kdnetsdk help

Introduction

With the kdnetsdk, you can manage channels/events/networks/recordings/captures, configure storage/system/maintenance/intelligent analysis/archive/intelligent search settings, and subscribe to notifications of channel status changes, network changes, and alarm generation and clearing.

System Support

Applicable Operating Systems
windows x86, x64
linux 32-/64-bit, supporting his3536, his3536c, skylake, centos
Thread Safety
netsdk interface thread safety

Macro Definition

Macro	Value	Description
NET_MAXLEN_8	8	
NET_MAXLEN_16	16	
NET_MAXLEN_32	32	
NET_MAXLEN_64	64	
NET_MAXLEN_128	128	
NET_MAXLEN_256	256	
NET_MAXLEN_512	512	
NET_NVR_USER_NAME_MAX_LEN	64	Maximum length of an NVR username, that is, 32 characters
NET_NVR_USER_PWD_MAX_LEN	16	Maximum length of an NVR password, that is, 16 characters
NET_NVR_DISK_MAX_NUM	50	Maximum number of disks on an NVR
NET_NVR_USER_MAX_NUM	16	Maximum number of NVR users
NET_NVR_ETH_MAX_NUM	12	Maximum number of NICs in an NVR
NET_PER_GET_SEARCHED_DEV_MAX_NUM	32	Maximum number of devices that can be searched at a time
NET_PER_ADD_DEV_MAX_NUM	64	Maximum number of devices that can be added at a time
NET_PER_DEL_DEV_MAX_NUM	64	Maximum number of devices that can be deleted at a time
NET_PER_UPGRADE_DEV_MAX_NUM	64	Maximum number of devices that can be upgraded at a time
NET_PER_UPLOAD_UPGRADE_FILE_MAX_SIZE	(1 * 1024 * 1024)	Maximum size of an upgrade package

Macro	Value	Description
NET_UPLOAD_UPGRADE_FILE_CGI_MAX_SIZE	(2 * 1024 * 1024)	Maximum size of an upgrade package after being encoded or decoded
NET_PER_GET_NETDISK_MAX_NUM	16	Maximum number of network storage units that can be searched at a time
NET_PROTO_MAX_NUM	8	Maximum number of protocols
NET_PER_GET_CHNLIST_MAX_NUM	32	Maximum number of channels that can be searched at a time
NET_CHNGROUP_CHNITEM_MAX_NUM	64	Maximum number of members in a channel group
NET_CHNGROUP_MAX_NUM	16	Maximum number of channel groups
NET_FRAME_STYLE_GRID_MAX_NUM	128	Maximum number of viewing windows in a screen layout
NET_BIND_MAX_NUM	128	Maximum number of devices that can be bound to a screen layout
NET_PIP_MAX_NUM	16	Maximum number of live viewings on configuration interfaces
NET_DEC_STATE_MAX_NUM	256	Maximum number of windows whose decoding status can be queried at a time
NET_PER_GET_CHN_CAP_MAX_NUM	128	Maximum number of channels whose capability can be searched at a time
NET_PER_GET_CHNPERM_MAX_NUM	256	Maximum number of channels whose permission settings can be searched at a time
NET_VIDEOENC_RECMODE_MAX_NUM	8	Maximum number of video encoding bitrate modes
NET_VIDEOENC_RESOLUTION_MAX_NUM	32	Maximum number of video encoding resolutions

Macro	Value	Description
NET_VIDEOENC_ENCTYPE_MAX_NUM	8	Maximum number of video encoding types
NET_PER_CHN_ENCCHN_MAX_NUM	8	Maximum number of encoding channels of a channel
NET_AUDIOENC_ENCTYPE_MAX_NUM	8	Maximum number of audio encoding types
NET_LOG_MAINTYPR_MAX_NUM	4	Maximum number of log categories
NET_LOG_SUBTYPR_MAX_NUM	20	Maximum number of log types
NET_PER_GET_LOG_MAX_NUM	100	Maximum number of log entries that can be searched at a time
NET_CHN_ALIAS_LIST_MAX_NUM	128	Maximum number of channel names
NET_DISPLAY_PORT_MAX_NUM	4	Maximum number of display ports
NET_DISPLAY_PORT_RESOLUTION_MAX_NUM	16	Maximum number of resolutions of display ports
NET_PER_CHNGETRECLIST_MAX_NUM	150	Maximum number of record entries that can be obtained before playback at a time
NET_PER_CHNPLAYBACKLIST_MAX_NUM	32	Maximum number of channels that can be involved in a multi-record playback on the NVR GUI
NET_PER_CHNGETMONTHREC_MAX_NUM	32	Maximum number of channels displayed on a month view at a time
NET_GROUP_DISKS_MAX_NUM	64	Maximum number of disk IDs in a disk group
NET_GROUP_MAX_NUM	16	Maximum number of disk groups
NET_EXDISK_MAX_NUM	16	Maximum number of external USB storage units
NET_INTERNALDISK_MAX_NUM	16	Maximum number of HDDs in an NVR

