(java.lang.String sDvrIp, int iDvrPort, ava.lang.String sUserName, java.lang.String sPassword, NET_DVR_DEVICEINFO_V30 DeviceInfo)	
Parameter:	[in] sDvrIp	Device IP address or static domain.
	[in] iDvrPort	Device port No.
	[in] sUserName	Username.
	[in] sPassword	Password.
	[out] DeviceInfo	Device information. See: NET_DVR_DEVICEINFO_V30.
Returned Value:	Return -1 on failure, and other returned value indicates the user ID. The userID is unique, and is indispensable for operating the device. Call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. SDK supports static domain when registering device, which means setting sDVRIP as test.vicp.net.	

Back

3.3.5 User Logout NET_DVR_Logout_V30

Function:	public boolean NET_DVR_Logout_V30 (int lUserID)	
Parameter:	[in]lUserID	User ID. The returned value of NET_DVR_Login_V30.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

3.4 Get Device Ability Set

3.4.1 Get Device Ability Set NET_DVR_GetXMLAbility

Function:	public boolean NET_DVR_GetXMLAbility(int lUserID, int dwAbilityType, byte[] pInBuf, int dwInBufLen, byte[] pOutBuf, int dwOutBufLen, INT_PTR lpSizeReturned)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.

	[in] dwAbilityType	Ability type. See Table 3.3.
	[in] pInBuf	Input buffer. Different abilities correspond to different input content. See Table 3.3.
	[in] dwInLength	Input buffer length.
	[out] pOutBuf	Output buffer. Different abilities correspond to different output content. See Table 3.3.
	[in] dwOutLength	The size of receiving data buffer.
	[out] lpSizeReturned	pOutBuf actual effective length.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. When getting the ability set, the input and output parameter format is listed in Table 3.3.	

Table 3.3 Ability Set Definition

dwAbilityType Macro Definition	Implication	pInBuf	pOutBuf
DEVICE_SOFTHARDWARE_ABILITY	Get device hardware and software ability	None	Software and hardware ability XML description
DEVICE_ENCODE_ALL_ABILITY_V20	Get all encoding ability	Get encoding ability input XML description	All the encoding ability XML descriptions

***Note:* See *Device Network SDK Programming Manual.chm* for ability set struct and XML description.**

Back

3.5 Real-time Live View

Mandatory I Frame

3.5.1 Dynamically Create a Key Frame in Main Stream NET_DVR_MakeKeyFrame

Function:	public boolean NET_DVR_MakeKeyFrame(int lUserID, int lChannel)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] lChannel	Channel No., analog channel starting from 1 and IP channel starting from 33.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. The API is used to reset I frame. Call NET_DVR_MakeKeyFrame or NET_DVR_MakeKeyFrameSub to reset I frame for the main stream or sub stream according to the set live view parameter (NET_DVR_PREVIEWINFO).	

Back

3.5.2 Dynamically Create a Key Frame in Sub Stream

NET_DVR_MakeKeyFrameSub

Function: public boolean NET_DVR_MakeKeyFrameSub(int lUserID, int lChannel)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] lChannel	Channel No., analog channel starting from 1 and IP channel starting from 33.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. The API is used to reset I frame NET_DVR_MakeKeyFrame or NET_DVR_MakeKeyFrameSub to reset I frame for the main stream or sub stream according to the set live view parameter (NET_DVR_PREVIEWINFO).

Back

Real-time Live View

3.5.3 Real-time Live View NET_DVR_RealPlay_V40

Function: public int NET_DVR_RealPlay_V40(int lUserID, NET_DVR_PREVIEWINFO previewInfo, RealPlayCallBack CallBack)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] previewInfo	Live view parameter, including stream type, getting stream protocol, channel No. See: NET_DVR_PREVIEWINFO.
[in] CallBack	Stream data callback function.

```
public interface RealPlayCallBack {
    public void fRealDataCallBack(int iRealHandle, int iDataType, byte[] pDataBuffer, int iDataSize);
}
```

[out] iRealHandle	Current live view handle.
[out] iDataType	Data type.
[out] pDataBuffer	Buffer pointer for saving data.
[out] iDataSize	Buffer size.

Table 3.4 Stream Data Type

dwDataType Macro Definition	Value	Implication
NET_DVR_SYSHEAD	1	System head data
NET_DVR_STREAMDATA	2	Stream data (including video and audio stream, or only the video data of stream that video and audio is separated)
NET_DVR_AUDIOSTREAMDATA	3	Audio data

Returned Value: Return -1 on failure and other values as the handle parameters of NET_DVR_StopRealPlay. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Get real-time stream video and audio data by setting real-time callback function, and then decode to display via the player.

Back

3.5.4 Stop Live View NET_DVR_StopRealPlay

Function: public boolean NET_DVR_StopRealPlay(int iRealHandle)

Parameter: [in] iRealHandle Live view handle, the returned value of NET_DVR_RealPlay_V40.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Display Parameter Settings

3.5.5 Get Live View's Display Parameter NET_DVR_ClientGetVideoEffect

Function: public boolean NET_DVR_ClientGetVideoEffect(int lRealHandle, NET_DVR_VIDEOEFFECT VideoEffect)

Parameter: [in] lRealHandle Live view handle, the returned value of NET_DVR_RealPlay_V40.

[out] VideoEffect Display parameter. See: NET_DVR_VIDEOEFFECT.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API can be called during preview and the video parameters of the current channel (e.g. brightness, contrast) are returned by the device.

Back

3.5.6 Set Live View's Display Parameter NET_DVR_ClientSetVideoEffect

Function: public boolean NET_DVR_ClientSetVideoEffect(int lRealHandle, NET_DVR_VIDEOEFFECT VideoEffect)

Parameter: [in] lRealHandle Live view handle, the returned value of NET_DVR_RealPlay_V40.

[in] VideoEffect Display parameter. See: NET_DVR_VIDEOEFFECT.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API can be called during live view and set the video parameters of the current channel. This API will not return FALSE when the set brightness or contrast value is greater or less than the limit in SDK 4.0 and future versions. It will take the nearest upper and lower limit value as the actual parameter value.

Back

Channel-Zero Live View

3.5.7 Start Channel-Zero Live View NET_DVR_ZeroStartPlay

Function: public int NET_DVR_ZeroStartPlay(int lUserID, NET_DVR_CLIENTINFO ClientInfo, RealPlayCallBack CallBack, boolean bBlock)

Parameter:

[in] lUserID The returned value of NET_DVR_Login_V30.

[in] ClientInfo Live view parameter. See: NET_DVR_CLIENTINFO.

[in] CallBack Stream data callback function.

[in] bBlock Whether the stream request process is blocked: 0-unblocked, 1-blocked.

```
public interface RealPlayCallBack {
    public void fRealDataCallBack(int iRealHandle, int iDataType, byte[] pDataBuffer, int iDataSize);
}
```

[out] iRealHandle [out] iRealHandle

[out] iDataType [out] iDataType

[out] pDataBuffer [out] pDataBuffer

[out] iDataSize [out] iDataSize

Table 3.5 Stream Data Type

dwDataType Macro Definition	Value	Implication
NET_DVR_SYSHEAD	1	System head data
NET_DVR_STREAMDATA	2	Stream data (including video and audio stream, or only the video data of stream that video and audio is separated)
NET_DVR_AUDIOSTREAMDATA	3	Audio data

Returned Value: Return -1 on failure and other values as the handle parameters of NET_DVR_ZeroStopPlay. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API supports to set current live view operation to be blocked or not (by the parameter: bBlocked). If set to be unblocked, it means the connection is successful when start to connect with the device. If failed to receive stream and play, it will notify the upper layer by exception live view mode. If set to be blocked, it means it will return whether successful or not after playing operation.

Back

3.5.8 Stop Channel-Zero Live View NET_DVR_ZeroStopPlay

Function: public boolean NET_DVR_ZeroStopPlay(int iRealHandle)

Parameter: [in] iRealHandle Live view handle, the returned value of NET_DVR_ZeroStartPlay.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Control Client Recording

3.5.9 Get Live View Data and Save in Specified File NET_DVR_SaveRealData

Function: public boolean NET_DVR_SaveRealData(int lRealHandle, String sFileName)

Parameter:
[in] lRealHandle Live view handle, the returned value of NET_DVR_RealPlay_V40.
[in] sFileName File path name.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Users can save videos through this API, and the maximum limit of the file is 1024 MB. If the file is more than 1024 MB, the SDK will save it into the new file automatically. The first 40 bytes will be written into the file automatically. File naming rule: add the digital identity basis on the file name (for example: * _1.mp4, * _2.mp4).

Back

3.5.10 Stop Getting Data NET_DVR_StopSaveRealData

Function: public boolean NET_DVR_StopSaveRealData(int lRealHandle)

Parameter:
[in] lRealHandle Live view handle, the returned value of NET_DVR_RealPlay_V40.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.6 Capture Image

3.6.1 Capture a Frame and Save as JPEG Picture NET_DVR_CaptureJPEGPicture

Function: public boolean NET_DVR_CaptureJPEGPicture(int lUserID, int lChannel, NET_DVR_JPEGPARA lpJpegPara, java.lang.String sPicFileName)

Parameter:
[in] lUserID The returned value of NET_DVR_Login_V30.
[in] lChannel Channel No.
[in] lpJpegPara JPEG picture parameter. See: NET_DVR_JPEGPARA.
[in] sPicFileName The file path to save JPEG picture.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API is used to capture a frame data and save it as JPEG picture. Network camera supports capturing current video resolution.

Back

3.6.2 Capture a Frame and Save as JPEG Picture in Specified Memory Space

NET_DVR_CaptureJPEGPicture_NEW

Function:	public boolean NET_DVR_CaptureJPEGPicture_NEW(int lUserID, int lChannel, NET_DVR_JPEGPARA lpJpegPara, byte[] lpJpegPicBuffer, int dwPicSize, INT_PTR lpSizeReturned)	
Parameter:	[in]lUserID	The returned value of NET_DVR_Login_V30.
	[in]lChannel	Channel No.
	[in]lpJpegPara	JPEG picture parameter. See: NET_DVR_JPEGPARA.
	[in]sJpegPicBuffer	The buffer saving JPEG data.
	[in]dwPicSize	The input buffer size.
	[out]lpSizeReturned	The size of returned picture data.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. This API is used to capture a frame data and save it as JPEG picture. Network camera supports capturing current video resolution.	

Back

3.7 Arm and Disarm

Set Callback Function of Alarm Message Upload

3.7.1 Register Callback Function and Receive Alarm Message

NET_DVR_SetDVRMessageCallBack_V30

Function:	public boolean NET_DVR_SetDVRMessageCallBack_V30(FMSGCallBack fMessageCallBack, Pointer pUser)	
Parameter:	[in]fMessageCallBack	Alarm message callback function.
	[in] pUser	User Parameter

```
public interface FMSGCallBack extends Callback{
    public void invoke(int lCommand, NET_DVR_ALARMER pAlarmer, Pointer pAlarmInfo, int dwBufLen, Pointer pUser);
}
```

[out] lCommand	Uploaded message type. See Table 3.6.
[out] Alarmer	Alarm device information. See: NET_DVR_ALARMER.
[out] AlarmInfo	Alarm message. See: NET_DVR_BASE_ALARM.
[out] dwBufLen	Size of alarm information buffer
[out] pUser	User parameter

Table 3.6 Alarm Message Type

lCommand Macro Definition	Value	Implication
COMM_ALARM	0x1100	Alarm message uploading for the devices supported by version V3.0 or lower
COMM_ALARM_V30	0x4000	Alarm message uploading for the devices supported by version V3.0 or above
COMM_ALARM_V40	0x4007	The size of alarm message data is variable
COMM_ALARM_RULE	0x1102	Behavior analysis information uploading for the devices
COMM_UPLOAD_PLATE_RESULT	0x2800	Traffic capture results uploading for the devices (old alarm message, arming parameter byAlarmInfoType is set to 0)
COMM_ITS_PLATE_RESULT	0x3050	Traffic capture results uploading for the devices (new alarm message, arming parameter byAlarmInfoType is set to 1)

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hcnetsdk.jna.HCNetSDKByJNA, JNA API. The first parameter (lCommand) and the third parameter (pAlarmInfo) are closely related. See Table 3.7.

Table 3.7 Alarm Message

Message type lCommand	Uploaded Content	pAlarmInfo Corresponding Struct
COMM_ALARM	Alarm message of the devices supported by version V3.0 or lower	NET_DVR_ALARMINFO
COMM_ALARM_V30	Alarm message of the devices supported by version V3.0 or above	NET_DVR_ALARMINFO_V30
COMM_ALARM_V40	The size of alarm message of the devices is variable	NET_DVR_ALARMINFO_V40
COMM_ALARM_RULE	Behavior analysis information uploading for the devices	NET_VCA_RULE_ALARM
COMM_UPLOAD_PLATE_RESULT	Traffic capture results uploading for the devices (old alarm message, arming parameter byAlarmInfoType is set to 0)	NET_DVR_PLATE_RESULT
COMM_ITS_PLATE_RESULT	Traffic capture results uploading for the devices (new alarm message, arming parameter byAlarmInfoType is set to 1)	NET_ITS_PLATE_RESULT

Back

Arm and Disarm

3.7.2 Setup Alarm Uploading Channel for Alarm Message

NET_DVR_SetupAlarmChan_V41

Function:	public int NET_DVR_SetupAlarmChan_V41(int lUserID, Pointer lpSetupParam)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] lpSetupParam	Alarm arming parameters, the corresponding structure: NET_DVR_SETUPALARM_PARAM
Returned Value:	Return -1 on failure and other value as the handle parameter of NET_DVR_CloseAlarmChan_V30. Call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hcnetsdk.jna.HCNetSDKByJNA, JNA API. This API supports to upload alarm message for the devices supported by version V3.0 or above. Before enabling arming, it is required to call NET_DVR_SetDVRMessageCallBack_V30 to get the uploaded alarm message.	

Back

3.7.3 Revoke Alarm Uploading Channel NET_DVR_CloseAlarmChan_V30

Function:	public boolean NET_DVR_CloseAlarmChan_V30(int lAlarmHandle)	
Parameter:	[in] lAlarmHandle	The returned value of NET_DVR_SetupAlarmChan_V30.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hcnetsdk.jna.HCNetSDKByJNA, JNA API.	

Back

3.8 Remote Parameter Settings

General Parameter Settings

3.8.1 Get Device Settings Information NET_DVR_GetDVRConfig

Function:	public boolean NET_DVR_GetDVRConfig(int lUserID, int dwCommand, int lChannel, NET_DVR_CONFIG DVRConfig)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] dwCommand	Configuration command. See Table 3.8.
	[in] lChannel	Channel No.. Different commands correspond to different values. If this parameter is invalid, set it as 0xFFFFFFFF. See Table 3.8.
	[out] DVRConfig	Configuration information. Different configurations correspond to different types. See Table 3.8.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. The structs and commands are different according to the various getting functions. See Table 3.8.

Table 3.8 Getting Parameter Command

dwCommand Macro Definition	dwCommand Implication	Channel No.	DVRConfig Corresponding Type	Value
NET_DVR_GET_DEVICECFG_V40	Get device parameter	Invalid	NET_DVR_DEVICECFG_V40	1100
NET_DVR_GET_TIMECFG	Get time parameter	Invalid	NET_DVR_TIME	118
NET_DVR_GET_USERCFG_V30	Get user parameter	Invalid	NET_DVR_USER_V30	1006
NET_DVR_GET_PICCFG_V30	Get image parameter	Invalid	NET_DVR_PICCFG_V30	1002
NET_DVR_GET_COMPRESSCFG_V30	Get compression parameter	Invalid	NET_DVR_COMPRESSIONCFG_V30	1040
NET_DVR_GET_RECORDCFG_V30	Get recording parameter	Invalid	NET_DVR_RECORD_V30	1004
NET_DVR_GET_SHOWSTRING_V30	Get OSD parameter	Channel No.	NET_DVR_SHOWSTRING_V30	1030
NET_DVR_GET_ALARMINCFG_V30	Get alarm input parameter	Alarm input No.	NET_DVR_ALARMINCFG_V30	1024
NET_DVR_GET_ALARMOUTCFG_V30	Get alarm output parameter	Alarm output No. starting from 0.	NET_DVR_ALARMOUTCFG_V30	1026
NET_DVR_GET_DECODERCFG_V30	Get RS485 serial port parameter	Channel No.	NET_DVR_DECODERCFG_V30	1042
NET_DVR_GET_IPPARACFG_V40	Get IP access parameter	Group No.	NET_DVR_IPPARACFG_V40	1062
NET_DVR_GET_IPALARMOUTCFG	Get IP alarm output access paerameter	Invalid	NET_DVR_IPALARMOUTCFG	1052
NET_DVR_GET_NETCFG_V30	Get network parameter	Invalid	NET_DVR_NETCFG_V30	1000
NET_DVR_GET_DDNSCFG_V30	Get DDNS configuration	Invalid	NET_DVR_DDNSPARA_V30	1010
NET_DVR_GET_NTPCFG	Get NTP parameter	Invalid	NET_DVR_NTPPARA	224
NET_DVR_GET_WIFI_STATUS	Get WIFI status	Invalid	NET_DVR_WIFI_CONNECT_STATUS	310
NET_DVR_GET_AP_INFO_LIST	Get wireless network resource parameter	Invalid	NET_DVR_AP_INFO_LIST	305
NET_DVR_GET_WIFI_CFG	Get IP camera wireless parameter	Invalid	NET_DVR_WIFI_CFG	307
NET_DVR_GET_COMPRESSCFG_AUD	Get two-way audio parameter	Invalid	NET_DVR_COMPRESSION_AUDIO	1058
NET_IPC_GET_AUX_ALARMCFG	Get aux alarm parameter	Channel No.	NET_IPC_AUX_ALARMCFG	3209
NET_DVR_GET_ZEROCHANCFG	Get channel-zero compression parameter	Channel No.	NET_DVR_ZEROCHANCFG	1102
NET_DVR_GET_DIGITAL_CHANNEL_STATE	Get digital channel status	Invalid	NET_DVR_DIGITAL_CHANNEL_STATE	6126
NET_DVR_GET_PRESET_NAME	Get preset name	Channel No.	NET_DVR_PRESET_NAME_ARRAY	3383

Back

3.8.2 Set Device Settings Information NET_DVR_SetDVRConfig

Function: public boolean NET_DVR_SetDVRConfig(int lUserID, int dwCommand, int lChannel, NET_DVR_CONFIG DVRConfig)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] dwCommand	Device configuration command. See Table 3.9.
[in] lChannel	Channel No.. Different commands correspond to different values. If this parameter is invalid, set it as 0xFFFFFFFF. See Table 3.9.
[in] DVRConfig	Configuration information. Different functions correspond to different structs. Seee Table 3.9.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. The structs and command numbers are different according to the various configuration functions. See Table 3.9.

Table 3.9 Setting Device Parameter

dwCommand Macro Definition	dwCommand Implication	Channel No.	lpOutBuffer Corresponding Struct	Value
NET_DVR_SET_DEVICECFG_V40	Set device parameter	Invalid	NET_DVR_DEVICECFG_V40	1101
NET_DVR_SET_TIMECFG	Set time parameter	Invalid	NET_DVR_TIME	119
NET_DVR_SET_USERCFG_V30	Set user parameter	Invalid	NET_DVR_USER_V30	1007
NET_DVR_SET_PICCFG_V30	Set image parameter	Channel No.	NET_DVR_PICCFG_V30	1003
NET_DVR_SET_COMPRESSCFG_V30	Set compression parameter	Channel No.	NET_DVR_COMPRESSIONCFG_V30	1041
NET_DVR_SET_RECORDCFG_V30	Set recording parameter	Channel No.	NET_DVR_RECORD_V30	1005
NET_DVR_SET_SHOWSTRING_V30	Set OSD parameter	Channel No.	NET_DVR_SHOWSTRING_V30	1031
NET_DVR_SET_ALARMINCFG_V30	Set alarm input parameter	Alarm input No.	NET_DVR_ALARMINCFG_V30	1025
NET_DVR_SET_ALARMOUTCFG_V30	Set alarm output parameter	Alarm output No. starting from 0.	NET_DVR_ALARMOUTCFG_V30	1027
NET_DVR_SET_DECODERCFG_V30	Set RS485 serial port parameter	Channel No.	NET_DVR_DECODERCFG_V30	1043
NET_DVR_SET_IPPARACFG_V40	Set IP access parameter	Group No.	NET_DVR_IPPARACFG_V40	1063
NET_DVR_SET_NETCFG_V30	Set network parameter	Invalid	NET_DVR_NETCFG_V30	1001
NET_DVR_SET_DDNSCFG_V30	Set DDNS configuration	Invalid	NET_DVR_DDNSPARA_V30	1011
NET_DVR_SET_NTPCFG	Set NTP parameter	Invalid	NET_DVR_NTPPARA	225
NET_DVR_SET_WIFI_CFG	Get IP camera wireless parameter	Invalid	NET_DVR_WIFI_CFG	306
NET_IPC_SET_AUX_ALARMCFG	Get aux alarm parameter	Channel No.	NET_IPC_AUX_ALARMCFG	3210
NET_DVR_SET_ZEROCHANCFG	Get channel-zero compression parameter	Channel No.	NET_DVR_ZEROCHANCFG	1103

Back

3.8.3 Get Device Setting Information in Batch NET_DVR_GetDeviceConfig

Function: public boolean NET_DVR_GetDeviceConfig(int lUserID, int dwCommand, int dwCount, Pointer lpInBuffer, int dwInBufferSize, Pointer lpStatusList, Pointer lpOutBuffer, int dwOutBufferSize)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V40.
[in] dwCommand	Configuration command. See Table 3.10.
[in] dwCount	The number of the setting information for each time, 0 and 1 mean one information, 2 means two information, the maximum is 64.
[in] lpInBuffer	The setting condition buffer. See Table 3.11.
[in] dwInBufferSize	The size of input buffer.
[out] lpStatusList	The error information list, which is one-to-one corresponding to the configuration such as lpStatusList[2] corresponds to lpInBuffer[2]. The size of each error information is 4 bytes. Parameter: 0 or 1 means successful, the others mean the error code when failed.
[out] lpOutBuffer	The returned parameter by the device (see Table 3.11), which needs to one to one corresponds to the cameras. If lpStatusList value is larger than 0, the corresponding lpOutBuffer is invalid.
[in] dwOutBufferSize	The size of output buffer.

Returned Value: Return TRUE on success, FALSE on failure. If it returns TRUE, it doesn't mean each configuration is successful. You need to check the value of lpStatusList[n]. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hcnetsdk.jna.HCNetSDKByJNA, JNA API.This API is a common API which can send data and get setting information in batch. lpInBuffer is used to get information and lpOutBuffer is used to save the setting information. The structs and commands are different according to the various getting functions. See Table 3.11.

Table 3.1 Batch Getting Parameters Command

dwCommand Macro Definition	dwCommand Implication	Value
NET_DVR_GET_MULTI_STREAM_COMPRESSIONCFG	Remotely get multi-streams encoding parameter.	3216

Table 3.2 Batch Getting Device Parameter Command

dwCommand Macro Definition	lpInBuffer corresponding structure	lpOutBuffer corresponding structure
NET_DVR_GET_MULTI_STREAM_COMPRESSIONCFG	dwCount NET_DVR_MULTI_STREAM_COMPRESSIONCFG_COND	dwCount NET_DVR_MULTI_STREAM_COMPRESSIONCFG

Back

3.8.4 Set Device Setting Information in Batch NET_DVR_SetDeviceConfig

Function: public boolean NET_DVR_SetDeviceConfig (int lUserID, int dwCommand, int dwCount, Pointer lpInBuffer, int dwInBufferSize, Pointer lpStatusList, Pointer lpInParamBuffer, int dwInParamBufferSize)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V40.
[in] dwCommand	Configuration command. See Table 3.12.

	[in] dwCount	The number of the setting information for each time, 0 and 1 mean one information, 2 means two information, the maximum is 64.
	[in] lpInBuffer	The setting condition buffer. See Table 3.13.
	[in] dwInBufferSize	The size of input buffer.
	[out] lpStatusList	The error information list, which is one-to-one corresponding to the configuration such as lpStatusList[2] is corresponding to lpInBuffer[2]. The size of each error information is 4 bytes. Parameter: 0 or 1 means successful, the others mean the error code when failed.
	[in] lpInParamBuffer	The setting parameter for the device (see Table 3.13), which needs to one to one corresponds to lpInBuffer. If lpStatusList value is larger than 0, setting corresponding lpInBuffer fails. If the value is 0, setting corresponding lpInBuffer succeeds.
	[in] dwInParamBufferSize	The size of setting buffer.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns TRUE, it doesn't mean each configuration is successful. You need to check the value of lpStatusList[n]. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hcnetsdk.jna.HCNetSDKByJNA, JNA API.This API is a common API which can send data and set setting information for device in batch. lpInBuffer is used to set *dwCount* and lpInParamBuffe is used to set the dwCount setting information. The structures and commands are different according to the various setting functions. See Table 3.13.	