Macro	Value	Description
NET_RECTIMEPLAN_MAX_NUM	8	Maximum number of recording schedules
NET_PRE_DAY_RECTIMEPLAN_MAX_NUM	8	Maximum number of recording durations in a day in a recording schedule
NET_ALARM_GUARD_TIME_SEG_MAX_NUM	8	Maximum number of alarming durations in a day
NET_ALARM_GUARD_TIME_MAX_NUM	8	Maximum number of alarming durations
NET_ALARM_IN_ID_MAX_NUM	20	Maximum number of alarm input IDs
NET_ALARM_IN_COPY_MAX_NUM	64	Maximum number of alarm inputs to which alarm input settings can be copied
NET_ALARM_LINK_OUT_MAX_NUM	256	Maximum number of alarm linkage target devices
NET_ALARM_LINK_PTZ_MAX_NUM	256	Maximum number of alarm linkage PTZ target devices
NET_SYS_ALARM_CFG_MAX_NUM	16	Maximum number of exception items
NET_COPYCHNRECPLAN_MAX_NUM	256	Maximum number of devices to which recording schedules can be copied
NET_HOLIDAY_MAX_NUM	12	Maximum number of holidays
NET_DISKGROUP_CHN_MAX_NUM	128	Maximum number of members in a disk group
NET_PLAYBACK_MAX_NUM	16	Maximum number of channels that can be involved in a multi-record playback on the NVR Web or an NVR client
NET_PLAYBACK_ITEM_MAX_NUM	4	Maximum number of video or audio streams in a channel playback
NET_COPYDISKQUOTA_MAX_NUM	512	Maximum number of channels to which storage quota settings can be copied

Macro	Value	Description
NET_PER_QUERYSNAPPIC_MAX_NUM	300	Maximum number of captures that can be searched at a time
NET_PER_QUERYRECTAG_MAX_NUM	200	Maximum number of record tags that can be searched at a time
NET_REAL_LOG_MAX_NUM	100	Maximum number of real-time log entries that can be displayed
NET_REAL_STATE_MAX_NUM	15	Maximum number of system status entries that can be obtained at a time
NET_COMPOSITE_CHN_MAX_NUM	16	Maximum number of virtual channels
NET_DEVOSD_MAX_NUM	16	Maximum number of OSD texts
NET_USERLEVEL_MAX_NUM	3	Maximum number of user levels
NET_PER_GET_CHN_ALARMCFG_MAX_NUM	32	Maximum number of channels whose event settings can be obtained at a time
NET_CHNID_AUTO	0	Automatically assign a channel ID
NET_MAX_SMART_ATTR_ITEM_NUM	32	Maximum number of attributes in a SMART test
NET_MAX_ERROR_SECTOR_NUM	50	Maximum number of incorrect LBAs that can be detected in a bad-sector detection
NET_UPNP_PORT_MAP_MAX_NUM	10	Maximum number of mapping ports involved in UPnP
NET_NVR_VIDEO_ENC_MAX_NUM	4	Maximum number of NVR video encoding channels
NET_NVR_AUDIO_ENC_MAX_NUM	4	Maximum number of NVR audio encoding channels

Macro	Value	Description
NET_PER_GET_CHNRECSTATE_MAX_NUM	32	Maximum number of channels whose recording status can be queried at a time
NET_PER_GET_ALARM_STATE_MAX_NUM	128	Maximum number of alarm status entries that can be obtained at a time
NET_CLOUD_QR_CODE_MAX_NUM	4	Maximum number of cloud-service QR codes
NET_PER_DEV_ACTIVE_MAX_NUM	32	Maximum number of devices that can be involved in a batch activation
NET_PER_LOCK_REC_MAX_NUM	100	Maximum number of records that can be locked at a time
NET_NVR_NTY_EVENT_MAX_NUM	50	Maximum number of NVR event notifications
NET_GB28181_PRECHN_ENC_MAX_NUM	4	Maximum number of encoding IDs of a SIP channel
NET_GB28181_CHN_MAX_NUM	64	Maximum number of SIP channels
NET_SIM_CARD_MAX_NUM	4	Maximum number of SIM cards
NET_GB28181_REG_PLAT_MAX_NUM	4	Maximum number of SIP platforms
NET_PUBSEC_REG_PLAT_MAX_NUM	4	Maximum number of VIIDs
NET_AIU_RECENT_SNAP_PIC_GROUP_MAX_NUM	10	Maximum number of recent intelligent capture groups
NET_AIU_RECENT_SNAP_PIC_ITEM_MAX_NUM	5	Maximum number of captures in a recent intelligent capture group
NET_PER_GET_AIU_CHN_CAP_MAX_NUM	32	Maximum number of channels whose intelligence capability can be obtained at a time
NET_FACE_MAX_NUM	16	Maximum number of faces
NET_CAR_PLATE_MAX_NUM	32	Maximum number of vehicle license plates