Table 3.3 Setting Parameters in Batch Command

dwCommand Macro Definition	dwCommand Implication	Value
NET_DVR_SET_MULTI_STREAM_COMPRESSIONCFG	Remotely set multi-streams encoding parameter.	3217

Table 3.4 Setting Device Parameters in Batch Command

DwCommand Macro Definition	lpInBuffer Corresponding Structure	lpInParamBuffer Corresponding Structure
NET_DVR_SET_MULTI_STREAM_COMPRESSIONCFG	DwCount NET_DVR_MULTI_STREAM_COMPRESSIONCFG_COND	DwCount NET_DVR_MULTI_STREAM_COMPRESSIONCFG

Back

Alarm Output Settings

3.8.5 Get Device Alarm Output NET_DVR_GetAlarmOut_V30

Function:	public boolean NET_DVR_GetAlarmOut_V30(int lUserID, NET_DVR_ALARMOUTSTATUS_V30 AlarmStatus)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[out] AlarmStatus	Alarm output status. See: NET_DVR_ALARMOUTSTATUS_V30.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.8.6 Set Device Alarm Output NET_DVR_SetAlarmOut

Function: public boolean NET_DVR_SetAlarmOut(int lUserID, int lAlarmOutPort, int lAlarmOutStatic)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] lAlarmOutPort	Alarm output port, starting from 0. 0x00ff means all analog output, and 0xff00 means all digital output. Device supports handling alarm output of IP access, and the values 32 to 95 are digital alarm output.
[in] lAlarmOutStatic	Alarm output status: 0-Stop output, 1-Output

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

PTZ Protocol Supported by Device

3.8.7 Get PTZ Protocol Supported by Device NET_DVR_GetPTZProtocol

Function: public boolean NET_DVR_GetPTZProtocol(int lUserID, NET_DVR_PTZCFG struPtz)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[out] struPtz	PTZ protocol. See: NET_DVR_PTZCFG.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Call this API when setting the device PTZ protocol to get the supported PTZ protocol.

Back

3.9 Record Playback, Download, Lock and Backup

Refresh Record Index

3.9.1 Refresh Record Index Instantly NET_DVR_UpdateRecordIndex

Function: public boolean NET_DVR_UpdateRecordIndex(int lUserID, int dwChannel)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] dwChannel	Channel No.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. After refreshing the record index, it can playback the video file before refreshing. It is relatedtothe channel, and should be supported by device. By default, the device refreshes every two minutes.

Back

Search Video File

3.9.2 Search Video File by File Type and Time NET_DVR_FindFile_V30

Function: public int NET_DVR_FindFile_V30(int lUserID, NET_DVR_FILECOND pFindCond)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[in] pFindCond The file information structure to be searched. See: NET_DVR_FILECOND.

Returned Value: Return -1 on failure, other values as the parameters of NET_DVR_FindClose. If it returns -1, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API specifies the video file information to be searched. After calling this API, you can call NET_DVR_FindNextFile_V30 to get file information.

Back

3.9.3 Get Searched File Information One by One NET_DVR_FindNextFile_V30

Function: public int NET_DVR_FindNextFile_V30(int lFindHandle, NET_DVR_FINDDATA_V30 lpFindData)

Parameter: [in] lFindHandle File searching handle. The returned value of NET_DVR_FindFile_V30.

[in] lpFindData The pointer saving file information. See: NET_DVR_FINDDATA_V30.

Returned Value: Return -1 on failure, other values as the current status. See Table 3.10.

Table 3.10 Searching Result

Macro Definition	Value	Implication
NET_DVR_FILE_SUCCESS	1000	File information gotten
NET_DVR_FILE_NOFIND	1001	No file found
NET_DVR_ISFINDING	1002	Searching. Please wait.
NET_DVR_NOMOREFILE	1003	No more files. Searching ends.
NET_DVR_FILE_EXCEPTION	1004	Exception when searching file.

If it returns failure, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Before calling this API, you need to call NET_DVR_FindFile_V30 to get the current searching handle. This API is used to get one searched file. If you need to get all searched files, you need to loop call this API. You can get the related card No. and information that whether the file is locked.

The maximum file number for each search is 4000.

Back

3.9.4 Cancel File Searching, Release Resource NET_DVR_FindClose_V30

Function: public boolean NET_DVR_FindClose_V30(int lFindHandle)

Parameter: [in]lFindHandle File searching handle. The returned value of NET_DVR_FindFile_V30.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Search Video File by Event

3.9.5 Search Video File by Event NET_DVR_FindFileByEvent

Function: public int NET_DVR_FindFileByEvent(int lUserID, NET_DVR_SEARCH_EVENT_PARAM lpSearchEventParam)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[in] lpSearchEventParam Event searching condition. See: NET_DVR_SEARCH_EVENT_PARAM.

Returned Value: Return -1 on failure, other values as the parameters of NET_DVR_FindNextEvent. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API specifies the searching condition. After calling successfully, we can call NET_DVR_FindNextFile to get file information. This function needs to be supported by devices, and if the device doesn't support, it will return failure. The error code is 23.

Back

3.9.6 Get Searched File One by One NET_DVR_FindNextEvent

Function: public int NET_DVR_FindNextEvent(int lFindHandle, NET_DVR_SEARCH_EVENT_RET lpSearchEventRet)

Parameter: [in] lFindHandle File searching handle. The returned value of NET_DVR_FindFileByEvent.

[in] lpSearchEventRet The pointer saving file information. See: NET_DVR_SEARCH_EVENT_RET.

Returned Value: Return -1 on failure, other values as the current status. See Table 3.11. If it returns failure, call NET_DVR_GetLastError to get the error code and find the reason.

Table 3.11 Searching Result

Macro Definition	Value	Implication
NET_DVR_FILE_SUCCESS	1000	File information gotten
NET_DVR_FILE_NOFIND	1001	No file found

NET_DVR_ISFINDING	1002	Searching. Please wait.
NET_DVR_NOMOREFILE	1003	No more files. Searching ends.
NET_DVR_FILE_EXCEPTION	1004	Exception when searching file.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Before calling this API, you need to call NET_DVR_FindFile_V30 to get the current searching handle. This API is used to get one searched file. If you need to get all searched files, you need to loop call this API. You can get the related card No. and information that whether the file is locked.
For some devices, the maximum file number for each search is 2000, and for some is 4000. As a result, when the searching number is 2000 or 4000, new time period is needed for searching more files.

Back

3.9.7 Cancel File Searching, Release Resource NET_DVR_FindClose_V30

Function: public boolean NET_DVR_FindClose_V30(int lFindHandle)
Parameter: [in]lFindHandle File searching handle. The returned value of NET_DVR_FindFileByEvent.
Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.
Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Playback Video File

3.9.8 Register Callback Function and Capture Video Data

NET_DVR_SetPlayDataCallBack

Function: public boolean NET_DVR_SetPlayDataCallBack(int lPlayHandle, PlaybackCallBack cbPlayDataCallBack)
Parameter: [in]lPlayHandle Play handle. The returned value of NET_DVR_PlayBackByName or NET_DVR_PlayBackByTime.
[in] PlaybackCallBack Video data callback function.

```
public void fPlayDataCallBack(int iPlayHandle, int iDataType, byte[] pDataBuffer, int iDataSize)
   public interface PlaybackCallBack {
      public void fPlayDataCallBack(int iPlayHandle, int iDataType, byte[] pDataBuffer, int iDataSize);
}
```

[out] iPlayHandle Current play handle.
[out] iDataType Data type. See Table 3.12.
[out] pDataBuffer Buffer pointer for saving data.
[out] iDataSize Buffer size.

Table 3.12 Playback Data Type

dwDataType Macro Definition	Value	Implication
NET_DVR_SYSHEAD	1	System head data.
NET_DVR_STREAMDATA	2	Stream data (including video and audio stream, or only the video data of stream that video and audio is separated)

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This function includes starting and stopping the user handling the data captured by SDK. When cbPlayDataCallBack is not NULL, it means SDK will call back the data and user can handle the data. When cbPlayDataCallBack is NULL, it means stopping calling back the data and handling the data. The first package called back by the function is a system head of 40 bytes, and it is used to decode the stream data. The afterward data called back is the compressed data stream.

Back

3.9.9 Playback Video File by File Name NET_DVR_PlayBackByName

Function: public int NET_DVR_PlayBackByName(int lUserID, java.lang.String sPlayBackFileName)

Parameter:
[in] lUserID The returned value of NET_DVR_Login_V30.
[in] sPlayBackFileName The file name. The length should be less than 100 bytes.

Returned Value: Return -1 on failure, other values as the parameters of NET_DVR_StopPlayBack. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. You need to call NET_DVR_SetPlayDataCallBack first to register callback function, capture the stream data and handle it (decode to display). This API specifies the video file to play. After calling, you need to call the NET_DVR_PLAYSTART of NET_DVR_PlayBackControl_V40 to realize playback.

Back

3.9.10 Playback by Time NET_DVR_PlayBackByTime

Function: public int NET_DVR_PlayBackByTime(int lUserID, int lChannel, NET_DVR_TIME lpStartTime, NET_DVR_TIME lpStopTime)

Parameter:
[in] lUserID The returned value of NET_DVR_Login_V30.
[in] lChannel Channel No.
[in] lpStartTime Start time. See: NET_DVR_TIME.
[in] lpStopTime End time. See: NET_DVR_TIME.

Returned Value: Return -1 on failure and other values as the parameters of NET_DVR_StopPlayBack. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. You need to call NET_DVR_SetPlayDataCallBack first to register callback function, capture the stream data and handle it (decode to display). This API specifies the video file's start time and end time. After calling successfully, you need to call the NET_DVR_PLAYSTART of NET_DVR_PlayBackControl_V40 to realize playback.
When playing back video file searched by event, due to the pre-recording and delay part, it is recommended to extend the end time and ahead the starting time to playback. The recommended value: up to 10 minutes, at least 5 seconds.

Back

3.9.11 Control Playback Status NET_DVR_PlayBackControl_V40

Function: public boolean NET_DVR_PlayBackControl_V40(int lPlayHandle, int dwControlCode, byte[] lpInBuffer, int dwInLen, NET_DVR_PLAYBACK_INFO lpOutValue)

Parameter:

[in] lPlayHandle	Play handle, the returned value of NET_DVR_PlayBackByName or NET_DVR_PlayBackByTime.
[in] dwControlCode	Command to control playback status. See Table 3.13
[in] lpInBuffer	Pointer to input parameter. Set as NULL.
[in] dwInLen	Length of input parameter. Reserved. Set as NULL.
[out] lpOutBuffer	Pointer to output parameter. Reserved. Set as NULL.

Table 3.13 Playback Control Command

dwControlCode Macro Definition	Value	Implication
NET_DVR_PLAYSTART	1	Start
NET_DVR_PLAYPAUSE	3	Pause
NET_DVR_PLAYRESTART	4	Resume

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.9.12 Get Playback Process NET_DVR_GetPlayBackPos

Function: public int NET_DVR_GetPlayBackPos(int lPlayHandle)

Parameter:

[in] lPlayHandle	Playback handle. The returned value of NET_DVR_PlayBackByName or NET_DVR_PlayBackByTime.

Returned Value: Values 0 to 100 represent the process. 200 represents playback exception. Return -1 on failure. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. The process is the received data percentage. For example, the file total size is 100 M, and the received size is 10 M, then it will return 10. If decoding during callback, before stopping playback you need to call this API to make sure the process is 100, and then call the player API to make sure there is no data in decoding buffer. Only playback by file has prcess 0 to 100. Playback by time has process 0 and 100.

Back

3.9.13 Stop Playback NET_DVR_StopPlayBack

Function: public boolean NET_DVR_StopPlayBack(int lPlayHandle)

Parameter:

[in]lPlayHandle	Playback handle. The returned value of NET_DVR_PlayBackByName or NET_DVR_PlayBackByTime.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Download Video Files

3.9.14 Download Video Files by Name NET_DVR_GetFileByName

Function: public int NET_DVR_GetFileByName(int lUserID, java.lang.String sDVRFileName, java.lang.String sSavedFileName)

Parameter:

[in]lUserID	The returned value of NET_DVR_Login_V30.
[in]sDVRFileName	The video file name to download. File name should be less than 100 bytes.
[in]sSavedFileName	The files path after downloading to the computer. It should be absolute path.

Returned Value: Return -1 on failure and other values as the parameters of NET_DVR_StopGetFile. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Before downloading the video file, you can call video file searching API to get file name. This API specifies the file to be downloaded currently. After calling it successfully, it needs to call NET_DVR_PLAYSTART of NET_DVR_PlayBackControl to download the file.

Back

3.9.15 Download Video Files by Time NET_DVR_GetFileByTime

Function: public int NET_DVR_GetFileByTime(int lUserID, int lChannel, NET_DVR_TIME lpStartTime, NET_DVR_TIME lpStopTime, java.lang.String sSavedFileName)

Parameter:

[in]lUserID	The returned value of NET_DVR_Login_V30.
[in]lChannel	Channel No.
[in]lpStartTime	Start time. See: NET_DVR_TIME.
[in]lpStopTime	End time. See: NET_DVR_TIME.
[in]sSavedFileName	The files path after downloading to the computer. It should be

absolute path.

Returned Value: Return -1 on failure and other values as the parameters of NET_DVR_StopGetFile. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API specifies the file to be downloaded currently. After calling it successfully, it needs to call NET_DVR_PLAYSTART of NET_DVR_PlayBackControl to download the file.

If users save videos through this API, the maximum limit of the file is 1024 MB. If the file is more than 1024 MB, the SDK will save it into the new file automatically. The first 40 bytes will be written into the file automatically. File naming rule: add the digital identity basis on the file name (for example: * _1.mp4, * _2.mp4).

Back

3.9.16 Control Record Download Status NET_DVR_PlayBackControl_V40

Function: public boolean NET_DVR_PlayBackControl_V40(int lPlayHandle, int dwControlCode, byte[] lpInBuffer, int dwInLen, NET_DVR_PLAYBACK_INFO lpOutValue)

Parameter:

[in]lPlayHandle	Download handle. The returned value of NET_DVR_GetFileByName or NET_DVR_GetFileByTime.
[in]dwControlCode	The command of controlling playback status. See Table 3.14.
[in]lpInBuffer	Pointer to input parameter. Set as NULL.
[in]dwInLen	Length of input parameter.
[out]lpOutValue	Pointer to output parameter. Set as NULL.

Table 3.14 Download Control Command

dwControlCode Macro Definition	Value	Implication
NET_DVR_PLAYSTART	1	Start
NET_DVR_PLAYPAUSE	3	Pause
NET_DVR_PLAYRESTART	4	Resume

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.9.17 Get the Current Downloading Process NET_DVR_GetDownloadPos

Function: public int NET_DVR_GetDownloadPos(int lFileHandle);

Parameter: [in]lFileHandle Downloading handle. The returned value of NET_DVR_GetFileByName or NET_DVR_GetFileByTime.

Returned Value: Return -1 on failure, 0 to 100 as the downloading process, 100 as downloading finished, 200 as network exception. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Only playback by file has prcess 0 to 100. Playback by time has process 0 and 100.

Back

3.9.18 Stop Downloading Record File NET_DVR_StopGetFile

Function: public boolean NET_DVR_StopGetFile(int lFileHandle)

Parameter: [in]lFileHandle Downloading handle. The returned value of NET_DVR_GetFileByName or NET_DVR_GetFileByTime.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.10 PTZ Control

PTZ Control Operation

3.10.1 PTZ Control Operation (need to start live view) NET_DVR_PTZControl

Function: public boolean NET_DVR_PTZControl(int lRealHandle, int dwPTZCommand, int dwStop)

Parameter: [in] lRealHandle The returned value of NET_DVR_RealPlay_V40.

[in] dwPTZCommand PTZ control command. See Table 3.15.

[in] dwStop PTZ stop or start operation: 0-Start, 1-Stop.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every PTZ movement needs to call this API twice: start and stop control, decided by the last parameter (dwStop) in the API. It needs to start live view before calling this API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

By default, the PTZ turns around at the maximum speed.

Table 3.15 PTZ Control Command

wPTZCommand Macro Definition	Value	Implication
LIGHT_PWRON	2	Connect lighting power
WIPER_PWRON	3	Turn on wiper switch
FAN_PWRON	4	Turn on fan switch
HEATER_PWRON	5	Turn on heater switch
AUX_PWRON1	6	Turn on auxiliary device switch
AUX_PWRON2	7	Turn on auxiliary device switch
ZOOM_IN	11	Zoom in (Magnification enlarge)

ZOOM_OUT	12	Zoom out (Magnification decrease)
FOCUS_NEAR	13	Focus front
FOCUS_FAR	14	Focus back
IRIS_OPEN	15	Iris enlarge
IRIS_CLOSE	16	Iris narrow
TILT_UP	21	Tilt up
TILT_DOWN	22	Tilt down
PAN_LEFT	23	Pan left
PAN_RIGHT	24	Pan right
UP_LEFT	25	Tilt up and pan left
UP_RIGHT	26	Tilt up and pan right
DOWN_LEFT	27	Tilt down and pan left
DOWN_RIGHT	28	Tilt down and pan right
PAN_AUTO	29	PTZ scans left and right automatically

Back

3.10.2 PTZ Control Operation (no need to start live view)

NET_DVR_PTZControl_Other

Function: public boolean NET_DVR_PTZControl_Other(int lUserID, int lChannel, int dwPTZCommand, int dwStop)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] lChannel	Channel No.
[in] dwPTZCommand	PTZ control command. See Table 3.15.
[in] dwStop	PTZ stop or start operation: 0-Start, 1-Stop.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every PTZ movement needs to call this API twice: start and stop control, decided by the last parameter (dwStop) in the API. It needs to register device before calling this API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

By default, the PTZ turns around at the maximum speed.

Back

3.10.3 PTZ Control Operation with Speed (need to start live view)

NET_DVR_PTZControlWithSpeed

Function:	public boolean NET_DVR_PTZControlWithSpeed(int lRealHandle, int dwPTZCommand, int dwStop, int dwSpeed)	
Parameter:	[in] lRealHandle	The returned value of NET_DVR_RealPlay_V40.
	[in] dwPTZCommand	PTZ control command. See Table 3.15.
	[in] dwStop	PTZ stop or start operation: 0-start, 1-stop.
	[in] dwSpeed	PTZ control speed. Set based on different decoder speeds. Value range [1,7].
Returned Value:	In class com.hikvision.netsdk.HCNetSDK, JNI API. Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	

Back

3.10.4 PTZ Control Operation with Speed (no need to start live view)

NET_DVR_PTZControlWithSpeed_Other

Function:	public boolean NET_DVR_PTZControlWithSpeed_Other(int lUserID, int lChannel, int dwPTZCommand, int dwStop, int dwSpeed)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] lChannel	Channel No.
	[in] dwPTZCommand	PTZ control command. See Table 3.15.
	[in] dwStop	PTZ stop or start operation: 0-start, 1-stop.
	[in] dwSpeed	PTZ control speed. Set based on different decoder speeds. Value range [1,7].
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. Every PTZ movement needs to call this API twice: start and stop control, decided by the last parameter (dwStop) in the API. It doesn't need to start live view before calling this API, and you can control PTZ after login device. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.	

Back

PTZ Preset Operation

3.10.5 PTZ Preset Operation (need to start live view) NET_DVR_PTZPreset

Function: public boolean NET_DVR_PTZPreset(int lRealHandle, int dwPTZPresetCmd, int dwPresetIndex)

Parameter:
[in] lRealHandle The returned value of NET_DVR_RealPlay_V40.
[in] dwPTZPresetCmd The commands to control PTZ preset. See Table 3.16.
[in] dwPresetIndex Preset No. (Start from 1). It supports max 255 presets.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

Table 3.16 Preset Operation Command

dwPTZPresetCmd Macro Definition	Value	Implication
SET_PRESET	8	Set preset
CLE_PRESET	9	Clear preset
GOTO_PRESET	39	Go to preset

Back

3.10.6 PTZ Preset Operation NET_DVR_PTZPreset_Other

Function: public boolean NET_DVR_PTZPreset_Other(int lUserID, int lChannel, int dwPTZPresetCmd, int dwPresetIndex)

Parameter:
[in] lUserID The returned value of NET_DVR_Login_V30.
[in] lChannel Channel No.
[in] dwPTZPresetCmd The commands to control PTZ preset. See Table 3.16.
[in] dwPresetIndex Preset No. (Start from 1). It supports max 255 presets.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

PTZ control by NET_DVR_PTZPreset API: when device receives the control command, the PTZ will operate as the command. If operation failed, it will return FAULT; if operation normal, it will return TRUE.

PTZ control by NET_DVR_PTZPreset_Other API: it will return TRUE directly when the device

receives the control command.

Back

PTZ Patrol Operation

3.10.7 PTZ Control Operation (need to start live view) NET_DVR_PTZPCruise

Function: public boolean NET_DVR_PTZCruise(int lRealHandle, int dwPTZCruiseCmd, byte byCruiseRoute, byte byCruisePoint, short wInput)

Parameter:

[in] lRealHandle	The returned value of NET_DVR_RealPlay_V40.
[in] dwPTZCruiseCmd	PTZ patrol command. See Table 3.17.
[in] byCruiseRoute	Patrol route. Maximum 32 routes (number starts from 1).
[in] byCruisePoint	Patrol point. Maximum 32 points (number starts from 1).
[in] wInput	The value is different for different commands. Preset (max. 255), Dwell time (max. 255), Speed (max. 40).

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

Table 3.17 Patrol Operation Command

dwPTZCruiseCmd Macro Definition	Value	Implication
FILL_PRE_SEQ	30	Add preset to the patrol sequence
SET_SEQ_DWELL	31	Set dwell time of the patrol point
SET_SEQ_SPEED	32	Set patrol speed
CLE_PRE_SEQ	33	Delete preset point from the patrol sequence
RUN_SEQ	37	Start running the patrol
STOP_SEQ	38	Stop running the patrol

Back

3.10.8 PTZ Patrol Operation NET_DVR_PTZPCruise_Other

Function: public boolean NET_DVR_PTZCruise_Other(int lUserID, int lChannel, int dwPTZCruiseCmd, byte byCruiseRoute, byte byCruisePoint, short wInput)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] lChannel	Channel No.
[in] dwPTZCruiseCmd	PTZ patrol command. See Table 3.17.
[in] byCruiseRoute	Patrol route. Maximum 32 routes (number starts from 1).
[in] byCruisePoint	Patrol point. Maximum 32 points (number starts from 1).

[in] wInput The value is different for different commands. Preset (max. 255), Dwell time (max. 255), Speed (max. 40).

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

Back

PTZ Pattern Operation

3.10.9 PTZ Pattern Operation (need to start live view) NET_DVR_PTZTrack

Function: public boolean NET_DVR_PTZTrack(int lRealHandle, int dwPTZTrackCmd)

Parameter: [in] lRealHandle The returned value of NET_DVR_RealPlay_V40.

[in] dwPTZTrackCmd PTZ pattern command. See Table 3.18.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every operation command corresponds to the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

Table 3.18 Pattern Operation Command

dwPTZTrackCmd Macro Definition	Value	Implication
STA_MEM_CRUISE	34	Start recording pattern
STO_MEM_CRUISE	35	Stop recording pattern
RUN_CRUISE	36	Start running according to the pattern

Back

3.10.10 PTZ Pattern Operation bNET_DVR_PTZTrack_Other

Function: public boolean NET_DVR_PTZTrack_Other(int lUserID, int lChannel, int dwPTZTrackCmd)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[in] lChannel Channel No.

[in] dwPTZTrackCmd PTZ pattern command. See Table 3.18.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. Every operation command corresponds to

the control code between the device and the PTZ, and the device will send control code to PTZ based on the current decoder type and address. If decoder configuration of the current device doesn't match the PTZ device, it needs to re-configure the decoder parameter. If the PTZ doesn't support the parameter, it will not able to control PTZ.

Back

PTZ Area Zoom Control

3.10.11 PTZ Zoom In or Zoom Out NET_DVR_PTZSelZoomIn

Function: public boolean NET_DVR_PTZSelZoomIn(int lRealHandle, NET_DVR_POINT_FRAME pPointFrame)

Parameter:
[in] lRealHandle The returned value of NET_DVR_RealPlay_V40.
[in] pStruPointFrame PTZ image position. See: NET_DVR_POINT_FRAME.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API can realize 3D positioning if supported.

Now we suppose that the display box is 352*288, and set upper left point to be original point. Calculation method of the first four parameters is as below:

xTop = (the value of upper left coordinate of currently selected area) *255/352;

Zoom-out condition: xBottom - xTop > 2.

Zoom-in condition: xBottom - xTop > 0, and yBottom – yTop > 0.

Back

3.10.12 PTZ Zoom In or Zoom Out NET_DVR_PTZSelZoomIn_Ex

Function: public boolean NET_DVR_PTZSelZoomIn_EX(int lUserID, int lChannel, NET_DVR_POINT_FRAME pPointFrame)

Parameter:
[in] lUserID The returned value of NET_DVR_Login_V30.
[in] lChannel Channel No.
[in] pStruPointFrame PTZ image position. See: NET_DVR_POINT_FRAME.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API can realize 3D positioning if supported.

Now we suppose that the display box is 352*288, and set upper left point to be original point. Calculation method of the first four parameters is as below:

xTop = (the value of upper left coordinate of currently selected area) *255/352;

Zoom-out condition: xBottom - xTop > 2.

Zoom-in condition: xBottom - xTop > 0, and yBottom – yTop > 0.

Back

3.11 Audio Forwarding

3.11.1 Get Effective Audio Compression Parameter

NET_DVR_GetCurrentAudioCompress

Function:	public boolean NET_DVR_GetCurrentAudioCompress(int lUserID, NET_DVR_COMPRESSION_AUDIO lpCompressAudio)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[out] lpCompressAudio	Two-way audio compression parameter. See: NET_DVR_COMPRESSION_AUDIO
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

3.11.2 Eanbel Audio Forwarding, Get Encoded Audio Data

NET_DVR_StartVoiceCom_MR_V30

Function:	public int NET_DVR_StartVoiceCom_MR_V30(int lUserID, int lVoiceChan, VoiceDataCallBack CallBack)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] lVoiceChan	Audio channel No. (starting from 1).
	[in] CallBack	Audio data callback function. Get the encoded audio data and call audio library to decode.

```
public interface VoiceDataCallBack {
    public void fVoiceDataCallBack(int lVoiceComHandle, byte[] pDataBuffer, int iDataSize, int iAudioFlag);
}
```

	[out] lVoiceComHandle	The returned value of NET_DVR_StartVoiceCom_MR_V30.
	[out] pDataBuffer	Buffer pointer saving audio data.
	[out] iDataSize	Audio data size.
	[out] iAudioFlag	Audio data type: 1-Audio data sent by device
Returned Value:	Return -1 on failure, other values as the handle parameters of NET_DVR_VoiceComSendData or NET_DVR_StopVoiceCom. Call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. Before calling starting audio forwarding, get the audio encoding type via NET_DVR_GetCurrentAudioCompress.	

Back

3.11.3 Forward Audio Data NET_DVR_VoiceComSendData

Function: public boolean NET_DVR_VoiceComSendData(int lVoiceComHandle, byte[] pSendBuf, int lBufSize)

Parameter:

[in] lVoiceComHandle	The returned value of NET_DVR_StartVoiceCom_MR_V30.
[in] pSendBuf	The buffer saving audio data.
[in] dwBufSize	Audio data size.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. This API can forward the encoded audio data to the device. For G722 audio encoding type, the data size per forwarding is 80 bytes. For G711 audio encoding type, the data size per forwarding is 160 bytes.

Back

3.11.4 Stop Audio Forward NET_DVR_StopVoiceCom

Function: public boolean NET_DVR_StopVoiceCom(int lVoiceComHandle)

Parameter:

[in]lVoiceComHandle	The returned value of NET_DVR_StartVoiceCom_MR_V30.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.12 Transparent Transmitting

Transparent Channel

3.12.1 Bulid Transparent Channel NET_DVR_SerialStart_V40

Function: public int NET_DVR_SerialStart_V40(int lUserID, NET_DVR_SERIAL_COND serialCond, SerialDataCallBackV40 fSerialDataCallBackV40)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] serialCond	Serial port parameter. See: NET_DVR_SERIAL_COND.
[in] fSerialDataCallBackV40	Callback function of transparent channel data

```
public interface SerialDataCallBackV40{
    public void fSerialDataCallBackV40(int lSerialHandle, int lChannel, byte[] pDataBuffer, int iDataSize);
}
```

[out] lSerialHandle	The returned value of NET_DVR_SerialStart_V40.
[out] lChannel	Serial port No.
[out] pDataBuffer	The buffer point saving data

	[out] iDataSize	Data size
Returned Value:	Return -1 on failure, other values as the handle parameters of NET_DVR_SerialSend. Call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. It is required to get data decoder from callback function, and it must support data copying pack. Otherwise the callback function cannot return anything even it sends successfully.	

Back

3.12.2 Send Data to Serial Port via Transparent Channel NET_DVR_SerialSend

Function:	public boolean NET_DVR_SerialSend(int lSerialHandle, int lChannel, byte[] lpSendBuf, int dwBufSize)	
Parameter:	[in] lSerialHandle	The returned value of NET_DVR_SerialStart_V40.
	[in] lChannel	Valid when using RS485 serial port, starting from 1. Set it as 0 when using RS232 serial port.
	[in] lpSendBuf	Buffer point sending data.
	[in] dwBufSize	Buffer size. Maxmum 1016 bytes.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

3.12.3 Disconnect Transparent Channel NET_DVR_SerialStop

Function:	public boolean NET_DVR_SerialStop(int lSerialHandle)	
Parameter:	[in]lSerialHandle	The returned value of NET_DVR_SerialStart_V40.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

Send Data to Serial Port

3.12.4 Send Data to Serial Port Without Transparent Channel NET_DVR_SendToSerialPort

Function:	public boolean NET_DVR_SendToSerialPort(int lUserID, int dwSerialPort, int dwSerialIndex, byte[] lpSendBuf, int dwBufSize)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] dwSerialPort	Serial port type: 1-232, 2-485.
	[in] dwSerialIndex	Serial port index. Start from 1.

	[in] lpSendBuf	The buffer point sending data.
	[in] dwBufSize	Buffer size. Maximum 1016 bytes.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

3.12.5 Send Data to 232 Serial Port Without Transparent Channel

NET_DVR_SendTo232Port

Function:	public boolean NET_DVR_SendTo232Port(int lUserID, byte[] lpSendBuf, int dwBufSize)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] lpSendBuf	The buffer point sending data.
	[in] dwBufSize	Buffer size. Maximum 1016 bytes.
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API.	

Back

3.13 Manual Recording

3.13.1 Remotely Start Manual Recording NET_DVR_StartDVRRecord

Function:	public boolean NET_DVR_StartDVRRecord(int lUserID, int lChannel, int lRecordType)	
Parameter:	[in] lUserID	The returned value of NET_DVR_Login_V30.
	[in] lChannel	Channel number. 0x00ff-All analog channels. 0xff00-All digital channel. 0xffff-All analog and digital channel.
	[in] lRecordType	Recording type: 0-Manual, 1-Alarm, 2-Copy back, 3-Signal, 4-Motion, 5-Tampering
Returned Value:	Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.	
Note:	In class com.hikvision.netsdk.HCNetSDK, JNI API. The device should support the selected recording type. If not support, it is manual recording by default. When the channel has enabled scheduled recording, if you try to enable manual recording for the first time, you will find this operation failed, it is still in scheduled recording status, and the device status is recording status if you call NET_DVR_WORKSTATE_V30 of NET_DVR_GetDVRWorkState_V30. Now disable manual recording and scheduled recording, the device status is not in recording status. Enable manual recording again (for the second time), and this time manual recording starts. Disable manual recording and reboot device, the scheduled recording will be enabled again.	

Back

3.13.2 Remotely Stop Manual Recording NET_DVR_StopDVRRecord

Function: public boolean NET_DVR_StopDVRRecord(int lUserID, int lChannel)

Parameter: [in]lUserID The returned value of NET_DVR_Login_V30.

[in]lChannel Channel number. 0x00ff-All analog channels. 0xff00-All digital channels. 0xffff-All analog and digital channels.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.14 Remote Panel Control

3.14.1 Remotely Control the Buttons on the Panel NET_DVR_ClickKey

Function: public boolean NET_DVR_ClickKey(int lUserID, int lKeyIndex)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[in] lKeyIndex Buttons on the panel. See Table 3.19.

Table 3.19 Buttons on the Panel

lKeyIndex Macro Definition	Value	Implication
KEY_CODE_MENU	12	MENU
KEY_CODE_ENTER	13	ENTER
KEY_CODE_CANCEL	14	ESCS
KEY_CODE_UP/KEY_PTZ_UP_START	15	Up or PTZ Tile Up
KEY_CODE_DOWN/KEY_PTZ_DOWN_START	16	Down or PTZ Tile Down
KEY_CODE_LEFT/KEY_PTZ_LEFT_START	17	Left or PTZ Pan Left
KEY_CODE_RIGHT/KEY_PTZ_RIGHT_START	18	Right or PTZ Pan Right

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.15 HDD Management

3.15.1 Remotely Format HDD NET_DVR_FormatDisk

Function: public int NET_DVR_FormatDisk(int lUserID, int lDiskNumber)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[in] lDiskNumber HDD No., starting from 0. 0xff means valid to all HDDs (exclude

read-only HDD).

Returned Value: Return -1 on failure and other value as parameters of NET_DVR_CloseFormatHandle. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. If network breaks down during formatting, the device will continue formatting, but the client can't receive the status.

Back

3.15.2 Get Formatting Process NET_DVR_GetFormatProgress

Function: public boolean NET_DVR_GetFormatProgress(int lFormatHandle, INT_PTR pCurrentFormatDisk, INT_PTR pCurrentDiskPos, INT_PTR pFormatStatic)

Parameter:

[in] lFormatHandle	Formatting handle. The returned value of NET_DVR_FormatDisk.
[out] pCurrentFormatDisk	The pointer of the HDD which is formatted currently. HDD number begins from 0, and -1 is the initial status.
[out] pCurrentDiskPos	The pointer of formatting process of current HDD and the process ranges from 0 to 100.
[out] FormatStatic	The pointer of HDD formatting status: 0-Formatting; 1-Finished; 2-Formatting error. Formatting is stopped. This issue may occur in both local and network disk; 3-Network exception which results in the loss of network disk, and the HDD can't be formatted.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.15.3 Close HDD Formatting Handle and Release Resource NET_DVR_CloseFormatHandle

Function: public boolean NET_DVR_CloseFormatHandle(int lFormatHandle)

Parameter: [in] lFormatHandle Formatting handle. The returned value of NET_DVR_FormatDisk.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.16 Device Maintenance Management

Device Working Status

3.16.1 Get Device Working Status NET_DVR_GetDVRWorkState_V30

Function: public boolean NET_DVR_GetDVRWorkState_V30(int lUserID, NET_DVR_WORKSTATE_V30 lpWorkState)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[out] lpWorkState Device working status parameter. See: NET_DVR_WORKSTATE_V30.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

UPNP Port Maping Status

3.16.2 Get UPNP Port Maping Status NET_DVR_GetUpnpNatState

Function: public boolean NET_DVR_GetUpnpNatState(int lUserID, NET_DVR_UPNP_NAT_STATE lpState)

Parameter: [in] lUserID The returned value of NET_DVR_Login_V30.

[out] lpState UPNP port mapping status. See: NET_DVR_UPNP_NAT_STATE.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Remote Upgrade

3.16.3 Set Network Environment during Remote Upgrading

NET_DVR_SetNetworkEnvironment

Function: public boolean NET_DVR_SetNetworkEnvironment(int dwEnvironmentLevel)

Parameter: [in] dwEnvironmentLevel Network environment level.

```
enum{
    LOCAL_AREA_NETWORK = 0,// local area network environment
    WIDE_AREA_NETWORK      // wide area network environment
}
```

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API. There're two network environment levels:

- LOCAL_AREA_NETWORK indicates local area network environment (good network, and smooth communication).
- WIDE_AREA_NETWORK indicates wide area network environment (poor network, and easy to be blocked).

Before calling NET_DVR_Upgrade, call this API to adapt to different upgrading environment.

Back

3.16.4 Remote Upgrade NET_DVR_Upgrade

Function: public int NET_DVR_Upgrade(int lUserID, String sFileName)

Parameter:

[in] lUserID	The returned value of NET_DVR_Login_V30.
[in] sFileName	Upgrading file path (including file name). Path length is related to the operation system.

Returned Value: Return -1 on failure and other values as parameters of NET_DVR_GetUpgradeState. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.16.5 Get Remote Upgrade Process NET_DVR_GetUpgradeProgress

Function: public int NET_DVR_GetUpgradeProgress(int lUpgradeHandle);

Parameter:

[in] lUpgradeHandle	The returned value of NET_DVR_Upgrade.

Returned Value: Return -1 on failure, 0 to 100 as the upgrading process. Call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.16.6 Get Remote Upgrade Status NET_DVR_GetUpgradeState

Function: public int NET_DVR_GetUpgradeState(int lUpgradeHandle);

Parameter:

[in] lUpgradeHandle	The returned value of NET_DVR_Upgrade.

Returned Value: -1-Failed to call this API. Other values: 1-succeeded; 2-upgrading; 3-upgrade failed; 4-network disconnect and the status is unknown; 5-language version not match. If returned -1, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.16.7 Get Remote Upgrade Step Information NET_DVR_GetUpgradeStep

Function: public int NET_DVR_GetUpgradeStep(int lUpgradeHandle, INT_PTR pSubProgress)

Parameter: [in] lUpgradeHandle The returned value of NET_DVR_Upgrade.

[out] pSubProgress Sub process of upgrade steps.

Returned Value: Return -1 on failure and other values are defined in Table 3.20.

Table 3.20 Upgrading Step Information

Value	Implication
1	Receive upgrade package data
2	Upgrade system
3	Backup system
255	Searching upgrade file

If returned -1, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

3.16.8 Close Remote Upgrade Handle and Release Resource NET_DVR_CloseUpgradeHandle

Function: public boolean NET_DVR_CloseUpgradeHandle(int lUpgradeHandle);

Parameter: [in] lUpgradeHandle The returned value of NET_DVR_Upgrade.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

Remote Reboot

3.16.9 Reboot Device NET_DVR_RebootDVR

Function: public boolean NET_DVR_RebootDVR(int lUserID)

Parameter: [in]lUserID The returned value of NET_DVR_Login_V30.

Returned Value: Return TRUE on success, FALSE on failure. If it returns FALSE, call NET_DVR_GetLastError to get the error code and find the reason.

Note: In class com.hikvision.netsdk.HCNetSDK, JNI API.

Back

4 Error Code Definition

4.1 Error Code of Network Communication Library

Error	Value	Implication
NET_DVR_NOERROR	0	No error.
NET_DVR_PASSWORD_ERROR	1	Wrong username or password.
NET_DVR_NOENOUGHPRI	2	Not permission. The registered user has no permission for the operation. Compare with the remote user parameter configuration.
NET_DVR_NOINIT	3	SDK is uninitialized.
NET_DVR_CHANNEL_ERROR	4	Incorrect channel No., no corresponding channel on the device.
NET_DVR_OVER_MAXLINK	5	No more users allowed.
NET_DVR_VERSIONNOMATCH	6	The SDK version and device do not match.
NET_DVR_NETWORK_FAIL_CONNECT	7	Connection failed: device offline or connection timeout.
NET_DVR_NETWORK_SEND_ERROR	8	Failed to send data to the device.
NET_DVR_NETWORK_RECV_ERROR	9	Failed to receive data from the device.
NET_DVR_NETWORK_RECV_TIMEOUT	10	Receiving data from server is timeout.
NET_DVR_NETWORK_ERRORDATA	11	Data error. The data sent to or received from the device is incorrect. E.g., input value not supported when remote parameter configuration.
NET_DVR_ORDER_ERROR	12	Order error.
NET_DVR_OPERNOPERMIT	13	No permission.
NET_DVR_COMMANDTIMEOUT	14	Device execution timeout.
NET_DVR_ERRORSERIALPORT	15	Serial port error. The specified serial port does not exist.
NET_DVR_ERRORALARMPORT	16	Alarm port error. The specified alarm port does not exist.
NET_DVR_PARAMETER_ERROR	17	Incorrect parameter. The input or output parameter in SDK APIs is NULL.
NET_DVR_CHAN_EXCEPTION	18	Abnormal channel status.
NET_DVR_NODISK	19	No HDD. Failed to operate the record files, HDD, etc.
NET_DVR_ERRORDISKNUM	20	Incorrect HDD number. The specified HDD No. does not exist.
NET_DVR_DISK_FULL	21	HDD full.
NET_DVR_DISK_ERROR	22	HDD error.
NET_DVR_NOSUPPORT	23	Not supported.
NET_DVR_BUSY	24	Device is busy.
NET_DVR_MODIFY_FAIL	25	Failed to modify device.
NET_DVR_PASSWORD_FORMAT_ERROR	26	Incorrect password format
NET_DVR_DISK_FORMATING	27	Formatting HDD. Operation failed.
NET_DVR_DVRNORESOURCE	28	Inadequate resources.
NET_DVR_DVROPRATEFAILED	29	Operation failed.
NET_DVR_OPENHOSTSOUND_FAIL	30	Failed to sample local audio or enable audio output during two-way audio or broadcast.

NET_DVR_DVRVOICEOPENED	31	Two-way audio occupied.
NET_DVR_TIMEINPUTERROR	32	Invalid time.
NET_DVR_NOSPECFILE	33	Cannot find the specified files.
NET_DVR_CREATEFILE_ERROR	34	Failed to create file during local recording, picture saving, configuration file acquiring or remote video downloading.
NET_DVR_FILEOPENFAIL	35	Failed to open file while setting configuration file, upgrading device or uploading trial file.
NET_DVR_OPERNOTFINISH	36	Last operation has not completed yet.
NET_DVR_GETPLAYTIMEFAIL	37	Failed to get the current play time.
NET_DVR_PLAYFAIL	38	Failed to play the record.
NET_DVR_FILEFORMAT_ERROR	39	Incorrect file format.
NET_DVR_DIR_ERROR	40	Incorrect path.
NET_DVR_ALLOC_RESOURCE_ERROR	41	SDK resource allocation error.
NET_DVR_AUDIO_MODE_ERROR	42	Incorrect sound card mode; the current sound playing mode and the actual configured mode don't match.
NET_DVR_NOENOUGH_BUF	43	Inadequate buffer which receives data or saves pictures.
NET_DVR_CREATESOCKET_ERROR	44	Failed to create SOCKET.
NET_DVR_SETSOCKET_ERROR	45	Failed to set SOCKET.
NET_DVR_MAX_NUM	46	No more devices can be connected. The registered and connected devices have reached the maximum number supporting by SDK.
NET_DVR_USERNOTEXIST	47	User does not exist.
NET_DVR_WRITEFLASHERROR	48	Writing FLASH error during upgrading.
NET_DVR_UPGRADEFAIL	49	Upgrade failed due to network disconnection or language mismatch.
NET_DVR_CARDHAVEINIT	50	The decoding card has been initialized.
NET_DVR_PLAYERFAILED	51	Failed to call a function from the record library.
NET_DVR_MAX_USERNUM	52	No more users allowed.
NET_DVR_GETLOCALIPANDMACFAIL	53	Failed to get local IP or MAC address.
NET_DVR_NOENCODEING	54	This camera has not started encoding yet.
NET_DVR_IPMISMATCH	55	IP address mismatch.
NET_DVR_MACMISMATCH	56	MAC address mismatch.
NET_DVR_UPGRADELANGMISMATCH	57	Upgrade file language mismatch.
NET_DVR_MAX_PLAYERPORT	58	No more player channels allowed.
NET_DVR_NOSPACEBACKUP	59	No enough space for backup in the backup device.
NET_DVR_NODEVICEBACKUP	60	No backup device found.
NET_DVR_PICTURE_BITS_ERROR	61	Incorrect pixel bits. (Max.: 24)
NET_DVR_PICTURE_DIMENSION_ERROR	62	Incorrect height or width. (Max.: 128*256)
NET_DVR_PICTURE_SIZ_ERROR	63	Picture size should be less than 100 KB.
NET_DVR_LOADPLAYERSDKFAILED	64	Failed to load the Player SDK to the current directory.
NET_DVR_LOADPLAYERSDKPROC_ERROR	65	Cannot find the function entry in the player SDK.
NET_DVR_LOADDSSDKFAILED	66	Failed to load the DsSDK to the correct directory.
NET_DVR_LOADDSSDKPROC_ERROR	67	Cannot find the function entry in the DsSDK.
NET_DVR_DSSDK_ERROR	68	Failed to call the function from the hardware decoding library

		DsSDK.
NET_DVR_VOICEMONOPOLIZE	69	Sound card exclusively occupied.
NET_DVR_JOINMULTICASTFAILED	70	Failed to add the device to multicast group.
NET_DVR_CREATEDIR_ERROR	71	Failed to create log file directory.
NET_DVR_BINDSOCKET_ERROR	72	Failed to bind the socket.
NET_DVR_SOCKETCLOSE_ERROR	73	Socket disconnected due to connection failure or destination unreachable.
NET_DVR_USERID_ISUSING	74	The user ID is under operation.
NET_DVR_SOCKETLISTEN_ERROR	75	Listening failed.
NET_DVR_PROGRAM_EXCEPTION	76	Program exception.
NET_DVR_WRITEFILE_FAILED	77	Failed to write file during local recording or remote video or picture downloading.
NET_DVR_FORMAT_READONLY	78	Read-only HDD cannot be formatted.
NET_DVR_WITHSAMEUSERNAME	79	There exists same user name in the remote user configuration structs.
NET_DVR_DEVICETYPE_ERROR	80	Device model does not match when importing parameters.
NET_DVR_LANGUAGE_ERROR	81	Language mismatches while importing parameters.
NET_DVR_PARAVERSION_ERROR	82	Software version mismatches when importing parameters.
NET_DVR_IPCHAN_NOTALIVE	83	IP camera is offline during live view.
NET_DVR_RTSP_SDK_ERROR	84	Failed to load standard protocol communication library StreamTransClient.
NET_DVR_CONVERT_SDK_ERROR	85	Failed to load encapsulation transformation library.
NET_DVR_IPC_COUNT_OVERFLOW	86	No more IP cameras can be added.
NET_DVR_MAX_ADD_NUM	87	No more record tags can be added.
NET_DVR_PARAMMODE_ERROR	88	Image intensifier, parameter mode error. This error may occur when client sets software or hardware parameters.
NET_DVR_CODESPITTER_OFFLINE	89	Codesplitter is offline.
NET_DVR_BACKUP_COPYING	90	Backing up.
NET_DVR_CHAN_NOTSUPPORT	91	Channel not support.
NET_DVR_CALLINEINVALID	92	Vertical line too concentrated or horizontal line less inclined.
NET_DVR_CALCANCELCONFLICT	93	If rule and global actual size filter has been set, cancel the calibration confliction.
NET_DVR_CALPOINTOUTRANGE	94	Calibration points are out of range.
NET_DVR_FILTERRECTINVALID	95	Size filter does not meet the requirements.
NET_DVR_DDNS_DEVOFFLINE	96	The device is not registered in the DDNS.
NET_DVR_DDNS_INTER_ERROR	97	DDNS server internal error.
NET_DVR_ALIAS_DUPLICATE	150	Nickname of EasyDDNS has been used.
NET_DVR_DEV_NET_OVERFLOW	800	Exceed the max. network traffic.
NET_DVR_STATUS_RECORDFILE_WRITING_NOT_LOCK	801	Record file is under writing and cannot be locked.
NET_DVR_STATUS_CANT_FORMAT_LITTLE_DISK	802	The hard disk capacity is too small and cannot be formatted.
Capture Camera Error Code		
NET_DVR_ERR_LANENUM_EXCEED	1400	No more lanes can be added.
NET_DVR_ERR_PRAREA_EXCEED	1401	The license plate recognization area is too large.
NET_DVR_ERR_LIGHT_PARAM	1402	Traffic light access parameter error.