Macro	Value	Description
NET_PIC_UPLOAD_PUBSEC_PLAT_MAX_NUM	4	Maximum number of platforms to which pictures/videos can be uploaded
NET_MANUALEVENT_MAXNUM	32	Maximum number of custom events in intelligent analysis
NET_IPADDRFILTERLIST_MAXNUM	64	Maximum number of IP addresses on an IP address filtering list
NET_SYSTIME_SYNCTYPE_MAX_NUM	16	Maximum number of time synchronization types
NET_FISH_EYE_RECT_MAX_NUM	8	Maximum number of rectangles in fisheye display settings
NET_DISK_PART_MAX_NUM	10	Maximum number of partitions
NET_PTZ_3D_NRM_MAX_NUM	64	Maximum number of devices whose high-precision zooming capability can be obtained at a time
NET_CTRLLIB_MAX_NUM	32	Maximum number of archives
NET_CTRLLIB_CUSTOM_LABEL_MAX_NUM	3	Maximum number of archive tags
NET_RECOG_ENGINE_MAX_NUM	4	Maximum number of recognition engines
NET_PERSON_ITEM_MAX_NUM	30	Maximum number of archived personnel that can be obtained at a time
NET_GET_ERR_PERSON_ITEM_MAX_NUM	100	Maximum number of incorrect-information-contained personnel that can be obtained at a time
NET_PER_DEL_PERSON_MAX_NUM	30	Maximum number of personnel that can be deleted at a time
NET_ALGENGINE_MAX_NUM	16	Maximum number of algorithm engines that can be loaded

Macro	Value	Description
NET_AIUUPLOADCAPS_MAX_NUM	16	Maximum number of supported upload modes
NET_AIU_GETBIGDATACFG_MAX_NUM	16	Maximum number of channels whose VIID settings can be obtained at a time
NET_AIU_BIGDATA_UPLOAD_MAX_NUM	16	Maximum number of platforms to which pictures/videos can be uploaded
NET_AIU_IMPORT_CTRLLIB_FILE_MAX_SIZE	(1 * 1024 * 1024)	Maximum size of a file to be imported to an archive
NET_AIU_IMPORT_CTRLLIB_FILE_CGI_MAX_SIZE	(2 * 1024 * 1024)	Maximum size of a file to be imported to an archive after being encoded or decoded
NET_AIP_DETECT_CHN_MAX_NUM	128	MMaximum number of channels whose intelligent processing capability can be obtained at a time
NET_AIP_AI_CHN_ALG_PARAM_MAX_NUM	4	Maximum number of intelligent processing types of an algorithm engine
NET_AIP_COMPARE_RULE_CHN_MAX_NUM	64	Maximum number of compare channels involved in intelligent processing
NET_AIP_COMPARE_RULE_MAX_NUM	30	Maximum number of compare rules
NET_AISEARCH_CHN_MAX_NUM	128	Maximum number of channels that can be obtained in an intelligent search
NET_FACE_SNAP_ITEM_MAX_NUM	30	Maximum number of personnel whose information can be obtained in an intelligent search
NET_DETECT_SNAP_LIST_MAX_NUM	16	Maximum number of captures that can be displayed on intelligent viewing

Macro	Value	Description
NET_CMP_ALARM_LIST_MAX_NUM	50	Maximum number of matching alarms that can be displayed on intelligent viewing
NET_GB28181_ENC_CHN_MAX_NUM	260	Maximum number of video channels on a SIP platform
NET_GB28181_ALARM_CHN_MAX_NUM	256	Maximum number of alarm channels on a SIP platform
NET_SUBSCRIBE_MSG_MAX_NUM	5	Maximum number of messages that can be subscribed to

Enumeration Definition

GUI Display Languages

```
enum EGuiLangType
{
GUILANGTYPE_CN  = 0x01, ///< 简体中文
GUILANGTYPE_CHT = 0x02, ///< 繁体中文
GUILANGTYPE_EN  = 0x04, ///< English
GUILANGTYPE_ES  = 0x08, ///< Español
GUILANGTYPE_TR  = 0x10, ///< Türkçe
};
```

NVR Status

```
enum ENvrState
{
NVRSTATE_DISCONNECTED = 0x00000001, ///< The link is broken.
NVRSTATE_CHNLIST      = 0x00000002, ///< The channel list is changed.
NVRSTATE_CHNALIAS     = 0x00000004, ///< The channel name is changed.
NVRSTATE_CHNSTATE     = 0x00000008, ///< The channel status is changed.
NVRSTATE_CHNGROUP     = 0x00000010, ///< The channel group is changed.
NVRSTATE_RESOLUTION   = 0x00000020, ///< The resolution is changed.
NVRSTATE_DISKHOTSWAP  = 0x00000040, ///< Disk hot-swap is performed.
NVRSTATE_REALLOG      = 0x00000080, ///< Real-time logs
NVRSTATE_REALSTATE    = 0x00000100, ///< Real-time status
NVRSTATE_SYSSTATE     = 0x00000200, ///< System status (inserting/removing
of a USB flash drive or a network cable)
NVRSTATE_NETSTATE     = 0x00000400, ///< The network status is changed.
NVRSTATE_AIUSTATE     = 0x00000800, ///< The intelligence status is
changed.
NVRSTATE_ALARMSTATE   = 0x00001000, ///< The alarm status is changed.
};
```

Message Types

```
enum EMsgType
{
MSGTYPE_FACE_DETECT_SNAP  = 1, ///< Face detection message; message
structure: TNetDetectSnapItem
MSGTYPE_CAR_DETECT_SNAP   = 2, ///< Vehicle detection message; message
structure: TNetDetectSnapItem
MSGTYPE_CMP_ALARM         = 3, ///< Matching alarm message; message
structure: TNetCmpAlarmItem
MSGTYPE_PIC_UPLOAD_STATUS = 4, ///< Picture upload status message; message
structure: TNetPicUploadStatus
};
```

PTZ Controls

```
enum EPTZCMD
{
PTZCMD_MOVEUP              = 1,  ///< Move up
PTZCMD_MOVEDOWN            = 2,  ///< Move down
PTZCMD_MOVELEFT            = 3,  ///< Move to the left
PTZCMD_MOVERIGHT           = 4,  ///< Move to the right
PTZCMD_MOVELEFTUP          = 5,  ///< Move to the top left
PTZCMD_MOVELEFTDOWN        = 6,  ///< Move to the bottom left
PTZCMD_MOVERIGHTUP         = 7,  ///< Move to the top right
PTZCMD_MOVERIGHTDOWN       = 8,  ///< Move to the bottom right
PTZCMD_MOVESTOP            = 9,  ///< Stop moving
PTZCMD_RESET               = 10, ///< Reset
PTZCMD_FOCUSFAR            = 11, ///< Far focus
PTZCMD_FOCUSNEAR           = 12, ///< Near focus
PTZCMD_FOCUSAUTO           = 13, ///< Automatic focus
PTZCMD_FOCUSSTOP           = 14, ///< Automatic focus
PTZCMD_IRISPLUS            = 15, ///< Increase the aperture
PTZCMD_IRISMINUS           = 16, ///< Decrease the aperture
PTZCMD_IRISAUTO            = 17, ///< Automatic aperture
PTZCMD_IRISSTOP            = 18, ///< Stop adjusting the aperture
PTZCMD_ZOOMTELE            = 19, ///< Zoom in
PTZCMD_ZOOMWIDE            = 20, ///< Zoom out
PTZCMD_ZOOMSTOP            = 21, ///< Stop zooming
PTZCMD_LIGHTOPEN           = 22, ///< Turn on the light
PTZCMD_LIGHTCLOSE          = 23, ///< Turn off the light
PTZCMD_WIPEROPEN           = 24, ///< Turn on the wiper
PTZCMD_WIPERCLOSE          = 25, ///< Turn off the wiper
PTZCMD_PRESET_CALL         = 26, ///< Call a preset
PTZCMD_PRESET_SAVE         = 27, ///< Save a preset
PTZCMD_PRESET_DEL          = 28, ///< Delete a preset
PTZCMD_PATHCRUISE_CALL     = 29, ///< Call a tour path
PTZCMD_PATHCRUISE_STOP     = 30, ///< Stop a tour
PTZCMD_ZOOM_PART           = 31, ///< Electronically zoom in a pane
PTZCMD_ZOOM_WHOLE          = 32, ///< Electronically zoom out a pane
PTZCMD_GOTO_POINT          = 33, ///< Double-click to move to the center
};
```