NET_DVR_ERR_LANE_LINE_INVALID	1403	Lane configuration error.
NET_DVR_ERR_STOP_LINE_INVALID	1404	Stop line configuration error.
NET_DVR_ERR_LEFTORRIGHT_LINE_INVALID	1405	Turning left/right boundary configuration error.
NET_DVR_ERR_LANE_NO_REPEAT	1406	Overlay lane No. duplicated.
NET_DVR_ERR_PRAREA_INVALID	1407	License plate polygon invalid.
NET_DVR_ERR_LIGHT_NUM_EXCEED	1408	The number of video analysis traffic light reaches the limit.
NET_DVR_ERR_SUBLIGHT_NUM_INVALID	1409	Invalid sub light number of video analysis traffic light
NET_DVR_ERR_LIGHT_AREASIZE_INVALID	1410	Invalid frame size of traffic light
NET_DVR_ERR_LIGHT_COLOR_INVALID	1411	Invalid color of traffic light
NET_DVR_ERR_LIGHT_DIRECTION_INVALID	1412	Invalid direction of traffic light
NET_DVR_ERR_LACK_IOABLITY	1413	Inadequate ability supported by IO port.
NET_DVR_ERR_FTP_PORT	1414	Invalid FTP port (duplicated or exceptional)
NET_DVR_ERR_FTP_CATALOGUE	1415	Invalid FTP directory name (enable muiti-level directory, and the pass value is NULL)
NET_DVR_ERR_FTP_UPLOAD_TYPE	1416	Invalid FTP uploading type (for single FTP, it supports all types.For dual FTP, it only supports monitor point and violation.)
NET_DVR_ERR_FLASH_PARAM_WRITE	1417	Write FLASH failed when configuring parameters.
NET_DVR_ERR_FLASH_PARAM_READ	1418	Read FLASH failed when configuring parameters.
NET_DVR_ERR_PICNAME_DELIMITER	1419	Invalid separator of FTP image name
NET_DVR_ERR_PICNAME_ITEM	1420	Invalid FTP image name (e.g., separator)
NET_DVR_ERR_PLATE_RECOGNIZE_TYPE	1421	Invalid plate recognization area type (rectangle and polygon validation)
NET_DVR_ERR_CAPTURE_TIMES	1422	Invalid capture times (valid: 0 to 5)
NET_DVR_ERR_LOOP_DISTANCE	1423	Invalid coil distance (valid: 0 to 2000ms)
NET_DVR_ERR_LOOP_INPUT_STATUS	1424	Invalid coil input status
NET_DVR_ERR_RELATE_IO_CONFLICT	1425	Related IO conflict.
NET_DVR_ERR_INTERVAL_TIME	1426	Invalid continuous capture interval (valid: 0 to 6000ms)
NET_DVR_ERR_SIGN_SPEED	1427	Invalid sign speed limit value
NET_DVR_ERR_PIC_FLIP	1428	Image flip
NET_DVR_ERR_RELATE_LANE_NUMBER	1429	Related lane number error (duplicated. Valid: 1 to 99).
NET_DVR_ERR_TRIGGER_MODE	1430	Invalid configuration of triggering mode
NET_DVR_ERR_DELAY_TIME	1431	Triggering delay time error (2000ms)
NET_DVR_ERR_EXCEED_RS485_COUNT	1432	Exceed the RS485 limit.
NET_DVR_ERR_RADAR_TYPE	1433	Radar type error.
NET_DVR_ERR_RADAR_ANGLE	1434	Radar angle error.
NET_DVR_ERR_RADAR_SPEED_VALID_TIME	1435	Radar effective time error.
NET_DVR_ERR_RADAR_LINE_CORRECT	1436	Radar line correction parameter error.
NET_DVR_ERR_RADAR_CONST_CORRECT	1437	Radar constant correction parameter error.
NET_DVR_ERR_RECORD_PARAM	1438	Invalid recording parameter (pre-recording time≤10s)
NET_DVR_ERR_LIGHT_WITHOUT_COLOR_AND_DIRECTION	1439	Configured traffic light number without direction and color.
NET_DVR_ERR_LIGHT_WITHOUT_DETECTION_REGION	1440	Configured traffic light number without detection area.
NET_DVR_ERR_RECOGNIZE_PROVINCE_PARAM	1441	The validity of province parameter on the plate.

NET_DVR_ERR_SPEED_TIMEOUT	1442	Invalid timeout of IO velocity measurement (valid value: larger than 0).
NET_DVR_ERR_NTP_TIMEZONE	1443	NTP timezone parameter error.
NET_DVR_ERR_NTP_INTERVAL_TIME	1444	NTP synchronization interval error.
NET_DVR_ERR_NETWORK_CARD_NUM	1445	Configurable NIC number error.
NET_DVR_ERR_DEFAULT_ROUTE	1446	Default route error.
NET_DVR_ERR_BONDING_WORK_MODE	1447	Bonding NIC working mode error.
NET_DVR_ERR_SLAVE_CARD	1448	Slave NIC error.
NET_DVR_ERR_PRIMARY_CARD	1449	Primary NIC error.
NET_DVR_ERR_DHCP_PPOE_WORK	1450	DHCP and PPPOE cannot be enabled at the same time.
NET_DVR_ERR_NET_INTERFACE	1451	Network interface error.
NET_DVR_ERR_MTU	1452	MTU error.
NET_DVR_ERR_NETMASK	1453	Incorrect subnet mask.
NET_DVR_ERR_IP_INVALID	1454	Invalid IP address.
NET_DVR_ERR_MULTICAST_IP_INVALID	1455	Invalid multicast address.
NET_DVR_ERR_GATEWAY_INVALID	1456	Invalid gateway.
NET_DVR_ERR_DNS_INVALID	1457	Invalid DNS.
NET_DVR_ERR_ALARMHOST_IP_INVALID	1458	Invalid security control panel address.
NET_DVR_ERR_IP_CONFLICT	1459	IP conflict.
NET_DVR_ERR_NETWORK_SEGMENT	1460	IP doesn’t support the same network segment.
NET_DVR_ERR_NETPORT	1461	Incorrect port.
NET_DVR_ERR_PPPOE_NOSUPPORT	1462	PPPOE not supported.
NET_DVR_ERR_DOMAINNAME_NOSUPPORT	1463	Domain name not supported.
NET_DVR_ERR_NO_SPEED	1464	Velocity measurement disabled.
NET_DVR_ERR_IOSTATUS_INVALID	1465	IO status error.
NET_DVR_ERR_BURST_INTERVAL_INVALID	1466	Invalid continuous capture interval.
NET_DVR_ERR_RESERVE_MODE	1467	Backup mode error.
NET_DVR_ERR_LANE_NO	1468	Overlay lane No. error.
NET_DVR_ERR_COIL_AREA_TYPE	1469	Incorrect coil area type.
NET_DVR_ERR_TRIGGER_AREA_PARAM	1470	Incorrect triggering area parameter.
NET_DVR_ERR_SPEED_LIMIT_PARAM	1471	Incorrect violation and speed limit parameter.
NET_DVR_ERR_LANE_PROTOCOL_TYPE	1472	Incorrect line related protocol typt.
NET_DVR_ERR_INTERVAL_TYPE	1473	Invalid continuous capture interval type.
NET_DVR_ERR_INTERVAL_DISTANCE	1474	Invalid distance of continuous capture interval.
NET_DVR_ERR_RS485_ASSOCIATE_DEVTYPE	1475	Invalid RS485 related type.
NET_DVR_ERR_RS485_ASSOCIATE_LANENO	1476	Invalid RS485 related lane No.
NET_DVR_ERR_LANENO_ASSOCIATE_MULTIRS485	1477	Multiple RS485 related to lane No.
NET_DVR_ERR_LIGHT_DETECTION_REGION	1478	Configured traffic light number, and the detected area’s width or height is 0.
NET_DVR_ERR_DN2D_NOSUPPORT	1479	Capture frame 2D noise reduction unsupported.
NET_DVR_ERR_IRISMODE_NOSUPPORT	1480	Lens type unsupported.
NET_DVR_ERR_WB_NOSUPPORT	1481	White balance mode unsupported.
NET_DVR_ERR_IO_EFFECTIVENESS	1482	IO port effectiveness.
NET_DVR_ERR_LIGHTNO_MAX	1483	The traffic light exceeds the limit of red/yellow light (16).

NET_DVR_ERR_LIGHTNO_CONFLICT	1484	The traffic red/yellow light conflicted.
NET_DVR_ERR_CANCEL_LINE	1485	The triggering line.
NET_DVR_ERR_STOP_LINE	1486	The stop line of waiting area.
NET_DVR_ERR_RUSH_REDLIGHT_LINE	1487	The triggering line of running the red light.
NET_DVR_ERR_IOOUTNO_MAX	1488	IO output port No. exceeds the limit.
NET_DVR_ERR_IOOUTNO_AHEADTIME_MAX	1489	IO output port aheading time exceeds the limit.
NET_DVR_ERR_IOOUTNO_IOWORKTIME	1490	IO output port valid working time exceeds the limit.
NET_DVR_ERR_IOOUTNO_FREQMULTI	1491	Incorrect frequency multiplication of IO output port in pulse mode.
NET_DVR_ERR_IOOUTNO_DUTYRATE	1492	Incorrect duty ratio of IO output port in pulse mode.
NET_DVR_ERR_VIDEO_WITH_EXPOSURE	1493	Valid with exposure. Video unsupported.
NET_DVR_ERR_PLATE_BRIGHTNESS_WITHOUT_FLASH DET	1494	The auto flash only valid in plate brightness compensation mode.
NET_DVR_ERR_RECOGNIZE_TYPE_PARAM	1495	Invalid recognization type.
NET_DVR_ERR_PALTE_RECOGNIZE_AREA_PARAM	1496	Invalid plate recognization parameter. Error in plate recognization area configuration.
NET_DVR_ERR_PORT_CONFLICT	1497	Port conflict.
NET_DVR_ERR_LOOP_IP	1498	IP cannot be set as loop address.
NET_DVR_ERR_DRIVELINE_SENSITIVE	1499	Incorrect sensitive of running on the lane line (in video alarm mode)
NET_DVR_ERR_EXCEED_MAX_CAPTURE_TIMES	1600	The maximum number is 2 pictures when the capture modeis stroboscopic
NET_DVR_ERR_REDAR_TYPE_CONFLICT	1601	Radar type conflict between the same RS485 port.

4.2 Error Code of RTSP Communication Library

Error	Value	Implication
NET_DVR_RTSP_GETPORTFAILED	407	Failed to get RTSP port.
NET_DVR_RTSP_DESCRIBESENDTIMEOUT	411	Send RTSP DECRIBE timeout.
NET_DVR_RTSP_DESCRIBESENDERROR	412	Failed to send RTSP DECRIBE.
NET_DVR_RTSP_DESCRIBERECVTIMEOUT	413	Receive RTSP DECRIBE timeout.
NET_DVR_RTSP_DESCRIBERECVDATALOST	414	Incorrect received data of RTSP DECRIBE.
NET_DVR_RTSP_DESCRIBERECVERROR	415	Failed to receive RTSP DECRIBE.
NET_DVR_RTSP_DESCRIBESERVERERR	416	Error status returned by RTSP DECRIBE server. E.g., 401,501.
NET_DVR_RTSP_SETUPSENDTIMEOUT	421	Send RTSP SETUP timeout.
NET_DVR_RTSP_SETUPSENDERROR	422	Failed to send RTSP SETUP.
NET_DVR_RTSP_SETUPRECVTIMEOUT	423	Receive RTSP SETUP timeout
NET_DVR_RTSP_SETUPRECVDATALOST	424	Incorrect received data of RTSP SETUP
NET_DVR_RTSP_SETUPRECVERROR	425	Failed to receive RTSP SETUP.
NET_DVR_RTSP_OVER_MAX_CHAN	426	No more servers can be connected or inadequate server resource. Return this code when server returns 453.
NET_DVR_RTSP_PLAYSENDTIMEOUT	431	Send RTSP PLAY timeout.

NET_DVR_RTSP_PLAYSENDERROR	432	Failed to send RTSP PLAY.
NET_DVR_RTSP_PLAYRECVTIMEOUT	433	Receive RTSP PLAT timeout.
NET_DVR_RTSP_PLAYRECVDATALOST	434	Incorrect received data of RTSP PLAY.
NET_DVR_RTSP_PLAYRECVERROR	435	Failed to receive RTSP PLAY.
NET_DVR_RTSP_PLAYSERVERERR	436	RTSP PLAY server returns error status.
NET_DVR_RTSP_TEARDOWNSENDTIMEOUT	441	Send RTSP TEARDOWN timeout.
NET_DVR_RTSP_TEARDOWNSENDERROR	442	Failed to send RTSP TEARDOWN.
NET_DVR_RTSP_TEARDOWNRECVTIMEOUT	443	Receive RTSP TEARDOWN timeout.
NET_DVR_RTSP_TEARDOWNRECVDATALOST	444	Incorrect received data of RTSP TEARDOWN.
NET_DVR_RTSP_TEARDOWNRECVERROR	445	Failed to receive RTSP TEARDOWN.
NET_DVR_RTSP_TEARDOWNSERVERERR	446	RTSP TEARDOWN server returns error status.

4.3 Error Code of Software Decoding Library

Error	Value	Implication
NET_PLAYM4_NOERROR	500	No error.
NET_PLAYM4_PARA_OVER	501	Invalid input parameter.
NET_PLAYM4_ORDER_ERROR	502	Incorrect calling order.
NET_PLAYM4_TIMER_ERROR	503	Failed to set multimedia clock.
NET_PLAYM4_DEC_VIDEO_ERROR	504	Failed to decode video.
NET_PLAYM4_DEC_AUDIO_ERROR	505	Failed to decode audio.
NET_PLAYM4_ALLOC_MEMORY_ERROR	506	Failed to allocate memory.
NET_PLAYM4_OPEN_FILE_ERROR	507	Failed to open the file.
NET_PLAYM4_CREATE_OBJ_ERROR	508	Failed to create thread event.
NET_PLAYM4_CREATE_DDRAW_ERROR	509	Failed to create directDraw.
NET_PLAYM4_CREATE_OFFSCREEN_ERROR	510	Failed to create back-end buffer.
NET_PLAYM4_BUF_OVER	511	Buffer full. Failed to input stream.
NET_PLAYM4_CREATE_SOUND_ERROR	512	Failed to create audio device.
NET_PLAYM4_SET_VOLUME_ERROR	513	Failed to set the volume.
NET_PLAYM4_SUPPORT_FILE_ONLY	514	This API is only used in file playback mode.
NET_PLAYM4_SUPPORT_STREAM_ONLY	515	This API is only used when playing stream.
NET_PLAYM4_SYS_NOT_SUPPORT	516	Not supported. Decoder can only work on the system above Pentium 3.
NET_PLAYM4_FILEHEADER_UNKNOWN	517	No file header.
NET_PLAYM4_VERSION_INCORRECT	518	Decoder version doesn’t match the encoder.
NET_PALYM4_INIT_DECODER_ERROR	519	Failed to initialize the decoder.
NET_PLAYM4_CHECK_FILE_ERROR	520	File too short or failed to recognize the stream.
NET_PLAYM4_INIT_TIMER_ERROR	521	Failed to initialize the multimedia clock.

NET_PLAYM4_BLT_ERROR	522	BLT error.
NET_PLAYM4_UPDATE_ERROR	523	Failed to update overlay surface.
NET_PLAYM4_OPEN_FILE_ERROR_MULTI	524	Failed to open video & audio stream file.
NET_PLAYM4_OPEN_FILE_ERROR_VIDEO	525	Failed to open video stream file.
NET_PLAYM4_JPEG_COMPRESS_ERROR	526	JPEG compression error.
NET_PLAYM4_EXTRACT_NOT_SUPPORT	527	This file version is not supported.
NET_PLAYM4_EXTRACT_DATA_ERROR	528	Failed to extract file data.

5 Struct Definition

5.1 EAP_PEAP: EAP_PEAP Authentication Parameter

```
public class EAP_PEAP
{
    public byte       byEapolVersion;
    public byte       byAuthType;
    public byte       byPeapVersion;
    public byte       byPeapLabel;
    public byte[]     byAnonyIdentity = new byte[HCNetSDK.NAME_LEN];
    public byte[]     byUserName = new byte[HCNetSDK.NAME_LEN];
    public byte[]     byPassword = new byte[HCNetSDK.NAME_LEN];
}
```

Members

byEapolVersion

EAPOL version: 0-Version 1, 1-Version 2

byAuthType

Internal authentication mode: 0-GTC, 1-MD5, 2-MSCHAPV2

byPeapVersion

PEAP version: 0-Version 0, 1-Version 1

byPeapLabel

PEAP label: 0-Old, 1-New

byAnonyIdentity

Anonymous

byUserName

User name

byPassword

Password

5.2 EAP_TLS: EAP_TLS Authentication Parameter

```
public class EAP_TLS
{
    public byte       byEapolVersion;
    public byte[]     byIdentity = new byte[HCNetSDK.NAME_LEN];
    public byte[]     byPrivateKeyPswd = new byte[HCNetSDK.NAME_LEN];
}
```

Members

byEapolVersion

EAPOL version: 0-Version 1, 1-Version 2

byIdentity

Identity

byPrivateKeyPswd

Private password

5.3 EAP_TTLS: EAP_TTLS Authentication Parameter

```
public class EAP_TTLS
{
    public byte       byEapolVersion;
    public byte       byAuthType;
    public byte[]     byAnonyIdentity = new byte[HCNetSDK.NAME_LEN];
    public byte[]     byUserName = new byte[HCNetSDK.NAME_LEN];
    public byte[]     byPassword = new byte[HCNetSDK.NAME_LEN];
}
```

Members

byEapolVersion

EAPOL version: 0-Version 1, 1-Version 2

byAuthType

Internal authentication mode: 0-PAP, 1-MSCHAPV2

byRes1

Reserved. Set as 0.

byAnonyIdentity

Anonymous

byUserName

User name

byPassword

Password

5.4 NET_DVR_ACTIVATECFG: Device Activation Parameter

```
public class NET_DVR_ACTIVATECFG
{
    public byte[] sPassword = new byte[HCNetSDK.PASSWD_LEN];
}
```

Members

sPassword

Default password. Password level is weak or above.

Remarks

- The device needs to be activated first. Then you can login the device with the set password in activation.
- The password can be divided into four levels with numbers (0 to 9), lowercases (a to z), uppercases (A to Z), special characters (excluding \ : ").

  1) Level 0 (Risky Password): Password is less than 8 characters, or only contains one character type of the

above four types, or is the same as the username, or is the username reversed. Example: 12345, abcdef.

2) Level 1 (Weak Password): Password contains two character types and the combination is number with lowercase or number with uppercase. The password is more than or equal to 8 characters. Example: abc12345, 123ABCDEF.

3) Level 2 (Medium Password): Password contains two character types and the combination cannot be number with lowercase or number with uppercase. The password is more than or equal to 8 characters. Example: 12345***++, ABCDabcd.

4) Level 3 (Strong Password): Password contains three or more character types and is more than or equal to 8 characters. Example: Abc12345, abc12345++.

5.5 NET_DVR_ALARMER: Alarm Device Information

```
public class NET_DVR_ALARMER
{
    public byte      byUserIDValid;
    public byte      bySerialValid;
    public byte      byVersionValid;
    public byte      byDeviceNameValid;
    public byte      byMacAddrValid;
    public byte      byLinkPortValid;
    public byte      byDeviceIPValid;
    public byte      bySocketIPValid;
    public int       lUserID;
    public byte[]    sSerialNumber = new byte[HCNetSDK.SERIALNO_LEN];
    public int       dwDeviceVersion;
    public byte[]    sDeviceName = new byte[HCNetSDK.NAME_LEN];
    public byte[]    byMacAddr = new byte[HCNetSDK.MACADDR_LEN];
    public short     wLinkPort;
    public byte[]    sDeviceIP = new byte[128];
    public byte[]    sSocketIP = new byte[128];
    public byte      byIpProtocol;
    public byte[]    byRes2 = new byte[11];
}
```

Members

byUserIDValid

Whether userid is valid: 0-Invalid, 1-Valid

bySerialValid

Whether serial No. is valid: 0-Invalid, 1-valid

byVersionValid

Whether version No. is valid: 0-Invalid, 1-valid

byDeviceNameValid

Whether device name is valid: 0-Invalid, 1-valid

byMacAddrValid

Whether Mac address is valid: 0-Invalid, 1-valid

byLinkPortValid

Whether login port is valid: 0-Invalid, 1-valid

byDeviceIPValid

Whether device IP is valid: 0-Invalid, 1-valid

bySocketIPValid

Whether socket IP is valid: 0-Invalid, 1-valid

lUserID

The returned value of NET_DVR_Login or NET_DVR_Login_V30, valid when arming

sSerialNumber

Serial No.

dwDeviceVersion

Version information: For V3.0 and above, the highest 8 bits are main version number, higher 8 bits are sub version number and lower 16 bits are fix version number. For V3.0 and below, the high 16 bits are main version number and low 16 bits are sub vrsion number.

sDeviceName

Device number

byMacAddr

MAC address

wLinkPort

Communication port

sDeviceIP

Device IP address

sSocketIP

The Socket IP address when uploading alarm

byIpProtocol

IP protocol: 0-IPV4, 1-IPV6

byRes2

Reserved. Set as 0.

5.6 NET_DVR_ALARMINCFG_V30: Alarm Input Parameter

```
public class NET_DVR_ALARMINCFG_V30 extends NET_DVR_CONFIG
{
    public byte[]                          sAlarmInName = new byte[HCNetSDK.NAME_LEN];
    public byte                            byAlarmType;
    public byte                            byAlarmInHandle;
    public byte                            byChannel;
    public NET_DVR_HANDLEEXCEPTION_V30 struAlarmHandleType = new NET_DVR_HANDLEEXCEPTION_V30();
    public NET_DVR_SCHEDTIME[][]           struAlarmTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];
    public byte[]                          byRelRecordChan = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]                          byEnablePreset = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]                          byPresetNo = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]                          byEnableCruise = new byte[HCNetSDK.MAX_CHANNUM_V30];
```

```
    public byte[]                                   byCruiseNo = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]                                   byEnablePtzTrack = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]                                   byPTZTrack = new byte[HCNetSDK.MAX_CHANNUM_V30];

    public    NET_DVR_ALARMINCFG_V30()
    {
        for(int i=0; i<HCNetSDK.MAX_DAYS; i++)
        {
            for(int j=0; j<HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struAlarmTime[i][j] = new NET_DVR_SCHEDTIME();
            }
        }
    }
}
```

Members

sAlarmInName

Alarm input name

byAlarmType

Alarm type: 0-Normally Open, 1-Normally Closed

byAlarmInHandle

Handle or not: 0-No, 1-Yes

byChannel

Alarm input triggers intelligent identification channel

struAlarmHandleType

Handle mode

struAlarmTime

Arming time parameter

byRelRecordChan

The recording channel triggered by the alarm. If it equals to 1, this channel is triggered.

byEnablePreset

The channel enables calling preset or not (each array corresponds to one channel). 0-No, 1-Yes

byPresetNo

The PTZ preset No. called by the channel. One alarm input can call multiple presets (one preset for each channel).

byEnableCruise

Call patrol or not: 0-No, 1-Yes

byCruiseNo

Patrol path

byEnablePtzTrack

Call pattern or not: 0-No, 1-Yes

byPTZTrack

The pattern No.

5.7 NET_DVR_ALARMINFO: Alarm Information

```
public class NET_DVR_ALARMINFO extends NET_DVR_BASE_ALARM
{
    public int      dwAlarmType;
    public int      dwAlarmInputNumber;
    public int[]    dwAlarmOutputNumber = new int[HCNetSDK.MAX_ALARMOUT];
    public int[]    dwAlarmRelateChannel = new int[HCNetSDK.MAX_CHANNUM];
    public int[]    dwChannel = new int[HCNetSDK.MAX_CHANNUM];
    public int[]    dwDiskNumber = new int[HCNetSDK.MAX_DISKNUM] ;
}
```

Members

dwAlarmType

Alarm type: 0-Sensor Alarm, 1-HDD full, 2-Signal Loss, 3-Motion detection, 4-HDD unformatted, 5-HDD read and write error, 6-Video tampering alarm, 7-Video standard mismatch, 8-Invalid login

dwAlarmInputNumber

Alarm input port. When alarm type is 9, this value means serial port status: 0-Normal, 0xffffffff-Exceptional

dwAlarmOutputNumber

The triggered alarm output port. When the alarm type is 0 and this value is 1, it means alarm output from current port.

dwAlarmRelateChannel

The triggered recording channel. When alarm type is 0 and this value is 1, it means recording by current channel. Example: dwAlarmRelateChannel[0]=1 means triggering the first channel recording.

dwChannel

The channel triggering alarm. Valid when alarm type is 2, 3, and 6. Example: dwChannel[0]=1 means the first channel triggering alarm.

dwDiskNumber

The HDD triggering alarm. Valid when alarm type is 1, 4, and 5.

Example: dwDiskNumber[0]=1 means the No.1 HDD is exceptional.

5.8 NET_DVR_ALARMINFO_V30: The Uploaded Alarm Information

```
public class NET_DVR_ALARMINFO_V30 extends NET_DVR_BASE_ALARM
{
    public int          dwAlarmType;
    public int          dwAlarmInputNumber;
    public byte[]       byAlarmOutputNumber =    new byte[HCNetSDK.MAX_ALARMOUT_V30];
    public byte[]       byAlarmRelateChannel = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]       byChannel = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]       byDiskNumber = new byte[HCNetSDK.MAX_DISKNUM_V30];
}
```

Members

dwAlarmType

Alarm type: 0-Sensor Alarm, 1-HDD full, 2-Signal Loss, 3-Motion detection, 4-HDD unformatted, 5-HDD read and write error, 6-Video tampering alarm, 7-Video standard mismatch, 8-Invalid login, 9-Video signal exception, 10-Record/Capture exception, 11-Intelligent scene change, 12-Array exception, 13-Camera/Recording resolution mismatch

dwAlarmInputNumber

Alarm input port.

byAlarmOutputNumber

The triggered alarm output port. When it is 1, it means alarm output from current port. Example: byAlarmOutputNumber[0]=1 means alarm output from the first output port. byAlarmOutputNumber[1]=1 means alarm output from the second output port, etc.

byAlarmRelateChannel

The triggered recording channel. When it is 1, it means recording by current channel. Example: byAlarmRelateChannel[0]=1 means triggering the first channel recording.

byChannel

The channel triggering alarm. Valid when alarm type is 2, 3, 6, 9, 10, 11. Example: byChannel[0]=1 means the first channel triggering alarm.

byDiskNumber

The HDD triggering alarm. Valid when alarm type is 1, 4, and 5.

Example: byDiskNumber[0]=1 means the No.1 HDD is exceptional.

5.9 NET_DVR_ALARMOUTCFG_V30: Alarm Output Parameter

```
public class NET_DVR_ALARMOUTCFG_V30 extends NET_DVR_CONFIG
{
    public byte[]                       sAlarmOutName = new byte[HCNetSDK.NAME_LEN];
    public int                          dwAlarmOutDelay;
    public NET_DVR_SCHEDTIME[][]        struAlarmOutTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];
    public    NET_DVR_ALARMOUTCFG_V30()
    {
        for(int i=0; i<HCNetSDK.MAX_DAYS; i++)
        {
            for(int j=0; j<HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struAlarmOutTime[i][j] = new NET_DVR_SCHEDTIME();
            }
        }
    }
}
```

Members

sAlarmOutName

Alarm Output Name

dwAlarmOutDelay

The alarm output duration. 0-5s, 1-10s, 2-30s, 3-1 min, 4-2 min, 5-5 min, 6-10 min, 7-Manual (Manually

disable the alarm), other value-5 min

struAlarmOutTime

The alarm output activation time

5.10 NET_DVR_ALARMOUTSTATUS_V30: Alarm Output Status

```
public class NET_DVR_ALARMOUTSTATUS_V30 extends NET_DVR_CONFIG
{
    public byte[] Output = new byte[HCNetSDK.MAX_ALARMOUT_V30];
}
```

Members

Output

Alarm output status: 0-Invalid, 1-Valid

5.11 NET_DVR_AP_INFO: Single Wireless Network Resource Parameter

```
public class NET_DVR_AP_INFO
{
    public byte[]   sSsid = new byte[HCNetSDK.IW_ESSID_MAX_SIZE];
    public int      dwMode;
    public int      dwSecurity;
    public int      dwChannel;
    public int      dwSignalStrength;
    public int      dwSpeed;
}
```

Members

sSsid

SSID

dwMode

Working mode: 0-mange mode, 1-ad-hoc mode

dwSecurity

Security mode: 0-None, 1-wep, 2-wpa-psk, 3-wpa-Enterprise

dwChannel

Channel No.

dwSignalStrength

0 to10, signal becomes more and more strong.

dwSpeed

Speed rate (0.01 mbps)

5.12 NET_DVR_AP_INFO_LIST: Wireless Network Resource List

```
public class NET_DVR_AP_INFO_LIST extends NET_DVR_CONFIG
```

```
{
    public int                      dwCount;
    public NET_DVR_AP_INFO[]        struApInfo = new NET_DVR_AP_INFO[HCNetSDK.WIFI_MAX_AP_COUNT];
    public NET_DVR_AP_INFO_LIST()
    {
        for(int i = 0; i < HCNetSDK.WIFI_MAX_AP_COUNT; i++)
        {
            struApInfo[i] = new NET_DVR_AP_INFO();
        }
    }
}
```

Members

dwCount

Wireless AP number. Less than 20.

struApInfo

AP parameter struct.

5.13 NET_DVR_BASE_ALARM: Basic Alarm Information

```
public class NET_DVR_BASE_ALARM
{
    protected NET_DVR_BASE_ALARM(){}
}
```

Remarks

The callback function of NET_DVR_SetDVRMessageCallBack_V30 has different alarm information types (lCommand), which correspond to different sub type.

Information Type lCommand	Content	pAlarmInfo Corresponding Sub Type
COMM_ALARM	Alarm information suppoered by device of V3.0 and below	NET_DVR_ALARMINFO
COMM_ALARM_V30	Alarm information suppoered by device of V3.0 and above	NET_DVR_ALARMINFO_V30

5.14 NET_DVR_CHANNELSTATE_V30: Channel Status

```
public class NET_DVR_CHANNELSTATE_V30
{
    public byte                     byRecordStatic;
    public byte                     bySignalStatic;
    public byte                     byHardwareStatic;
    public int                      dwBitRate;
    public int                      dwLinkNum;
    public NET_DVR_IPADDR[]         struClientIP = new NET_DVR_IPADDR[HCNetSDK.MAX_LINK];
    public int                      dwIPLinkNum;
    public byte                     byExceedMaxLink;
```

```
    public NET_DVR_CHANNELSTATE_V30(){
        for(int i=0; i<HCNetSDK.MAX_LINK; i++){
            struClientIP[i] = new NET_DVR_IPADDR();
        }
    }
}
```

Members

byRecordStatic

Recording or not: 0-No, 1-Yes

bySignalStatic

The connected signal status: 0-Normal, 1-Signal loss

byHardwareStatic

Hardware status: 0-Normal, 1-Exceptional (e.g., DSP exception)

dwBitRate

Real bitrate

dwLinkNum

The client number connecting to the current channel

struClientIP

The client IP address connecting to the current channel

dwIPLinkNum

The connection number of IP access

byExceedMaxLink

Exceed the 6 connections for each channel or not: 0-No, 1-Exceeded.

5.15 NET_DVR_CLIENTINFO: Live View Parameter

```
public class NET_DVR_CLIENTINFO
{
    public int      lChannel;
    public int      lLinkMode;
    public String   sMultiCastIP;
}
```

Members

lChannel

Channel No. Currently, the device only supports one Channel-Zero with channel No.1.

lLinkMode

The highest bit (31): 0-Main stream, 1-Sub stream. Other bits (0 to30) indicate the connection mode: 0-TCP mode, 1-UDP mode, 2-Multicast mode.

Example: Sub stream TCP connection, lLinkMode=0x80000000.

sMultiCastIP

Multicast address

5.16 NET_DVR_COMPRESSION_AUDIO: Two-Way Audio Parameter

```
struct{
   BYTE      byAudioEncType;
   BYTE      byres[7];
}NET_DVR_COMPRESSION_AUDIO, *LPNET_DVR_COMPRESSION_AUDIO;
```

Members

byAudioEncType

Audio encoding type: 0-G722, 1-G711_U, 2- G711_A, 5-MP2L2, 6-G726, 7-AAC, 8-PCM

byres

Reserved, set as 0.

Remarks

You need to reboot the device after setting the two-way audio parameters. If NET_DVR_GetDVRConfig returns Fault and the error code is 23 (device not support), it means the audio encoding type is G722.

5.17 NET_DVR_COMPRESSION_INFO_V30: Stream Compression Parameter

```
public class NET_DVR_COMPRESSION_INFO_V30
{
      public byte       byStreamType;
      public byte       byResolution;
      public byte       byBitrateType;
      public byte       byPicQuality;
      public int        dwVideoBitrate;
      public int        dwVideoFrameRate;
      public short      wIntervalFrameI;
      public byte       byIntervalBPFrame;
      public byte       byENumber;
      public byte       byVideoEncType;
      public byte       byAudioEncType;
      public byte[]     byRes = new byte[10];
}
```

Members

byStreamType

Stream type: 0-video stream, 1-video and audio stream, 0xfe-Auto (consist with the source)

If it is event compression parameter (struEventRecordPara), the highest bit (byStreamType & 0x80) represents whether to enable it, which means when configuring struEventRecordPara:

byStreamType&0x80 == 0 Disable event compression parameter.

(byStreamType&0x80 == 1)&&(byStreamType&0x7f == 0) Enable event compression parameter. Set stream as video stream.

(byStreamType&0x80 == 1)&&(byStreamType&0x7f == 1) Enable event compression parameter. Set stream

as video and audio stream.

byStreamType==0xfe Enable event compression parameter. Set stream as the source.

byResolution

Resolution: 0-DCIF(528*384/528*320), 1-CIF(352*288/352*240), 2-QCIF(176*144/176*120), 3-4CIF(704*576/704*480) or D1(720*576/720*486), 4-2CIF(704*288/704*240), 6-QVGA(320*240), 7-QQVGA(160*120), 12-384*288, 13-576*576, 16-VGA(640*480), 17-UXGA(1600*1200), 18-SVGA(800*600), 19-HD720P(1280*720), 20-XVGA(1280*960), 21-HD900P(1600*900), 23-1536*1536, 24-1920*1920, 27-1920*1080p, 28-2560*1920, 29-1600*304, 30-2048*1536, 31-2448*2048, 32-2448*1200, 33-2448*800, 34-XGA(1024*768), 35-SXGA(1280*1024), 36-WD1(960*576/960*480), 37-1080i(1920*1080), 38-WXGA(1440*900), 39-HD_F(1920*1080/1280*720), 40-HD_H(1920*540/1280*360), 41-HD_Q(960*540/630*360), 42-2336*1744, 43-1920*1456, 44-2592*2048, 45-3296*2472, 46-1376*768, 47-1366*768, 48-1360*768, 49-WSXGA+, 50-720*720, 51-1280*1280, 52-2048*768, 53-2048*2048, 54-2560*2048, 55-3072*2048, 56-2304*1296, 57-WXGA(1280*800), 58-1600*600, 59-1600*900, 0xff-Auto(use current stream's resolution)

byBitrateType

Bitrate type: 0- Variable, 1-Constant

byPicQuality

Image quality: 0-Maximum, 1-High, 2-High, 3-Normal, 4-Low, 5-Minimun, 0xfe-Auto (consist with the source)

dwVideoBitrate

Bitrate: 0-Reserved, 1-16K (reserved), 2-32K, 3-48K, 4-64K, 5-80K, 6-96K, 7-128K, 8-160K, 9-192K, 10-224K, 11-256K, 12-320K, 13-384K, 14-448K, 15-512K, 16-640K, 17-768K, 18-896K, 19-1024K, 20-1280K, 21-1536K, 22-1792K, 23-2048K, 24-3072K, 25-4096K, 26-8192K, 27-16384K, 0xfffffffe-Auto (consist with the source)

The highest bit (31): 1-Self-defined stream. Other bits (0 to30) indicate stream value and the minimum is 16K.

dwVideoFrameRate

Frame rate: 0-All, 1-1/16, 2-1/8, 3-1/4, 4-1/2, 5-1, 6-2, 7-4, 8-6, 9-8, 10-10, 11-12, 12-16, 13-20, 14-15, 15-18, 16-22, 17-25, 18-30, 19-35, 20-40, 21-45, 22-50, 23-55, 24-60, 25-3, 26-5, 27-7, 28-9, 29-100, 30-120, 31-24, 32-48, 0xfffffffe-Auto (consist with the source)

wIntervalFrameI

I frame interval. 0xfffe-Auto (consist with the source), 0xffff-Invalid

byIntervalBPFrame

Frame format: 0-BBP frame, 1-BP frame, 2-Single P frame, 0xff-Invalid

byENumber

E frame number

byVideoEncType

Video encoding type: 0-Private 264, 1-Standard H.264, 2-Standard MPEG4, 7-M-JPEG, 8-MPEG2, 9-SVAC, 10-Standard H.265, 0xfe-Auto (consist with the source), 0xff-Invalid

byAudioEncType

Audio encoding type: 0-G722, 1-G711_U, 2-G711_A, 5-MP2L2, 6-G726, 7-AAC, 8-PCM, 0xfe-Auto (consist with the source), 0xff-Invalid

byRes

Reserved, set as 0.

5.18 NET_DVR_COMPRESSIONCFG_V30: Channel Compression Parameter

```
public class NET_DVR_COMPRESSIONCFG_V30 extends NET_DVR_CONFIG
{
    public NET_DVR_COMPRESSION_INFO_V30    struNormHighRecordPara = new
NET_DVR_COMPRESSION_INFO_V30();
    public NET_DVR_COMPRESSION_INFO_V30    struEventRecordPara = new
NET_DVR_COMPRESSION_INFO_V30();
    public NET_DVR_COMPRESSION_INFO_V30    struNetPara = new NET_DVR_COMPRESSION_INFO_V30();
    public NET_DVR_COMPRESSION_INFO_V30    struRes = new NET_DVR_COMPRESSION_INFO_V30();
}
```

Members

struNormHighRecordPara

Record's stream compression parameter (the main stream's compression parameter)

struEventRecordPara

Event triggering compression parameter

struNetPara

Stream compression parameter via network (the sub stream's compression parameter)

struRes

Reserved, set as 0.

5.19 NET_DVR_DDNSPARA_V30: Network Application Parameter (DDNS)

```
public class NET_DVR_DDNSPARA_V30 extends NET_DVR_CONFIG
{
    public byte                         byEnableDDNS;
    public byte                         byHostIndex;
    public byte[]                       byRes1 = new byte[2];
    public NET_DVR_SINGLE_DDNS[] struDDNS = new NET_DVR_SINGLE_DDNS[HCNetSDK.MAX_DDNS_NUMS];
    public byte[]                       byRes2 = new byte[16];

    public NET_DVR_DDNSPARA_V30(){
        for(int i = 0; i < HCNetSDK.MAX_DDNS_NUMS; i++)
        {
            struDDNS[i] = new NET_DVR_SINGLE_DDNS();
        }
    }
}
```

Members

byEnableDDNS

Enable or not: 0-No, 1-Yes

byHostIndex

Domain name type: 0-Private DNS, 1-Dyndns, 2-PeanutHull, 3-NO-IP, 4-hiDDNS

byRes1

Reserved, set as 0.

sUsername

DDNS account username

struDDNS

DDNS server parameter

byRes

Reserved, set as 0.

5.20 NET_DVR_DECODERCFG_V30: PTZ Decoder Parameter

```
public class NET_DVR_DECODERCFG_V30 extends NET_DVR_CONFIG
{
    public int          dwBaudRate;
    public byte         byDataBit;
    public byte         byStopBit;
    public byte         byParity;
    public byte         byFlowcontrol;
    public short        wDecoderType;
    public short        wDecoderAddress;
    public byte[]       bySetPreset = new byte[HCNetSDK.MAX_PRESET_V30];
    public byte[]       bySetCruise = new byte[HCNetSDK.MAX_CRUISE_V30];
    public byte[]       bySetTrack = new byte[HCNetSDK.MAX_TRACK_V30];
}
```

Members

dwBaudRate

Bitrate (bps). 0-50, 1-75, 2-110, 3-150, 4-300, 5-600, 6-1200, 7-2400, 8-4800, 9-9600, 10-19200, 11-38400, 12-57600, 13-76800, 14-115.2k

byDataBit

The data bit: 0-5 bits, 1-6 bits, 2-7 bits, 3-8 bits

byStopBit

Stop bit: 0-1 bit, 1-2 bit

byParity

Parity or nor: 0-Null, 1-Odd, 2-Even

byFlowcontrol

Flow control or not: 0-Null, 1-Software, 2-Hardware

wDecoderType

Decoder type. Get by NET_DVR_GetPTZProtocol. This value corresponds to the dwType of NET_DVR_PTZ_PROTOCOL.

wDecoderAddress

Decoder address: [0,255]

bySetPreset

Set preset or not: 0-No, 1-Yes

bySetCruise

Set patrol or not: 0-No, 1-Yes

bySetTrack

Set pattern or not: 0-No, 1-Yes

5.21 NET_DVR_DEVICECFG_V40: Device Parameter

```
public class NET_DVR_DEVICECFG_V40 extends NET_DVR_CONFIG{
    public byte[] byDevTypeName = new byte[24];
}
```

Members

byDevTypeName

(Read only, not editable) Device mode

5.22 NET_DVR_DEVICEINFO_V30: Device Parameter

```
public class NET_DVR_DEVICEINFO_V30
{
    public byte[]   sSerialNumber = new byte[SERIALNO_LEN];
    public byte     byAlarmInPortNum;
    public byte     byAlarmOutPortNum;
    public byte     byDiskNum;
    public byte     byDVRType;
    public byte     byChanNum;
    public byte     byStartChan;
    public byte     byAudioChanNum;
    public byte     byIPChanNum;
    public byte     byZeroChanNum;
    public short    wDevType;
    public byte     byStartDChan
    public byte     byHighDChanNum
}
```

Members

sSerialNumber

erial No.

byAlarmInPortNum

Alarm input No.

byAlarmOutPortNum

Alarm output No.

byDiskNum

HDD No.

byDVRType

Device type

byChanNum

Analog channel No.

byStartChan

The start channel No. of analog channel

byAudioChanNum

Device audio channel No.

byIPChanNum

The maximum IP channel number. Low 8 bits.

byZeroChanNum

Channel-zero decoding No.

wDevType

Device type

byStartDChan

The start digital channel No.

byHighDChanNum

Digital channel number. High 8 bits.

Members

If byDVRType=0, then the API will take wDevType as the device mode. If byDVRType is not 0, then byDVRType=wDevType=byDVRType.

We recommend that you may use wDevType as the device type.

Table 5.1 byDVRType and wDevType Value and Definition

Macro Definition	Value	Device Type
DVR	1	Undefinied DVR type
ATMDVR	2	ATM DVR
DVS	3	DVS
DEC	4	6001D
ENC_DEC	5	6001F
DVR_HC	6	8000HC
DVR_HT	7	8000HT
DVR_HF	8	8000HF
DVR_HS	9	8000HS DVR (no audio)
DVR_HTS	10	8016HTS DVR (no audio)
DVR_HB	11	HB DVR (SATA HD)
DVR_HCS	12	8000HCS DVR
DVS_A	13	DVS with ATA HDD
DVR_HC_S	14	8000HC-S
DVR_HT_S	15	8000HT-S

DVR_HF_S	16	8000HF-S
DVR_HS_S	17	8000HS-S
ATMDVR_S	18	ATM-S
DVR_7000H	19	7000H series
DEC_MAT	20	Multi-channel decoder
DVR_MOBILE	21	Mobile DVR
DVR_HD_S	22	8000HD-S
DVR_HD_SL	23	8000HD-SL
DVR_HC_SL	24	8000HC-SL
DVR_HS_ST	25	8000HS_ST
DVS_HW	26	6000HW
DS630X_D	27	Multi-channel decoder
DS640X_HD	28	640X HD decoder
DS610X_D	29	610X decoder
IPCAM	30	Network camera
MEGA_IPCAM	31	HD network camera
IPCAM_X62MF	32	X62MF series camera
ITCCAM	35	Intelligent traffic camera
IVS_IPCAM	36	VCA HD IP camera (capture camera)
ZOOMCAM	38	Zoom camera
IPDOME	40	IP SD speed dome
IPDOME_MEGA200	41	IP 200 MP HD speed dome
IPDOME_MEGA130	42	IP 130 MP HD speed dome
TII_IPCAM	44	Thermal camera
IPMOD	50	IP module
IDS6501_HF_P	60	6501 license plate recognization
IDS6101_HF_A	61	Intelligent ATM
IDS6002_HF_B	62	Dual-camera tracking: DS6002-HF/B
IDS6101_HF_B	63	Behavior analysis: DS6101-HF/B
IDS52XX	64	Smart analyzer
IDS90XX	65	9000 intelligence
IDS8104_AHL_S_HX	66	HAIXING face detection ATM
IDS8104_AHL_S_H	67	Private face detection ATM
IDS91XX	68	9100 intelligence

IIP_CAM_B	69	VCA IP camera
IIP_CAM_F	70	Smart face detection IP camera
DS71XX_H	71	DS71XXH_S
DS72XX_H_S	72	DS72XXH_S
DS73XX_H_S	73	DS73XXH_S
DS72XX_HF_S	74	DS72XX_HF_S
DS73XX_HFI_S	75	DS73XX_HFI_S
DS76XX_H_S	76	DS76XX_H_S
DS76XX_N_S	77	DS76XX_N_S
DS81XX_HS_S	81	DS81XX_HS_S
DS81XX_HL_S	82	DS81XX_HL_S
DS81XX_HC_S	83	DS81XX_HC_S
DS81XX_HD_S	84	DS81XX_HD_S
DS81XX_HE_S	85	DS81XX_HE_S
DS81XX_HF_S	86	DS81XX_HF_S
DS81XX_AH_S	87	DS81XX_AH_S
DS81XX_AHF_S	88	DS81XX_AHF_S
DS90XX_HF_S	90	DS90XX_HF_S
DS91XX_HF_S	91	DS91XX_HF_S
DS91XX_HD_S	92	91XXHD-S(MD)
IDS90XX_A	93	9000 intelligent ATM
IDS91XX_A	94	9100 intelligent ATM
DS95XX_N_S	95	DS95XXN-S NVR
DS96XX_N_SH	96	DS96XXN-SH NVR
DS90XX_HF_SH	97	DS90XX_HF_SH
DS91XX_HF_SH	98	DS91XX_HF_SH
DS_65XXHC	105	65XXHC DVS
DS_65XXHC_S	106	65XXHC-SATA DVS
DS_65XXHF	107	65XXHF DVS
DS_65XXHF_S	108	65XXHF-SATA DVS
DS_6500HF_B	109	65 rack DVS
IVMS_6200_C	110	iVMS-6200(/C) people counting statistics
IVMS_6200_B	111	IVMS_6200_B behavior analysis
DS_72XXHV_ST15	112	72XXHV_ST15 DVR

DS_72XXHV_ST20	113	72XXHV_ST20 DVR
IVMS_6200_T	114	IVMS-6200(/T)
IVMS_6200_BP	115	IVMS-6200(/BP)
DS_81XXHC_ST	116	DS_81XXHC_ST
DS_81XXHS_ST	117	DS_81XXHS_ST
DS_81XXAH_ST	118	DS_81XXAH_ST
DS_81XXAHF_ST	119	DS_81XXAHF_ST
DS_66XXDVS	120	DS_66XXDVS
DS_19AXX	142	General security control panel
DS_19CXX	144	ATM security control panel
DS_19DXX	145	Power supply monitoring security control panel
DS_19XX	146	1900 series security control panel
DS_19SXX	147	Video security control panel
DS_1HXX	148	ATM protective cabin controller
DS_C10H	161	Multi-screen controller
DS_C10N_BI	162	BNC processer
DS_C10N_DI	163	RGB processer
DS_C10N_SI	164	Stream processer
DS_C10N_DO	165	Display processer
DS_C10N_SERVER	166	Distributed server
IDS_8104_AHFL_S_H	171	8104 ATM
IDS_65XX_HF_A	172	65 ATM
IDS90XX_HF_RH	173	9000 smart RH
IDS91XX_HF_RH	174	9100 smart RH
IDS_65XX_HF_B	175	65 behavior analysis
IDS_65XX_HF_P	176	65 license plate recognization
IVMS_6200_F	177	IVMS-6200(/F) face analyzer
IVMS_6200_F_S	179	IVMS-6200(/F_S)
DS90XX_HF_RH	181	DS90XX_HF_RH
DS91XX_HF_RH	182	9100 RH
DS78XX_S	183	78 series
DS81XXHW_S	185	DVR_81XXHW_S
DS81XXHW_ST	186	DVR_81XXHW_ST
DS91XXHW_ST	187	DVR_91XXHW_ST

DS91XX_ST	188	DVR_91XX_ST
DS81XX_ST	189	DVR_81XX_ST
DS81XXH_ST	190	DS81XXHDI_ST,DS81XXHE_ST
DS73XXH_ST	191	DS73XXHI_ST
DS81XX_SH	192	Trial 81SH, 81SHF
DS81XX_SN	193	Trial 81SNL
DS96XXN_ST	194	NVR: DS96xxN_ST
DS86XXN_ST	195	NVR: DS86xxN_ST
DS80XXHF_ST	196	DS80xxHF_ST
DS90XXHF_ST	197	DS90xxHF_ST
DS76XXN_ST	198	NVR: DS76xxN_ST
DS_9664N_RX	199	NVR: DS-9664N-RH, DS-9664N-RT
ENCODER_SERVER	200	Encoder server
DECODER_SERVER	201	Decoder server
PCNVR_SERVER	202	PCNVR storage server
CVR_SERVER	203	CVR
DS_91XXHFH_ST	204	HD DVR: DS_91xxHFH_ST
DS_66XXHFH	205	66 HD encoder
TRAFFIC_TS_SERVER	210	Terminal server
TRAFFIC_VAR	211	Video analysis recorder
IPCALL	212	IP video intercom
DS64XXHD_T	701	64-T HD decoder
DS_65XXD	703	65 universal decoder
DS63XXD_T	704	63-T HD decoder
DS_64XXHD_S	706	DS-64xxHD-S HD decoder
DS_68XXT	707	Multi-functional video and audio distributor
DS_65XXD_T	708	65D-T universal decoder
IPCAM_FISHEYE	1002	Fisheye camera
TRAFFIC_ECT	1400	Enerance/Exit terminal server
TRAFFIC_PARKING_SERVER	1401	Parking server
DS90XXHW_ST	2001	H-DVR: DS-90xxHW-ST
DS72XXHX_SH	2002	DS-72xxHV-SH, DS-72xxHF-SH
DS_92XX_HF_ST	2003	DS-92xxHF-ST
DS_91XX_HF_XT	2004	Netra DVR: DS-91xxHF-XT

DS_90XX_HF_XT	2005	Netra H-DVR: DS-90xxHF-XT
DS_73XXHX_SH	2006	Netra DVR: DS-73xxHX-SH
DS_72XXHFH_ST	2007	Netra DVR: DS-72xxHFH-ST
DS_67XXHF_SATA	2008	DVS: DS-67xxHF-SATA
DS_67XXHW	2009	DVS: DS-67xxHW
DS_67XXHW_SATA	2010	DVS: DS-67xxHW-SATA
DS_67XXHF	2011	DVS: DS-67xxHF
DS_72XXHF_SV	2012	DVR: DS-72xxHF-SV
DS_72XXHW_SV	2013	DVR: DS-72xxHW-SV
DS_81XXHX_SH	2014	DVR: DS-81xxHX-SH
DS_71XXHX_SL	2015	DVR: DS-71xxHX-SL
DS_77XXN_ST	2201	Netra NVR: DS-77xxN-ST
DS_95XX_N_ST	2202	Netra NVR: DS-95xxN-ST
DS_85XX_N_ST	2203	Netra NVR: DS-85xxN-ST
DS_96XX_N_XT	2204	Netra NVR: DS-96xxN-XT
DS_76XX_N_SE	2205	Netra NVR: DS-76xxN-SE, DS-78xxN-SH
DS_86XXSN_SX	2206	Netra trial NVR: DS-8608SNL-SP, DS-8608SNL-ST, DS-8608SN-SP, DS-8608SN-ST, L:LCD, P: POE
DS_96XX_N_RX	2207	Netra NVR: DS-96xxN-RX
DS_96XXX_N_E	2213	High performance NVR (256-ch)
DS_76XXN_EX	2214	76/78N-EX(/N/P) series NVR, including 4/8/16-ch E1 and 8/16/32-ch E2
DS_77XXN_E4	2215	77N-E4(/N/P) series NVR, including 8/16/32-ch
DS_86XXN_E8	2216	86N-E8(/N/P) series NVR, including 8/16/32-ch
PCNVR_IVMS_4200	2301	iVMS-4200 storage server
IVMS_6200_TP	2401	IVMS-6200 traffic guidance analyzer
IVMS_6200_TF	2402	IVMS-6200 traffic enforcement analyzer

5.23 NET_DVR_DISKSTATE: HDD Information

```
public class NET_DVR_DISKSTATE
{
    public int      dwVolume;
    public int      dwFreeSpace;
    public int      dwHardDiskStatic;
}
```

Members

dwVolume

HDD capacity (MB)

dwFreeSpace

HDD free space (MB)

dwHardDiskStatic

HDD status: 0-Active, 1-Sleeping, 2-Exceptional, 3-Sleeping HDD error, 4-Unformatted, 5-Disconnected (network HDD), 6-HDD formatting

5.24 NET_DVR_DIGITAL_CHANNEL_STATE: Digital Channel Status

```
public class NET_DVR_DIGITAL_CHANNEL_STATE extends NET_DVR_CONFIG
{
    public byte[]    byDigitalAudioChanTalkState = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]    byDigitalChanState = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]    byDigitalAudioChanTalkStateEx = new byte[HCNetSDK.MAX_CHANNUM_V30*3];
    public byte[]    byDigitalChanStateEx = new byte[HCNetSDK.MAX_CHANNUM_V30*3];
    public byte[]    byAnalogChanState = new byte[HCNetSDK.MAX_ANALOG_CHANNUM];
    public byte[]    byRes = new byte[32];
}
```

Members

byDigitalAudioChanTalkState

The two-way audio status of IP audio channel. From the No.1 to No.MAX_CHANNUM_V30. 0-Unused, 1-Under use, 0xff-Invalid

byDigitalChanState

Digital channel status. From the No.1 to No.MAX_CHANNUM_V30. 0-Invalid, other value-See Table 5.2.

byDigitalAudioChanTalkStateEx

The two-way audio status of IP audio channel. From theNo.MAX_CHANNUM_V30+1 to No.MAX_CHANNUM_V30*4. 0-Unused, 1-Under use, 0xff-Invalid.

byDigitalChanStateEx

Digital channel status. From the No.MAX_CHANNUM_V30+1 to No.MAX_CHANNUM_V30*4. 0-Invalid, other value-See Table 5.2.

byRes

Reserved, set as 0.

Remarks

Digital channel status (byDigitalChanState, byDigitalChanStateEx).

Table 5.2 Digital Channel Status

Macro Definition	Value	Implication
NET_SDK_DC_STATUS_CONNECTED	1	Connected
NET_SDK_DC_STATUS_CONNECTING	2	Connecting
NET_SDK_DC_STATUS_BAND_WIDTH_EXCEED	3	Exceed system bandwidth
NET_SDK_DC_STATUS_DOMAIN_ERROR	4	Domain name error
NET_SDK_DC_STATUS_CHANNEL_ERROR	5	Channel No. error

NET_SDK_DC_STATUS_ACCOUNT_ERROR	6	Incorrect username or password
NET_SDK_DC_STATUS_STREAM_TYPE_NOT_SUPPORT	7	Stream type not supported
NET_SDK_DC_STATUS_CONFLICT_WITH_DVR	8	Conflict with device IP address
NET_SDK_DC_STATUS_CONFLICT_WITH_IPC	9	Conflict with IP camera IP address
NET_SDK_DC_STATUS_NETWORK_UNREACHBALE	10	Network unreachable
NET_SDK_DC_STATUS_IPC_NOT_EXIST	11	IP camera does not exist
NET_SDK_DC_STATUS_IPC_EXCEPTION	12	IP camera exception
NET_SDK_DC_STATUS_OTHER_ERROR	13	Other error
NET_SDK_DC_STATUS_RESOLUTION_NOT_SUPPORT	14	IP camera resolution is not supported.
NET_SDK_DC_STATUS_IPC_LAN_ERR	15	Language mismatches
NET_SDK_DC_STATUS_USER_LOCKED	16	User is locked
NET_SDK_DC_STATUS_NOT_ACTIVATED	17	Device inactivated
NET_SDK_DC_STATUS_USER_NOT_EXIST	18	User doesn't exist
NET_SDK_DC_STATUS_IPC_UNREGISTERED	19	The device is unregistered. (GB28181 protocol)

5.25 NET_DVR_ETHERNET_V30: Ethernet Settings

```
public class NET_DVR_ETHERNET_V30 {
    public NET_DVR_IPADDR    struDVRIP = new NET_DVR_IPADDR();
    public NET_DVR_IPADDR    struDVRIPMask = new NET_DVR_IPADDR();
    public int               dwNetInterface;
    public int               wDVRPort;
    public int               wMTU;
    public byte[]            byMACAddr = new byte[HCNetSDK.MACADDR_LEN];
}
```

Members

struDVRIP

Device IP address

struDVRIPMask

Device IP address mask

dwNetInterface

Network interface: 1-10MBase-T, 2-10MBase-T Full-dup, 3-100MBase-TX, 4-100M Full-dup, 5-10M/100M/1000M Self-adaptive, 6-1000M Full-dup

wDVRPort

Device port No.

wMTU

MTU settings. By default, it is 1500.

byMACAddr

Device MAC address

5.26 NET_DVR_FILECOND: The Searched Video File Information

```
public class NET_DVR_FILECOND
{
    public int              lChannel;
    public int              dwFileType;
    public int              dwIsLocked;
    public int              dwUseCardNo;
    public byte[]           sCardNumber = new byte[32];
    public NET_DVR_TIME     struStartTime = new NET_DVR_TIME();
    public NET_DVR_TIME     struStopTime = new NET_DVR_TIME();
}
```

Members

lChannel

Channel No.

dwFileType

Video file type (it can be divided into two types according to whether dwUseCardNo has card No.):

Type without card No.: 0xff-All, 0-Continuous recording, 1-Motion detection, 2-Alarm, 3-Alarm or motion detection, 4-Alarm and motion detection, 5-Command, 6-Manual recording, 7-VCA recording

Type with card No.: 0xff-All, 0-Continuous recording, 1-Motion detection, 2-Approaching alarm, 3-Cash-out alarm, 4-Cash-in alarm, 5-Command, 6-Manual recording, 7-Vibrating alarm, 8-Environmental alarm, 9-VCA alarm

dwIsLocked

Locked or not: 0-Unlocked file, 1-Locked file, 0xff-All files (including locked and unlocked)

dwUseCardNo

Search with card No. or not

sCardNumber

Card No., cannot be NULL when ATM searching by card No.

struStartTime

Start time

struStopTime

End time

5.27 NET_DVR_FINDDATA_V30: Video File Information

```
public class NET_DVR_FINDDATA_V30
{
    public byte[]       sFileName = new byte[100];
    NET_DVR_TIME        struStartTime;
    NET_DVR_TIME        struStopTime;
    public int          dwFileSize;
    public byte[]       sCardNum = new byte[32];
    public byte         byLocked;
```

```
    public byte[]          byRes = new byte[3];
}
```

Members

sFileName

File name

struStartTime

File start time

struStopTime

File end time

dwFileSize

File size

sCardNum

Card No.

byLocked

File locked or not, 1-Locked, 0-Unlocked

byRes

Reserved. Set as 0.

5.28 NET_DVR_HANDLEEXCEPTION_V30: Handle Alarm and Exception

```
public class NET_DVR_HANDLEEXCEPTION_V30
{
    Public int         dwHandleType;
    public byte[]      byRelAlarmOut = new byte[HCNetSDK.MAX_ALARMOUT_V30];
}
```

Members

dwHandleType

Handling mode:

0x00: No response

0x01: Full screen monitoring

0x02: Audible warning

0x04: Notify surveillance center

0x08: Trigger alarm output

0x10: Capture picture in jpeg format and upload to email

0x20: Wireless audible and visual alarm linkage

0x40: E-map linkage (only for PCNVR)

0x200: Capture and upload to ftp

byRelAlarmOut

Alarm output channel: 0-Not trigger, 1-Trigger. Present output channel by bit. Example: byRelAlarmOut[0]==1 means triggering output channel 1, byRelAlarmOut[1]==1 means triggering output channel 2, etc.

5.29 NET_DVR_HIDEALARM_V30: Video Tampering Alarm Parameter

```
public class NET_DVR_HIDEALARM_V30
{
    public int                                  dwEnableHideAlarm;
    public short                                wHideAlarmAreaTopLeftX;
    public short                                wHideAlarmAreaTopLeftY;
    public short                                wHideAlarmAreaWidth;
    public short                                wHideAlarmAreaHeight;
    public NET_DVR_HANDLEEXCEPTION_V30 struHideAlarmHandleType = new
NET_DVR_HANDLEEXCEPTION_V30();
    public NET_DVR_SCHEDTIME[][]                struAlarmTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];

    public NET_DVR_HIDEALARM_V30()
    {
        for(int i = 0; i < HCNetSDK.MAX_DAYS; i++)
        {
            for(int j = 0; j < HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struAlarmTime[i][j] = new NET_DVR_SCHEDTIME();
            }
        }
    }
}
```

Members

dwEnableHideAlarm

Enable video tampering alarm or not. 0-No, 1-Low sensitivity, 2-Medium sensitivity, 3-High sensitivity

wHideAlarmAreaTopLeftX

X-coordinate of tampering area

wHideAlarmAreaTopLeftY

Y-coordinate of tampering area

wHideAlarmAreaWidth

Width of tampering area

wHideAlarmAreaHeight

Height of tampering area

strHideAlarmHandleType

Handling mode

struAlarmTime

Arming time

Remarks

SDK sets the whole image size as 704*576. The coordinate, width, and height should be transformed into the value in 704*576 size.

5.30 NET_DVR_IPADDR: IP Address

```
public class NET_DVR_IPADDR
{
    public byte[] sIpV4 = new byte[16];
    public byte[] sIpV6 = new byte[128];
}
```

Members

sIpV4

Device IPv4 address

sIpV6

Device IPv6 address

5.31 NET_DVR_IPALARMOUTCFG: IP Alarm Output Settings

```
public class NET_DVR_IPALARMOUTCFG
{
    public NET_DVR_IPALARMOUTINFO[] struIPAlarmOutInfo = new
NET_DVR_IPALARMOUTINFO[HCNetSDK.MAX_IP_ALARMOUT];
}
```

Members

struIPAlarmOutInfo

IP alarm output information. Each array indicates one IP alarm output

5.32 NET_DVR_IPALARMOUTINFO: IP Alarm Output Information

```
public class NET_DVR_IPALARMOUTINFO
{
    public byte     byIPID;
    public byte     byAlarmOut;
    public byte[]   byRes = new byte[18];
}
```

Members

byIPID

IP camera ID. Value range:[1,MAX_IP_DEVICE], among them #define MAX_IP_DEVICE 32

byAlarmOut

Alarm output No.

byRes

Reserved, set as 0

5.33 NET_DVR_IPCHANINFO: IP Channel Information

```
public class NET_DVR_IPCHANINFO
{
    public byte    byEnable;
    public byte    byIPID;
    public byte    byChannel;
}
```

Members

byEnable

IP channel on-line status (read only): 0 means that digital channel of HDVR or NVR connecting to IP device is failed, and the channel is off-line; 1 means connection is successful, and the channel is on-line.

byIPID

IP camera ID

byChannel

Channel number of IP device. For example, if No.4 channel of device B (HDVR or NVR) is added to the IP channel No.1 of device A (HDVR or NVR), then byChannel=4.

5.34 NET_DVR_IPDEVINFO_V31: IP Device Information

```
public class NET_DVR_IPDEVINFO_V31
{
    public byte                byEnable;
    public byte                byProType;
    public byte[]              sUserName = new byte[HCNetSDK.NAME_LEN];
    public byte[]              sPassword = new byte[HCNetSDK.PASSWD_LEN];
    public byte[]              byDomain = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public NET_DVR_IPADDR      struIP = new NET_DVR_IPADDR();
    public int                 wDVRPort;
}
```

Members

byEnable

Whether the IP device is enabled

byProType

Protocol type: 0-private (default), 1-Panasonic, 2-sony.

sUserName

Username

sPassword

Password

byDomain

Device domain name

struIP

IP address

wDVRPort

Port No.

Remarks

When all IP channels of one IP device are deleted, which means IPID of all IP channel parameters in IP channel resource minus 1 isn't corresponding to subscript value of these IP device parameters, the local IP device parameter will be deleted.

In this struct, if domain name is null and ipv4 is valid, it will use the ipv4 to connect to the IP device; if both domain name and ipv4 are null, and ipv6 is valid, it will use the ipv6 to connect to the IP device.

5.35 NET_DVR_IPPARACFG_V40: IP Device Resource and IP Channel Resource Configuration

```
public class NET_DVR_IPPARACFG_V40 extends NET_DVR_CONFIG
{
    Public int                          dwGroupNum;
    public int                          dwAChanNum;
    public int                          dwDChanNum;
    public int                          dwStartDChan;
    public byte[]                       byAnalogChanEnable = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public NET_DVR_IPDEVINFO_V31[]      struIPDevInfo = new
NET_DVR_IPDEVINFO_V31[HCNetSDK.MAX_IP_DEVICE_V40];
    public NET_DVR_IPCHANINFO[]         struIPChanInfo = new
NET_DVR_IPCHANINFO[HCNetSDK.MAX_CHANNUM_V30];

    public    NET_DVR_IPPARACFG_V40()
    {
        for(int i=0; i<HCNetSDK.MAX_IP_DEVICE_V40; i++)
        {
            struIPDevInfo[i] = new NET_DVR_IPDEVINFO_V31();
        }
        for(int i=0; i<HCNetSDK.MAX_CHANNUM_V30; i++)
        {
            struIPChanInfo[i] = new NET_DVR_IPCHANINFO();
        }
    }
}
```

Members

dwGroupNum

Group number supported by device (read only). If the number is larger than 1, when call NET_DVR_GetDVRConfig (or NET_DVR_SetDVRConfig) to get (or set) channel parameters, you should call the same API dwGroupNum times to get channel parameters of each group.

lChannel in the configuration APIs corresponds to the group number.

dwAChanNum

The max number of analog channels (read only)

dwDChanNum

The number of digital channels (read only)

dwStartDChan

The starting number of digital channels (read only)

byAnalogChanEnable

Whether analog channels are enabled. The array index corresponds to the channel number.

The value: 0-Disable, 1-Enable. Example: byAnalogChanEnable[i]=1 means channel (i+1) is enabled.

struIPDevInfo

IP device information. The index 0 corresponds to device IPID=1

struIPChanInfo

IP channel information

5.36 NET_DVR_JPEGPARA: JPEG Image Parameter

```
public class NET_DVR_JPEGPARA
{
    public short    wPicSize;
    public short    wPicQulity;
}
```

Members

wPicSize

Picture size: 0-CIF(352*288/352*240), 1-QCIF(176*144/176*120), 2-4CIF(704*576/704*480) or D1(720*576/720*486), 3-UXGA(1600*1200), 4-SVGA(800*600), 5-HD720P(1280*720), 6-VGA(640*480), 7-XVGA(1280*960), 8-HD900P(1600*900), 9-HD1080P(1920*1080), 10-2560*1920, 11-1600*304, 12-2048*1536, 13-2448*2048, 14-2448*1200, 15-2448*800, 16-XGA(1024*768), 17-SXGA(1280*1024), 18-WD1(960*576/960*480), 19-1080I(1920*1080), 20-576*576, 21-1536*1536, 22-1920*1920, 0xff-Auto(use current stream resolution)

wPicQulity

Picture quality: 0-Best, 1-Better, 2-Normal

5.37 NET_DVR_MOTION_V30: Motion Detection Parameter

```
public class NET_DVR_MOTION_V30
{
    public byte[][]                         byMotionScope = new byte[64][96];
    public byte                             byMotionSensitive;
    public byte                             byEnableHandleMotion;
    public NET_DVR_HANDLEEXCEPTION_V30 struMotionHandleType = new
NET_DVR_HANDLEEXCEPTION_V30();
    public NET_DVR_SCHEDTIME[][]            struAlarmTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];
    public byte[]                           byRelRecordChan = new byte[HCNetSDK.MAX_CHANNUM_V30];
```

```
    public NET_DVR_MOTION_V30(){
        for(int i = 0; i < HCNetSDK.MAX_DAYS; i++)
        {
            for(int j = 0; j < HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struAlarmTime[i][j] = new NET_DVR_SCHEDTIME();
            }
        }
    }
}
```

Members

byMotionScope

Motion detection area, only 22*18 for PAL or 22*15 for NTSC format are valid in the array of 96*64. The value =1 means set the macroblock as the detection area, and the value =0 means not set as the detection area. The other is reserved.

byMotionSensitive

Motion detection sensitivity, range: [0,5]. Value = 0xff means disable the function. The value is larger, the detection is more sensitive

byEnableHandleMotion

Whether to handle motion detection: 0-No, 1-Yes

strMotionHandleType

Handling mode

struAlarmTime

Arming time

byRelRecordChan

Recording channel triggered by alarm. 1 means the channel is triggered

Remarks

The area size of total image is 704*576 for PAL or 704*480 for NTSC, and set the image of 704*576(or 704*480) to 22*18(or 22*15) macroblocks. Then we can set each macroblock to be the detection area or not.

For example, in PAL format, if the resolution of the image is not 704*576, such as 1280*720, you need to shrink the image into 704*576, and then set the motion detection area.

5.38 NET_DVR_NETCFG_V30: Network Settings

```
public class NET_DVR_NETCFG_V30 extends NET_DVR_CONFIG{
    public NET_DVR_ETHERNET_V30[]  struetherNet = new NET_DVR_ETHERNET_V30[HCNetSDK.MAX_ETHERNET];
    public NET_DVR_IPADDR          struAlarmHostIpAddr = new NET_DVR_IPADDR();
    public int                     wAlarmHostIpPort;
    public byte                    byUseDhcp;
    public NET_DVR_IPADDR          struDnsServer1IpAddr = new NET_DVR_IPADDR();
    public NET_DVR_IPADDR          struDnsServer2IpAddr = new NET_DVR_IPADDR();
    public byte []                 byIpResolver = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public int                     wIpResolverPort;
    public int                     wHttpPortNo;
```

```
    public NET_DVR_IPADDR              struMulticastIpAddr = new NET_DVR_IPADDR();
    public NET_DVR_IPADDR              struGatewayIpAddr = new NET_DVR_IPADDR();
    public NET_DVR_PPPOECFG            struPPPoE = new NET_DVR_PPPOECFG();
    public NET_DVR_NETCFG_V30(){
        for(int i=0; i<HCNetSDK.MAX_ETHERNET; i++){
            struetherNet[i] = new NET_DVR_ETHERNET_V30();
        }
    }
}
```

Members

struetherNet

Ethernet port

struAlarmHostIpAddr

Security control panel IP address

wAlarmHostIpPort

Security control panel port No.

byUseDhcp

Enable DHCP or not: 0xff-Invalid, 0-No, 1-Yes

struDnsServer1IpAddr

Domain name server 1 IP address

struDnsServer2IpAddr

Domain name server 2 IP address

byIpResolver

Domain name or IP address of IP resolver (8000 series devices don't support domain name)

wIpResolverPort

Port number of IP resolver

wHttpPortNo

HTTP port No.

struMulticastIpAddr

Multicast group IP address

struGatewayIpAddr

Gateway IP address

struPPPoE

PPPoE parameter

5.39 NET_DVR_NTPPARA: Network Application Parameter (NTP)

```
public class NET_DVR_NTPPARA extends NET_DVR_CONFIG
{
        public byte []      sNTPServer = new byte[64];
        public short        wInterval;
        public byte         byEnableNTP;
        public char         cTimeDifferenceH;
        public char         cTimeDifferenceM;
```

```
    public short        wNtpPort;
}
```

Members

sNTPServer

NTP server domain name or IP address

wInterval

Synchronization interval (hour or minute)

byEnableNTP

Enable NTP synchronization or not: 0-No, 1-Yes

cTimeDifferenceH

Time difference with international standard time (hour), value: -12 to +13

cTimeDifferenceM

Time difference with international standard time (minute), value: 0, 30, 45

wNtpPort

NTP server port. By default, it is 123

5.40 NET_DVR_PICCFG_V30: Channel Image

```
public class NET_DVR_PICCFG_V30 extends NET_DVR_CONFIG
{
    public byte[]                   sChanName = new byte[HCNetSDK.NAME_LEN];
    public int                      dwVideoFormat;
    public NET_DVR_VICOLOR          struViColor = new NET_DVR_VICOLOR();
    public int                      dwShowChanName;
    public short                    wShowNameTopLeftX;
    public short                    wShowNameTopLeftY;
    public NET_DVR_VILOST_V30       struVILost = new NET_DVR_VILOST_V30();
    public NET_DVR_MOTION_V30       struMotion = new NET_DVR_MOTION_V30();
    public NET_DVR_HIDEALARM_V30    struHideAlarm = new NET_DVR_HIDEALARM_V30();
    public int                      dwEnableHide;
    public NET_DVR_SHELTER[]        struShelter = new NET_DVR_SHELTER[HCNetSDK.MAX_SHELTERNUM];
    public int                      dwShowOsd;
    public short                    wOSDTopLeftX;
    public short                    wOSDTopLeftY;
    public byte                     byOSDType;
    public byte                     byDispWeek;
    public byte                     byOSDAttrib;
    public byte                     byHourOsdType;
    public byte                     byFontSize;
    public NET_DVR_PICCFG_V30(){
        for(int i = 0; i < HCNetSDK.MAX_SHELTERNUM; i++)
        {
            struShelter[i] = new NET_DVR_SHELTER();
        }
```

}
}

Members

sChanName

Channel name

dwVideoFormat

Video standard: 1-NTSC, 2-PAL

struViColor

Image parameter. Reserved.

dwShowChanName

Whether to display the channel name on the image of live view: 0-No, 1-Yes (area size is 704*576)

wShowNameTopLeftX

The X-coordinate of the display position of channel name

wShowNameTopLeftY

The Y-coordinate of the display position of channel name

struVILost

Video loss alarm parameter

struMotion

Motion detection alarm parameter

struHideAlarm

Tampering alarm parameter

dwEnableHide

Enable privacy mask or not: 0-No, 1-Yes

struShelter

Privacy mask parameter

dwShowOsd

Show the OSD on the image of live view or not: 0-No, 1-Yes (area size is 704*576)

wOSDTopLeftX

X-coordinate of OSD

wOSDTopLeftY

Y-coordinate of OSD

byOSDType

OSD type (year-month-day format):

0—YYYY-MM-DD

1—MM-DD-YYYY

2—YYYY Year MM Month DD Day

3—MM Month DD Day YYYY Year

4—DD-MM-YYYY

5—DD Day MM Month YYYY Year

byDispWeek

Display week or not: 0-No, 1-Yes

byOSDAttrib

OSD properties (Transparent/Twinkle):

1—Transparent and twinkle

2—Transparent, not twinkle

3—Twinkle, not transparent

4—Not transparent and not twinkle

byHourOsdType

Hour format: 0-24 hours, 1-12 hours (am/pm)

byFontSize

Font size: 0-16*16(CN)/8*16(EN), 1-32*32(CN)/16*32(EN), 2-64*64(CN)/32*64(EN), 3-48*48(CN)/24*48(EN), 0xff-Self adaptive

5.41 NET_DVR_POINT_FRAME: PTZ Image Area Position Information

```
public class NET_DVR_POINT_FRAME
{
    public   int     xTop;
    public   int     yTop;
    public   int     xBottom;
    public   int     yBottom;
    public   int     bCounter;
}
```

Members

xTop

X-coordinate of the rectangle starting point

yTop

Y-coordinate of the rectangle starting point

xBottom

X-coordinate of the rectangle ending point

yBottom

Y-coordinate of the rectangle ending point

bCounter

Reserved

Remarks

The coordinate value is related to the size of display box for current live view. Now we suppose that the display box is 352*288, and set upper left point to be original point. Calculation method of the first four parameters is as below:

xTop = (the value of upper left coordinate of currently selected area) *255/352;

xBottom = (the value of upper right coordinate of currently selected area) *255/352;

yTop = (the value of lower left coordinate of currently selected area) *255/288;

yBottom = (the value of lower right coordinate of currently selected area) *255/288;

Zoom-out condition: xTop - xBottom > 2.

Zoom-in condition: xBottom - xTop > 0.

5.42 NET_DVR_PPPOECFG: PPPoE Settings

```
public class NET_DVR_PPPOECFG
{
    public int                  dwPPPOE;
    public byte[]               sPPPoEUser = new byte[HCNetSDK.NAME_LEN];
    public byte[]               sPPPoEPassword = new byte[HCNetSDK.PASSWD_LEN];
    public NET_DVR_IPADDR       struPPPoEIP = new NET_DVR_IPADDR();
}
```

Members

dwPPPOE

Enable PPPoE or not: 0-No, 1-Yes

sPPPoEUser

PPPoE user name

sPPPoEPassword

PPPoE password

struPPPoEIP

PPPoE IP address

5.43 NET_DVR_PRESET_NAME: Preset Name Settings

```
public class NET_DVR_PRESET_NAME
{
    public int          wPresetNum;
    public byte[]       byName = new byte[HCNetSDK.NAME_LEN];
}
```

Members

wPresetNum

Preset No.

byName

Preset name

5.44 NET_DVR_PRESET_NAME_ARRAY: Preset Name Parameter

```
public class NET_DVR_PRESET_NAME_ARRAY extends NET_DVR_CONFIG
{
    public NET_DVR_PRESET_NAME[] struPresetName = new
NET_DVR_PRESET_NAME[HCNetSDK.MAX_PRESET_NUM];

    public NET_DVR_PRESET_NAME_ARRAY()
    {
        for(int i = 0; i < HCNetSDK.MAX_PRESET_NUM; i++)
```

```
            {
                struPresetName[i] = new NET_DVR_PRESET_NAME();
            }
        }
}
```

Members

struPresetName

Preset name. Each array indicates one preset.

5.45 NET_DVR_PREVIEWINFO: Live View Settings

```
public class NET_DVR_PREVIEWINFO
{
    public int      lChannel;
    public int      dwStreamType;
    public int      dwLinkMode
    public int      bBlocked
    public int      bPassbackRecord
    public byte     byPreviewMode
    public byte     byProtoType
    public int      nRTSPPort
}
```

Members

lChannel

The analog channel number start from 1, the IP channel start number

dwStreamType

Stream type: 0-main stream, 1-sub stream, 2-stream 3, 3- virtual stream , and so on

dwLinkMode

Link mode: 0-TCP, 1- UDP, 2-Multicast, 3-RTP, 4-RTP/RTSP, 5-RSTP/HTTP

bBlocked

0-Non-blocking stream getting, 1-Blocking stream getting

bPassbackRecord

0-disable video passback, 1-enable video passback. back tracking when ANR disconnected- devices send the data automaticly after the network recovery between client and devices.(need devices support)

byPreviewMode

Live view mode: 0- normally live view, 1- delay live view

byProtoType

Application layer protocol: 0- private protocol, 1- RTSP.

nRTSPPort

RTSP port

5.46 NET_DVR_PTZCFG: PTZ Protocol Settings

```
public class NET_DVR_PTZCFG
{
    public int                              dwPtzNum;
    public NET_DVR_PTZ_PROTOCOL[]   struPtz = new
NET_DVR_PTZ_PROTOCOL[HCNetSDK.PTZ_PROTOCOL_NUM];

    public NET_DVR_PTZCFG(){
        for(int i=0; i<HCNetSDK.PTZ_PROTOCOL_NUM; i++){
            struPtz[i] = new NET_DVR_PTZ_PROTOCOL();
        }
    }
}
```

Members

dwPtzNum

Valid PTZ protocol number, start from 0 (total count= *dwPtzNum + 1*)

struPtz

Protocol information, the maximum number is 200

5.47 NET_DVR_PTZ_PROTOCOL: PTZ Protocol Information Settings

```
public class NET_DVR_PTZ_PROTOCOL
{
    public int          dwType;
    public byte[]       byDescribe = new byte[HCNetSDK.DESC_LEN];
}
```

Members

dwType

Protocol type

byDescribe

Protocol description

5.48 NET_DVR_QUERY_COUNTRYID_COND: Query by Country ID

```
public class NET_DVR_QUERY_COUNTRYID_COND extends NET_DVR_ADDR_QUERY_COND
{
    public int        wCountryID;
    public byte[]     szSvrAddr         = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public byte[]     szClientVersion = new byte[HCNetSDK.CLIENT_VERSION_LEN];
}
```

Members

wCountryID

Country ID

szSvrAddr

Server address

szClientVersion

Client version, such as: iVMS4500 V4.0.0.0 build20150112

5.49 NET_DVR_QUERY_COUNTRYID_RET: Result of Query by Country ID

```
public class NET_DVR_QUERY_COUNTRYID_RET extends NET_DVR_ADDR_QUERY_RET
{
    public byte[]   szResolveSvrAddr = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public byte[]   szAlarmSvrAddr   = new byte[HCNetSDK.MAX_DOMAIN_NAME];
}
```

Members

szResolveSvrAddr

Analysis server address

szAlarmSvrAddr

Alarm server address

5.50 NET_DVR_QUERY_DDNS_COND: HIDDNS Query and Diagnosis Condition

```
public class NET_DVR_QUERY_DDNS_COND extends NET_DVR_ADDR_QUERY_COND
{
    public byte[]   szResolveSvrAddr = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public byte[]   szDevNickName = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public byte[]   szDevSerial = new byte[HCNetSDK.SERIALNO_LEN];
    public byte[]   szClientVersion = new byte[HCNetSDK.CLIENT_VERSION_LEN];
}
```

Members

szResolveSvrAddr

Analysis server address

szDevNickName

Device domain name, valid when searching by device domain name

szDevSerial

Device serial number, valid when searching by serial No.

szClientVersion

Client version, such as iVMS4500 V4.0.0.0 build20150112

5.51 NET_DVR_QUERY_DDNS_RET: HIDDNS Query Result

```
public class NET_DVR_QUERY_DDNS_RET extends NET_DVR_ADDR_QUERY_RET
{
    public byte[]    szDevIP = new byte[HCNetSDK.SDK_MAX_IP_LEN];
    public int       wCmdPort;
    public int       wHttpPort;
}
```

Members

szDevIP

Device IP address

wCmdPort

Device SDK port No.

wHttpPort

Device HTTP port

5.52 NET_DVR_CHECK_DDNS_RET: HIDDNS Diagnosis Results

```
public class NET_DVR_CHECK_DDNS_RET extends NET_DVR_ADDR_QUERY_RET
{
    public byte                          byDevStatus;
    public int                           wRegionID;
    public NET_DVR_QUERY_DDNS_RET        struQueryRet = new NET_DVR_QUERY_DDNS_RET();
}
```

Members

byDevStatus

Device status, 0- normal, 1- Not found, 2- Off-line, 3- Not in the area

wRegionID

Region ID: 1- USA, 2- South America, 3- Asia, 4-China, 5-Europe, 6-Others

struQueryRet

Device IP and port query result

5.53 NET_DVR_QUERY_IPSERVER_COND: IPServer Query Condition

```
public class NET_DVR_QUERY_IPSERVER_COND extends NET_DVR_ADDR_QUERY_COND
{
    public byte[]    szResolveSvrAddr = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public int       wResolveSvrPort;
    public byte[]    szDevNickName = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public byte[]    szDevSerial = new byte[HCNetSDK.SERIALNO_LEN];
}
```

Members

szResolveSvrAddr

IPServer address

wResolveSvrPort

IPServer port, 7071

szDevNickName

Device name, valid when searching by device name

szDevSerial

Device serial No. , valid when searching by device serial No.

5.54 NET_DVR_QUERY_IPSERVER_RET: IPServer Query Result

```
public class NET_DVR_QUERY_IPSERVER_RET extends NET_DVR_ADDR_QUERY_RET
{
    public byte[]    szDevIP = new byte[HCNetSDK.SDK_MAX_IP_LEN];
    public int       wCmdPort;
}
```

Members

szDevIP

Device IP address

wCmdPort

Device SDK port No.

5.55 NET_DVR_RECORDDAY: All-day Record Settings

```
public class NET_DVR_RECORDDAY
{
    public short     wAllDayRecord;
    public byte      byRecordType;
}
```

Members

wAllDayRecord

Enable all-day recording? 0-No, 1-Yes

byRecordType

Recording type, 0- scheduled recording, 1-Motion detection, 2-Alarm, 3-Alarm or motion detection, 4-Alarm and motion detection, 5-Command, 6-VCA alarm recording, 10-PIR alarm, 11-Wireless alarm, 12-Emergency alarm, 13- Motion detection or PIR or wireless or emergency, 14-Intelligent traffic event, 15-Line crossing detection, 16-Intrusion, 17-Audio exception, 18-Scene change detection, 19-VCA detection (line crossing or intrusion or region entering or region exiting or face detection), 20-Face detection

5.56 NET_DVR_RECORDSCHED: Time Recording Paramater

```
public class NET_DVR_RECORDSCHED {
```

```
    public NET_DVR_SCHEDTIME    struRecordTime = new NET_DVR_SCHEDTIME();
    public byte                 byRecordType;
}
```

Members

struRecordTime

Recording time

byRecordType

Recording type, 0-Scheduled recording, 1-Motion detection, 2-Alarm, 3-Alarm or motion detection, 4-Alarm and motion detection, 5-Command, 6-VCA alarm recording, 10-PIR alarm, 11-Wireless alarm, 12-Emergency alarm, 13- Motion detection or PIR or wireless or emergency, 14-Intelligent traffic event, 15-Line crossing detection, 16-Intrusion, 17-Audio exception, 18-Scene change detection, 19-VCA detection (line crossing or intrusion or region entering or region exiting or face detection), 20-Face detection

5.57 NET_DVR_RECORD_V30: Recording Settings

```
public class NET_DVR_RECORD_V30 extends NET_DVR_CONFIG
{
    public int                  dwRecord;
    public NET_DVR_RECORDDAY[]  struRecAllDay = new NET_DVR_RECORDDAY[HCNetSDK.MAX_DAYS];
    public NET_DVR_RECORDSCHED[][]  struRecordSched = new
NET_DVR_RECORDSCHED[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];
    public int                  dwRecordTime;
    public int                  dwPreRecordTime;
    public int                  dwRecorderDuration;
    public byte                 byRedundancyRec;
    public byte                 byAudioRec;
    public byte                 byStreamType;
    public byte                 byPassbackRecord;
    public short                wLockDuration;
    public byte                 byRecordBackup;
    public byte                 bySVCLevel;
    public byte[]               byReserve = new byte[4];
    public NET_DVR_RECORD_V30(){
        for(int i=0; i<HCNetSDK.MAX_DAYS; i++){
            struRecAllDay[i] = new NET_DVR_RECORDDAY();
            for(int j=0; j<HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struRecordSched[i][j] = new NET_DVR_RECORDSCHED();
            }
        }
    }
}
```

Members

dwRecord

Enable scheduled recording? 0-No, 1-Yes

struRecAllDay

All-day recording settings

struRecordSched

Scheduled recording settings

dwRecordTime

Video delay time, 0-5 seconds, 1-10 seconds, 2-30 seconds, 3-1 minutes, 4-2 minutes, 5-5 minutes, 6-10 minutes

dwPreRecordTime

pre-record time: 0-disable, 1-5 seconds, 2-10 seconds, 3-15 seconds, 4-20 seconds, 5-25 seconds, 6-30 seconds, 7-0xffffffff (max pre-record time)

dwRecorderDuration

Max. time for keeping the video file, Unit: day, 0xffffffff indicates invalid

byRedundancyRec

Enable redundancy record for important data backup? 0- No (default), 1-Yes

byAudioRec

Enable audio record on video& audio recording mode? 0-No, 1-Yes

byStreamType

Stream type: 0-Main stream, 1-Sub stream, 3-Third stream

byPassbackRecord

Record copy back or not. 0-No, 1-Yes

wLockDuration

Record locked duration (hour). 0-Unlocked, 0xffff-Locked forever. The record which duration is longer than the locked duration won't be locked.

byRecordBackup

0-No backup, 1-Backup. Scheduled recording has no backup.

bySVCLevel

SVC frame extracting type: 0-No, 1-1/2, 2-3/4

byReserve

Reserved. Set as 0.

5.58 NET_DVR_RESOLVE_DEVICEINFO: Resolve Device Information

```
public class NET_DVR_RESOLVE_DEVICEINFO
{
    public byte[]   sGetIP = new byte[64];
    public int      dwPort;
}
```

Members

sGetIP

Device IP address

dwPort

Device port No.

Remarks

Resolve the device's IP address and port No. via domain name, and then call NET_DVR_Login_V30 to login.

5.59 NET_DVR_SCHEDTIME: Start Time and End Time Settings

```
public class NET_DVR_SCHEDTIME
{
    Public byte    byStartHour;
    Public byte    byStartMin;
    Public byte    byStopHour;
    Public byte    byStopMin;
}
```

Members

byStartHour

Start time, hour

byStartMin

Start time, minute

byStopHour

End time, hour

byStopMin

End time, minute

5.60 NET_DVR_SDKLOCAL_CFG: SDK Local Parameter

```
public class NET_DVR_SDKLOCAL_CFG
{
    public byte    byEnableAbilityParse;
    public byte[]  byProtectKey = new byte[128];
    public byte    byCompatibleType;
}
```

Members

byEnableAbilityParse

Enable capability analysis library? 0-No(default), 1-Yes

byProtectKey

Reserved. Set as 0.

byCompatibleType

Reserved. Set as 0.

5.61 NET_DVR_SEARCH_EVENT_PARAM: Search by Event Parameter

```
public class NET_DVR_SEARCH_EVENT_PARAM
{
    public short                wMajorType;
```

```
    public short                wMinorType;
    public NET_DVR_TIME         struStartTime = new NET_DVR_TIME();
    public NET_DVR_TIME         struEndTime = new NET_DVR_TIME();
    public byte                 byLockType;
    public int[]                wAlarmInNo = new int[128];
    public int[]                wMotDetChanNo = new int[64];
    public int[]                wBehaviorChanNo = new int[64];
    public int[]                dwVCAChanNo = new int[HCNetSDK.MAX_CHANNUM_V30-1];
}
```

Members

wMajorType

Major searching type. See table below:

```
enum _MAIN_EVENT_TYPE_{
   EVENT_MOT_DET            = 0,
   EVENT_ALARM_IN           = 1,
   EVENT_VCA_BEHAVIOR       = 2,
   EVENT_INQUEST            = 3,
   EVENT_VCA_DETECTION      = 4
}MAIN_EVENT_TYPE
```

EVENT_MOT_DET

Motion detection

EVENT_ALARM_IN

Alarm input

EVENT_VCA_BEHAVIOR

Behavior analysis

EVENT_INQUEST

Trial event

EVENT_VCA_DETECTION

VCA detection

wMinorType

Minor searching type. Currently, it only supports 0xffff, indicating all.

struStartTime

Search start time

struEndTime

Search end time

byLockType

Lock or not: 0xff-All, 0-Unlocked, 1-Locked

wAlarmInNo

Alarm input number by value. Adopts compact array.

Example: wAlarmInNo[0]==1&&wAlarmInNo[1]==2 means searching the event triggered by alarm input 1 and alarm input 2.

wMotDetChanNo

Motion detection channel. byMotDetChanNo[0]==1 means searching the event triggered by motion detection in channel 1, byMotDetChanNo[1]==1 means searching the event triggered by motion detection in

channel 2, etc.

wBehaviorChanNo

Behavior analysis channel by value. Adopts compact array.

Example: dwChanNo[0]==1&&dwChanNo[1]==2 means searching the behavior analysis of channel 1 and 2.

dwVCAChanNo

VCA detection channel by value. Adopts compact array.

Example: dwChanNo[0]==1&&dwChanNo[1]==2 means searching theVCA event of channel 1 and 2.

5.62 NET_DVR_SEARCH_EVENT_RET: Searched Result Information by Event

```
public class NET_DVR_SEARCH_EVENT_RET
{
    public short                wMajorType;
    public short                wMinorType;
    public NET_DVR_TIME         struStartTime = new NET_DVR_TIME();
    public NET_DVR_TIME         struEndTime = new NET_DVR_TIME();
    public int                  dwAlarmInNo;
    public int                  dwMotDetNo;
    public int                  dwBehaviorChanNo;
}
```

Members

wMajorType

Major type. See table below:

```
enum _MAIN_EVENT_TYPE_{
   EVENT_MOT_DET            = 0,
   EVENT_ALARM_IN           = 1,
   EVENT_VCA_BEHAVIOR       = 2,
   EVENT_INQUEST            = 3,
   EVENT_VCA_DETECTION      = 4
}MAIN_EVENT_TYPE
```

EVENT_MOT_DET

Motion detection

EVENT_ALARM_IN

Alarm input

EVENT_VCA_BEHAVIOR

Behavior analysis

EVENT_INQUEST

Trial event

EVENT_VCA_DETECTION

VCA detection

wMinorType

Different major types correspond to the change of minor type. Motion detection and alarm input have no

minor type. For other major types, see Table below:

Table 5.3 Major and Minor Type

Major Type Macro Definition	Value	Implication
EVENT_VCA_BEHAVIOR	2	Behavior analysis
Minor Type Macro Definition	**Value**	**Implication**
EVENT_TRAVERSE_PLANE	0	Line crossing
EVENT_ENTER_AREA	1	Region entrance. Support region rule.
EVENT_EXIT_AREA	2	Region exiting. Support region rule.
EVENT_INTRUSION	3	Intrusion. Support region rule.
EVENT_LOITER	4	Loitering. Support region rule.
EVENT_LEFT_TAKE	5	Object removal or unattended baggage. Support region rule.
EVENT_PARKING	6	Parking. Support region rule.
EVENT_RUN	7	Running. Support region rule.
EVENT_HIGH_DENSITY	8	People gathering. Support region rule.
EVENT_STICK_UP	9	Sticking scrip. Support region rule.
EVENT_INSTALL_SCANNER	10	Installing scanner. Support region rule.
EVENT_OPERATE_OVER_TIME	11	Operation timeout
EVENT_FACE_DETECT	12	Abnormal face
EVENT_LEFT	13	Unattended baggage
EVENT_TAKE	14	Object removal
EVENT_LEAVE_POSITION	15	Absence
EVENT_TRAIL_INFO	16	Tailing
EVENT_FALL_DOWN_INFO	19	Falling down
EVENT_OBJECT_PASTE	20	Sticking Scrip area
EVENT_FACE_CAPTURE_INFO	21	Normal face capture
EVENT_MULTI_FACES_INFO	22	Multiple faces
EVENT_AUDIO_ABNORMAL_INFO	23	Sudden Change of Sound Intensity

Major Type Macro Definition	Value	Implication
EVENT_INQUEST	3	Trial event
Minor Type Macro Definition	**Value**	**Implication**
INQUEST_START_INFO	0x1001	Trial starting information
INQUEST_STOP_INFO	0x1002	Trial ending information
INQUEST_TAG_INFO	0x1003	Tag information
INQUEST_SEGMENT_INFO	0x1004	Trial fragment status information

Major Type Macro Definition	Value	Implication
EVENT_VCA_DETECTION	4	VCA detection
Minor Type Macro Definition	**Value**	**Implication**
EVENT_VCA_TRAVERSE_PLANE	1	Line crossing detection
EVENT_FIELD_DETECTION	2	Intrusion detection
EVENT_AUDIO_INPUT_ALARM	3	Audio loss detection
EVENT_SOUND_INTENSITY_ALARM	4	Sudden Decrease of Sound Intensity Detection
EVENT_FACE_DETECTION	5	Face detection
EVENT_VIRTUAL_FOCUS_ALARM	6	Defocus detection
EVENT_SCENE_CHANGE_ALARM	7	Scene change detection
EVENT_PIR_ALARM	8	PIR alarm

struStartTime

Start time

struEndTime

End time.

dwAlarmInNo

Alarm input No.

dwMotDetNo

The channel No. triggered by motion detection event

dwBehaviorChanNo

The channel No. triggered by behavior analysis event

5.63 NET_DVR_SERIALSTART_V40: Serial Port Settings

```
public class NET_DVR_SERIALSTART_V40 extends NET_DVR_SERIAL_COND
{
    public int    dwSerialPort;
    public int    wPort;
}
```

Members

dwSerialPort

Serial port type, 1- 232, 2- 485

wPort

Port No.

5.64 NET_DVR_SERIAL_COND: Serial Port Sub Type

```
public class NET_DVR_SERIAL_COND{
}
```

Remarks

Serial port parameter sub type: NET_DVR_SERIALSTART_V40.

5.65 NET_DVR_SHELTER: Privacy Mask Settings

```
public class NET_DVR_SHELTER
{
    public short    wHideAreaTopLeftX;
    public short    wHideAreaTopLeftY;
    public short    wHideAreaWidth;
    public short    wHideAreaHeight;
}
```

Members

wHideAreaTopLeftX

X-coordinate of the area

wHideAreaTopLeftY

Y-coordinate axis of the area

wHideAreaWidth

Width of the area

wHideAreaHeight

Height of the area

Remarks

The SDK considered that the entire image size should be 704*576,there for, the width and height of privacy mask area should be less than 704 and 576.

5.66 NET_DVR_SHOWSTRINGINFO: Text Overlay for Single Word

```
public class NET_DVR_SHOWSTRINGINFO
{
    public int      wShowString;
    public int      wStringSize;
    public int      wShowStringTopLeftX;
    public int      wShowStringTopLeftY;
    public byte[]   sString = new byte[44];
}
```

Members

wShowString

Enable text overlay on preview image? 0-Disable, 1-Enable. The display area ranges totally 704*576, with single character size as 32*32

wStringSize

Text length, no more than 44 text character

wShowStringTopLeftX

X-coordinate position for text overlay

wShowStringTopLeftY

Y-coordinate position for text overlay

sString

Displaying content

5.67 NET_DVR_SHOWSTRING_V30: Text Overlay Settings

```
public class NET_DVR_SHOWSTRING_V30 extends NET_DVR_CONFIG
{
    public    NET_DVR_SHOWSTRINGINFO[] struStringInfo = new
NET_DVR_SHOWSTRINGINFO[HCNetSDK.MAX_STRINGNUM_V30];

    public NET_DVR_SHOWSTRING_V30(){
        for(int i = 0 ; i < HCNetSDK.MAX_STRINGNUM_V30; i++)
        {
            struStringInfo[i] = new NET_DVR_SHOWSTRINGINFO();
        }
    }
}
```

Members

struStringInfo

Displaying content

5.68 NET_DVR_SINGLE_DDNS: DDNS Server Information

```
public class NET_DVR_SINGLE_DDNS
{
    public byte[]    sUserName = new byte[HCNetSDK.NAME_LEN];
    public byte[]    sPassword = new byte[HCNetSDK.PASSWD_LEN];
    public byte[]    sDomainName = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public byte[]    sServerName = new byte[HCNetSDK.MAX_DOMAIN_NAME];
    public int        wDDNSPort;
    public byte[]    byRes = new byte[16];
}
```

Members

sUsername

DDNS account name

sPassword

DDNS account password

sDomainName

Domain name

sServerName

DDNS server address (IP address or domain name)

wDDNSPort

DDNS port

byRes

Reserved. Set as 0.

5.69 NET_DVR_TIME: Time Settings

```
public class NET_DVR_TIME
{
    public int    dwYear;
    public int    dwMonth;
    public int    dwDay;
    public int    dwHour;
    public int    dwMinute;
    public int    dwSecond;
}
```

Members

dwYear

Year

dwMonth

Month

dwDay

Day

dwHour

Jour

dwMinute

Minute

dwSecond

Second

5.70 NET_DVR_UPNP_NAT_STATE: UPNP Port Mapping Status

```
public class NET_DVR_UPNP_NAT_STATE
{
    public NET_DVR_UPNP_PORT_STATE[]    strUpnpPort = new NET_DVR_UPNP_PORT_STATE[12];
}
```

Members

strUpnpPort

Port mapping state: strUpnpPort[0]- web server port number, strUpnpPort[1]- management port number, strUpnpPort[2]- rtsp port number

5.71 NET_DVR_UPNP_PORT_STATE: UPNP Port Mapping Status

```
public class NET_DVR_UPNP_PORT_STATE
{
    public int                  dwEnabled;
    public int                  wInternalPort;
    public int                  wExternalPort;
    public int                  dwStatus;
    public NET_DVR_IPADDR       struNatExternalIp;
    public NET_DVR_IPADDR       struNatInternalIp
}
```

Members

dwEnabled

Enable the port to be mapped?

wInternalPort

The No. of port before mapped

wExternalPort

The No. of port after mapped

dwStatus

Port mapping status: 0- not yet effective; 1- not yet effective: the mapped source port and destination port should be the same; 2- not yet effective: the mapped port number is already in use; 3- have taken effect

struNatExternalIp

The external address after being mapped

struNatInternalIp

NAT router LAN IP address

5.72 NET_DVR_USER_INFO_V30: User Settings for Single User

```
public class NET_DVR_USER_INFO_V30
{
    public byte[]               sUserName = new byte[HCNetSDK.NAME_LEN];
    public byte[]               sPassword = new byte[HCNetSDK.PASSWD_LEN];
    public byte[]               byLocalRight = new byte[HCNetSDK.MAX_RIGHT];
    public byte[]               byRemoteRight = new byte[HCNetSDK.MAX_RIGHT];
    public byte[]               byNetPreviewRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byLocalPlaybackRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byNetPlaybackRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byLocalRecordRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byNetRecordRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byLocalPTZRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byNetPTZRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]               byLocalBackupRight = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public NET_DVR_IPADDR       struUserIP = new NET_DVR_IPADDR();
```

```
    public byte[]           byMACAddr = new byte[HCNetSDK.MACADDR_LEN];
    public byte             byPriority;
    public byte[]           byRes = new byte[17];
}
```

Members

sUserName

User name

sPassword

Password

byLocalRight

Local privilege settings array

0: Local PTZ control array

1: Local manual record array

2: Local playback array

3: Local configuration array

4: Local log & status query array

5: Local advanced settings (upgrade, format) array

6: Local parameter query array

7: Local analog & IP camera management array

8: Local backup array

9: Local shut down/reboot

byRemoteRight

Remote privilege settings array

0: Remote PTZ control array

1: Remote manual record array

2: Remote playback array

3: Remote configuration array

4: Remote log & status query array

5: Remote advanced settings (upgrade, format) array

6: Remote start voice talk array

7: Remote preview array

8: Remote alarm upload to center, alarm output array

9: Remote control local output array

10: Remote serial port control array

11: Reserved array

12: Remote analog & IP camera management array

13: Remote shut down/reboot

byNetPreviewRight

Remote preview channel, 0-enable, 1-disable

byLocalPlaybackRight

Local playback channel, 0-enable, 1-disable

byNetPlaybackRight

Remote playback channel, 0-enable, 1-disable

byLocalRecordRight

Local record channel, 0-enable, 1-disable

byNetRecordRight

Remote record channel, 0-enable, 1-disable

byLocalPTZRight

Local PTZ channel, 0-enable, 1-disable

byNetPTZRight

Remote PTZ channel, 0-enable, 1-disable

byLocalBackupRight

Local backup channel, 0-enable, 1-disable

struUserIP

User IP (0 stands for no IP restriction), please refer to NET_DVR_IPADD

byMACAddr

MAC address

byPriority

Prioroty settings: 0xff-Disable, 0-Low, 1-Mid, 2-High

Disable: No priority settings

Low: Default privileges including local/remote playback, log & status query, reboot/shut down.

Mid: Include local/remote PTZ control, manual record, voice talk, playback, log & status query, reboot/shut down, log backup and remote preview.

High: Administrator

byRes

Reserved. Set as 0.

5.73 NET_DVR_USER_V30: User Settings

```
public class NET_DVR_USER_V30 extends NET_DVR_CONFIG
{
    public NET_DVR_USER_INFO_V30[] struUser = new
NET_DVR_USER_INFO_V30[HCNetSDK.MAX_USERNUM_V30];
    public NET_DVR_USER_V30()
    {
        for(int i = 0; i < HCNetSDK.MAX_USERNUM_V30; i++)
        {
            struUser[i] = new NET_DVR_USER_INFO_V30();
        }
    }
}
```

Members

struUser

User information. Maximun: 32 users.

5.74 NET_DVR_VICOLOR: Time Duration Image Parameter

```
public class NET_DVR_VICOLOR
{
    public NET_DVR_COLOR[]        struColor = new NET_DVR_COLOR[HCNetSDK.MAX_TIMESEGMENT_V30];
    public NET_DVR_SCHEDTIME[]  struHandleTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_TIMESEGMENT_V30];
    public NET_DVR_VICOLOR()
    {
        for(int i = 0; i < HCNetSDK.MAX_TIMESEGMENT_V30; i++)
        {
            struColor[i] = new NET_DVR_COLOR();
            struHandleTime[i] = new NET_DVR_SCHEDTIME();
        }
    }
}
```

Members

struColor

Image parameter. Reserved.

struHandleTime

Handle time duration. Not supported currently. Reserved.

5.75 NET_DVR_VIDEOEFFECT: Video Display Parameter

```
public class NET_DVR_VIDEOEFFECT
{
    public int    iBrightValue;
    public int    iContrastValue;
    public int    iSaturationValue;
    public int    iHueValue;
}
```

Members

iBrightValue

Brightness. Value range: [1,10]

iContrastValue

Contrast. Value range: [1,10]

iSaturationValue

Saturation. Value range: [1,10]

iHueValue

Hue. Value range: [1,10]

5.76 NET_DVR_VILOST_V30: Video Loss Alarm Settings

```
public class NET_DVR_VILOST_V30
{
    public byte                                byEnableHandleVILost;
    public NET_DVR_HANDLEEXCEPTION_V30 struVILostHandleType = new NET_DVR_HANDLEEXCEPTION_V30();
    public NET_DVR_SCHEDTIME[][]               struAlarmTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];
    public NET_DVR_VILOST_V30(){
        for(int i = 0; i < HCNetSDK.MAX_DAYS; i++)
        {
            for(int j = 0; j < HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struAlarmTime[i][j] = new NET_DVR_SCHEDTIME();
            }
        }
    }
}
```

Members

byEnableHandleVILost

Handle the video loss alarm? 0-No, 1-Yes

strVILostHandleType

Handling method

struAlarmTime

Arming time

5.77 NET_DVR_WIFIETHERNET: Wireless Network Port Settings

```
public class NET_DVR_WIFIETHERNET
{
    public byte[]    sIpAddress = new byte[16];
    public byte[]    sIpMask = new byte[16];
    public byte[]    byMACAddr = new byte[HCNetSDK.MACADDR_LEN];
    public int       dwEnableDhcp;
    public int       dwAutoDns;
    public           byte[] sFirstDns = new byte[16];
    public           byte[] sSecondDns = new byte[16];
    public           byte[] sGatewayIpAddr = new byte[16];
}
```

Members

sIpAddress

Device IP address

sIpMask

IP mask

byMACAddr

MAC address, read only.

dwEnableDhcp

Enable DHCP: 0- No, 1- Yes

dwAutoDns

Get DNS automatically when DHCP is enabled? 0-No, 1-Yes. For wire network, it gets DNS automatically.

sFirstDns

First DNS domain name

sSecondDns

Second DNS domain name

sGatewayIpAddr

IP address of gateway

5.78 NET_DVR_WIFI_CFG: WiFi Settings

```
public class NET_DVR_WIFI_CFG extends NET_DVR_CONFIG
{
    NET_DVR_WIFIETHERNET  struetherNet = new NET_DVR_WIFIETHERNET();
    public byte[]         sEssid = new byte[HCNetSDK.IW_ESSID_MAX_SIZE];
    public int            dwMode;
    public int            dwSecurity;
    public WEP            wep = new WEP();
    public WPA_PSK        wpa_psk = new WPA_PSK();
    public WPA_WPA2       wpa_wpa2 = new WPA_WPA2();
}
```

Members

struetherNet

Wifi network interface

sEssid

SSID

dwMode

Working mode: 0- mange mode, 1- ad-hoc mode

dwSecurity

Security mode: 0- no encryption, 1- WEP, 2- WPA-personal, 3- WPA-enterprise, 4- WPA2-personal, 5- WPA2-enterprise

wep

WEP encryption parameter

wpa_psk

WPA-personal/WPA2-personal encryption parameter

wpa_wpa2

WPA-enterprise/WPA2-enterpris encryption parameter

5.79 NET_DVR_WIFI_CONNECT_STATUS: WiFi Connection Status

```
public class NET_DVR_WIFI_CONNECT_STATUS extends NET_DVR_CONFIG
{
    public byte    byCurStatus;
    public int     dwErrorCode;
}
```

Members

byCurStatus

WiFi connections status: 1-Connected, 2-Unconnected, 3-Connecting

dwErrorCode

Error code, valid when byCurStatus==2, 1-Incorrect user name and password, 2-The router is not existed. 3-Unknown

5.80 NET_DVR_WORKSTATE_V30: Device Working Status Information

```
public class NET_DVR_WORKSTATE_V30
{
    public int                                  dwDeviceStatic;
    public NET_DVR_DISKSTATE[]                  struHardDiskStatic = new
NET_DVR_DISKSTATE[HCNetSDK.MAX_DISKNUM_V30];
    public NET_DVR_CHANNELSTATE_V30[]     struChanStatic = new
NET_DVR_CHANNELSTATE_V30[HCNetSDK.MAX_CHANNUM_V30];
    public byte[]                               byAlarmInStatic = new byte[HCNetSDK.MAX_ALARMIN_V30];
    public byte[]                               byAlarmOutStatic = new byte[HCNetSDK.MAX_ALARMOUT_V30];
    public int                                  dwLocalDisplay;
    public byte[]                               byAudioChanStatus = new byte[HCNetSDK.MAX_AUDIO_V30];

    public NET_DVR_WORKSTATE_V30(){
        for(int i=0; i<HCNetSDK.MAX_DISKNUM_V30; i++){
            struHardDiskStatic[i] = new NET_DVR_DISKSTATE();
        }
        for(int i=0; i<HCNetSDK.MAX_CHANNUM_V30; i++){
            struChanStatic[i] = new NET_DVR_CHANNELSTATE_V30();
        }
    }
}
```

Members

dwDeviceStatic

Device status: 0-Normal, 1-the occupancy of CPU is too high, more than 85%, 2-Error in hardware, e.g. the serial ports don't work

struHardDiskStatic

HD status

struChanStatic

Channel Status

byAlarmInStatic

Alarm input status: 0-No alarm, 1-Alarm

byAlarmOutStatic

Alarm output status: 0-No alarm output, 1-Alarm output

dwLocalDisplay

Local display status: 0-Normal, 1-Abnormal

byAudioChanStatus

Audio channel status: 0-Unused, 1-on using, 0xff-Invalid

5.81 NET_DVR_ZEROCHANCFG: Zero Channel Compression Settings

```
public class NET_DVR_ZEROCHANCFG extends NET_DVR_CONFIG
{
    public byte     byEnable;
    public int      dwVideoBitrate;
    public int      dwVideoFrameRate;
}
```

Members

byEnable

Enable zero channel encoding, 0-No, 1-Yes

dwVideoBitrate

Bit rate: 0-Reserved, 1-16K (reserved), 2-32K, 3-48k, 4-64K, 5-80K, 6-96K, 7-128K, 8-160k, 9-192K, 10-224K, 11-256K, 12-320K, 13-384K, 14-448K, 15-512K, 16-640K, 17-768K, 18-896K, 19-1024K, 20-1280K, 21-1536K, 22-1792K, 23-2048K

dwVideoFrameRate

Frame rate: 0-All, 1-1/16, 2-1/8, 3-1/4, 4-1/2, 5-1, 6-2, 7-4, 8-6, 9-8, 10-10, 11-12, 12-16, 13-20, 14-15, 15-18, 16-22

5.82 NET_IPC_AUX_ALARMCFG: Auxiliary Alarm Parameters

```
public class NET_IPC_AUX_ALARMCFG extends NET_DVR_CONFIG
{
    public NET_IPC_SINGLE_AUX_ALARMCFG[] struAlarm = new
NET_IPC_SINGLE_AUX_ALARMCFG[HCNetSDK.MAX_AUX_ALARM_NUM];

    public NET_IPC_AUX_ALARMCFG()
    {
        for(int i = 0; i < HCNetSDK.MAX_AUX_ALARM_NUM; i++)
        {
            struAlarm[i] = new NET_IPC_SINGLE_AUX_ALARMCFG();
        }
```

```
    }
}
```

Members

struAlarm

Auxiliary alarm settings

5.83 NET_IPC_CALLHELP_ALARMCFG: Emergency Alarm Parameter

```
public class NET_IPC_CALLHELP_ALARMCFG {
    public byte                                 byAlarmHandle;
    public NET_DVR_HANDLEEXCEPTION_V30 struAlarmHandleType = new NET_DVR_HANDLEEXCEPTION_V30();
    public byte[]                               byRelRecordChan = new byte[HCNetSDK.MAX_CHANNUM_V30];
}
```

Members

byAlarmHandle

Whether to handle the alarm: 0-No, 1-Yes

struAlarmHandleType

Handling type

byRelRecordChan

The channel to record triggered by the alarm, if the value is 1, it will trigger the channel to record. E.g. byRelRecordChan[0]==1, means to trigger the channel no.1, byRelRecordChan[1]==1, means to trigger the channel no.2, and so forth

5.84 NET_IPC_PIR_ALARMCFG: PIR Alarm Parameter

```
public class NET_IPC_PIR_ALARMCFG
{
    public byte[]                               byAlarmName = new byte[HCNetSDK.NAME_LEN];
    public byte                                 byAlarmHandle;
    public NET_DVR_HANDLEEXCEPTION_V30  struAlarmHandleType = new
NET_DVR_HANDLEEXCEPTION_V30();
    public byte[]                               byRelRecordChan = new byte[HCNetSDK.MAX_CHANNUM_V30];
    public NET_DVR_SCHEDTIME[][]                struAlarmTime = new
NET_DVR_SCHEDTIME[HCNetSDK.MAX_DAYS][HCNetSDK.MAX_TIMESEGMENT_V30];

    public NET_IPC_PIR_ALARMCFG()
    {
        for(int i = 0; i < HCNetSDK.MAX_DAYS; i++)
        {
            for(int j = 0; j < HCNetSDK.MAX_TIMESEGMENT_V30; j++)
            {
                struAlarmTime[i][j] = new NET_DVR_SCHEDTIME();
            }
```

```
        }
    }
}
```

Members

byAlarmName

Alarm name

byAlarmHandle

Whether to handle the alarm: 0-No, 1-Yes

struAlarmHandleType

Handling type

byRelRecordChan

The channel to record triggered by the alarm, if the value is 1, it will trigger the channel to record. E.g. byRelRecordChan[0]==1, means to trigger the channel no.1, byRelRecordChan[1]==1, means to trigger the channel no.2, and so forth

struAlarmTime

Arming schedule. 7 days each week and 8 period each day.

5.85 NET_IPC_SINGLE_AUX_ALARMCFG: Single Auxiliary Alarm Settings

```
public class NET_IPC_SINGLE_AUX_ALARMCFG
{
    public byte                                 byAlarmType;
    public NET_IPC_PIR_ALARMCFG                 struPIRAlarm = new NET_IPC_PIR_ALARMCFG();
    public NET_IPC_SINGLE_WIRELESS_ALARMCFG[] struWirelessAlarm = new
NET_IPC_SINGLE_WIRELESS_ALARMCFG[HCNetSDK.MAX_WIRELESS_ALARM_NUM];
    public NET_IPC_CALLHELP_ALARMCFG            struCallHelpAlarm = new NET_IPC_CALLHELP_ALARMCFG();

    public NET_IPC_SINGLE_AUX_ALARMCFG()
    {
        for(int i = 0; i < HCNetSDK.MAX_WIRELESS_ALARM_NUM; i++)
        {
            struWirelessAlarm[i] = new NET_IPC_SINGLE_WIRELESS_ALARMCFG();
        }
    }
}
```

Members

byAlarmType

Alarmer type, defined as below:

```
enum _IPC_AUX_ALARM_TYPE_{
   IPC_AUXALARM_UNKNOW = 0,
   IPC_AUXALARM_PIR        = 1,
   IPC_AUXALARM_WIRELESS = 2,
   IPC_AUXALARM_CALLHELP = 3
}IPC_AUX_ALARM_TYPE
```

IPC_AUXALARM_UNKNOW
Unknown
IPC_AUXALARM_PIR
PIR alarm
IPC_AUXALARM_WIRELESS
Wireless alarm
IPC_AUXALARM_CALLHELP
Emergency alarm

struPIRAlarm
PIR alarm parameter
struWirelessAlarm
Wireless alarm parameter
struCallHelpAlarm
Emergency alarm parameter

5.86 NET_IPC_SINGLE_WIRELESS_ALARMCFG: Single Wireless Alarm Parameter

```
public class NET_IPC_SINGLE_WIRELESS_ALARMCFG
{
    public byte[]                       byAlarmName = new byte[HCNetSDK.NAME_LEN];
    public byte                         byAlarmHandle;
    public byte                         byID;
    public NET_DVR_HANDLEEXCEPTION_V30  struAlarmHandleType = new
NET_DVR_HANDLEEXCEPTION_V30();
    public byte[]                       byRelRecordChan = new byte[HCNetSDK.MAX_CHANNUM_V30];
}
```

Members

byAlarmName
Alarm name
byAlarmHandle
Whether to handle the alarm or not: 0-No, 1-Yes
byID
Wireless alarm ID, value range: 1~8
struAlarmHandleType
Handling mode
byRelRecordChan
The channel to record triggered by the alarm, if the value is 1, it will trigger the channel to record. E.g. byRelRecordChan[0]==1, means to trigger the channel No.1, byRelRecordChan[1]==1, means to trigger the channel No.2.

5.87 WEP: WEP Encryption Parameter

```
public class WEP
{
    public int          dwAuthentication;
    public int          dwKeyLength;
    public int          dwKeyType;
    public int          dwActive;
    public byte[][]     sKeyInfo = new
byte[HCNetSDK.WIFI_WEP_MAX_KEY_COUNT][HCNetSDK.WIFI_WEP_MAX_KEY_LENGTH];
}
```

Members

dwAuthentication

Permission type: 0-Open, 1-Shared

dwKeyLength

Key length: 0-64 bits, 1-128 bits, 2-152 bits

dwKeyType

Key type: 0-Hex, 1-ASCII

dwActive

Activated key number. 0 indicates to activate the No.1 key, etc.

sKeyInfo

Key information

5.88 WPA_PSK: WPA_PSK Encryption Parameter

```
public class WPA_PSK
{
    public int          dwKeyLength;
    public byte[]       sKeyInfo = new byte[HCNetSDK.WIFI_WPA_PSK_MAX_KEY_LENGTH];
    public byte         byEncryptType;
}
```

Members

dwKeyLength

Key length: 8 to 63 ASCII characters

sKeyInfo

Key information

byEncryptType

Encryption type in WPA-personal/WPA2-personal mode: 0-AES, 1-TKIP

5.89 WPA_WPA2: WPA_WPA2 Encryption Parameter

```
public class WPA_WPA2
```

```
{
    public byte         byEncryptType;
    public byte         byAuthType;
    public EAP_TTLS     strueapTtls = new EAP_TTLS();
    public EAP_PEAP     strueapPeap = new EAP_PEAP();
    public EAP_TLS      strueapTls = new EAP_TLS();
}
```

Members

byEncryptType

Encryption type: 0-AES, 1-TKIP

sKeyInfo

Authentication type: 0-EAP_TTLS, 1-EAP_PEAP, 2-EAP_TLS

EAP_TTLS

EAP_TTLS authentication parameter

EAP_PEAP

EAP_PEAP authentication parameter

EAP_TLS

EAP_TLS authentication parameter

5.90Country Code

Country	Country Code
Europe	**100**
Andorra	101
Austria	102
Albania	103
Ireland	104
Estonia	105
Iceland	106
Belarus	107
Bulgaria	108
Poland	109
Bosnia	110
Belgium	111
Germany	112
Denmark	113
Russia	114
France	115
Finland	116
Holland	117
Czech	118
Croatia	119

Latvia	120
Lithuania	121
Liechtenstein	122
Romania	123
Macedonia	124
Malta	125
Luxembourg	126
Monaco	127
Moldova	128
Norway	129
Serbia	130
Portugal	131
Sweden	132
Switzerland	133
Slovak	134
Slovenia	135
San marino	136
Ukraine	137
Spain	138
Greece	139
Hungary	140
Italy	141
United Kingdom	142
Europe Other	143
Asia	**200**
Afghanistan	201
United Arab Emirates	202
Oman	203
Azerbaijan	204
Pakistan	205
Palestine	206
Bahrain	207
Bhutan	208
North Korea	209
Timor	210
Philippines	211
Georgia	212
Kazakhstan	213
Korea	214
Kirgizstan	215
Cambodia	216
Qatar	217

Kuwait	218
Laos	219
Lebanon	220
Maldives	221
Malaysia	222
Mongolia	223
Bangladesh	224
Myanmar	225
Nepal	226
Japan	227
Cyprus	228
Saudi Arabia	229
Srilanka	230
Tajikistan	231
Thailand	232
Turkey	233
Turkmenistan	234
Brunei	235
Uzbekistan	236
Singapore	237
Syria	238
Armenia	239
Yemen	240
Iran	241
Iraq	242
Israel	243
India	244
Indonesia	245
Jordan	246
Vietnam	247
China	248
Asia Other	249
America	**300**
Argentina	301
Antigua and Barbuda	302
Barbados	303
Bolivia	304
Brazil	305
Dominica	306
Ecuador	307
Cuba	308
Colombia	309

Grenada	310
Guyana	311
Canada	312
Peru	313
United States	314
Mexico	315
Surinam	316
Saint-Lucia	317
Trinidad and Tobago	318
Uruguay	319
Venezuela	320
Jamaica	321
Chile	322
Bahamas	323
America Other	324
Africa	**400**
Algeria	401
Egypt	402
Ethiopia	403
Angola	404
Benin	405
Botswana	406
Burkina Faso	407
Burundi	408
Equatorial Guinea	409
Togo	410
Eritrea	411
Verde	412
Gambia	413
Congo	414
Congo-Kinshasa	415
Djibouti	416
Guinea	417
Guinea-Bissau	418
Gabon	419
Ghana	420
Zimbabwe	421
Cameroon	422
Comoros	423
Cote d'Ivoire	424
Kenya	425
Lesotho	426

Liberia	427
Libya	428
Rwanda	429
Madagascar	430
Mali	431
Mauritius	432
Mauritania	433
Morocco	434
Mozambique	435
Namibia	436
South Africa	437
Niger	438
Nigeria	439
Sierra Leone	440
Senegal	441
Seychelles	442
Sao Tome and Principe	443
Sudan	444
Somali	445
Tanzania	446
Tunisia	447
Uganda	448
Zambia	449
Chad	450
Central African Republic	451
Africa Other	452
Oceania	**500**
Australia	501
Papua New Guinea	502
Fiji	503
Cook Islands	504
Samoa	505
Micronesia	506
Nauru	507
Tonga	508
Vanuatu	509
New Zealand	510
Oceania Other	511