RTSP Transmission Types

```
enum ERtspTransType
{
RTSPTRANSTYPE_UDP_UNICAST   = 1, ///< udp_unicast
RTSPTRANSTYPE_UDP_MULTICAST = 2, ///< udp_multicast
RTSPTRANSTYPE_TCP           = 3, ///< tcp
RTSPTRANSTYPE_HTTP          = 4, ///< http
};
```

Recording Event Types

```
enum ERecEventType
{
RECEVENTTYPE_RECORD           = 0x00000000, ///< A log, not an event
RECEVENTTYPE_ALL              = 0x00000001, ///< all_event, all events
RECEVENTTYPE_PIN              = 0x00000002, ///< pin, parallel-port alarm,
alarm input
RECEVENTTYPE_MOTIVE           = 0x00000004, ///< motive, motion detection
alarm
//RECEVENTTYPE_EXTERN         = 0x00000008, ///< extern, recording
triggered by an external alarm
RECEVENTTYPE_MANUAL           = 0x00000010, ///< manual, manual started
recording
RECEVENTTYPE_TIMER            = 0x00000020, ///< timer, scheduled recording
//RECEVENTTYPE_WEEKLY         = 0x00000040, ///< weekly, regularly started
recording
RECEVENTTYPE_INTEDETECT       = 0x10000000, ///< Basic intelligent feature
RECEVENTTYPE_COVERIMG         = 0x00000100, ///< cover_image, Video Blocked
RECEVENTTYPE_TRIPLINE         = 0x00000200, ///< trip_line, Tripwire
Detection
RECEVENTTYPE_DEFOCUS          = 0x00000400, ///< defocus, Defocus
RECEVENTTYPE_SCENECHANGE      = 0x00000800, ///< scene_change, Scene Change
RECEVENTTYPE_REGIONINVASION   = 0x00001000, ///< region_invasion, Enter
Guard Area
RECEVENTTYPE_REGIONLEAVING    = 0x00002000, ///< region_leaving, Exit Guard
Area
RECEVENTTYPE_OBJTAKEN         = 0x00004000, ///< object_taken, Object
Removal
RECEVENTTYPE_OBJLEFT          = 0x00008000, ///< object_left, Object Left
RECEVENTTYPE_PEOPLEGATHER     = 0x00010000, ///< people_gather, Gathering
RECEVENTTYPE_AUDIOABNORMAL    = 0x00020000, ///< audio_abnormal, Audio
Surge
RECEVENTTYPE_DETECTIN         = 0x00040000, ///< region_enter, Entry Guard
Area
RECEVENTTYPE_FACEDETECTION    = 0x00080000, ///< face_detect, Face
Detection
RECEVENTTYPE_IMPPERSON        = 0x00100000, ///< imp_person, Key Personnel
RECEVENTTYPE_STRANGEPERSON    = 0x00200000, ///< strange_person, Stranger
RECEVENTTYPE_LOCKED           = 0x20000000, ///< locked, Record Locked
RECEVENTTYPE_UNLOCKED         = 0x40000000, ///< unlocked, Record Unlocked
};
```

Playback Types

```
enum EPlaybackType
{
PLAYBACKTYPE_EVENTINDEX   = 1,  ///< event_index: playback by event
PLAYBACKTYPE_LABELINDEX   = 2,  ///< label_index: playback by tag
PLAYBACKTYPE_LOCKED       = 3,  ///< locked: playback by locked record
PLAYBACKTYPE_IMAGEINDEX   = 4,  ///< image_index: playback by pictures
PLAYBACKTYPE_EXTERNALFILE = 5,  ///< external_file: playback by external
file
PLAYBACKTYPE_TIMESCAL     = 6,  ///< time_scale: playback by period
PLAYBACKTYPE_BYFILE       = 7,  ///< by_file: playback by file
PLAYBACKTYPE_BYTIME       = 8,  ///< by_time: playback by time
};
```

Playback Status

```
enum EPlaybackVcrState
{
PLAYBACKSTATE_PLAY                      = 1, ///< play: playing
PLAYBACKSTATE_OVER                      = 2, ///< over: play finished
PLAYBACKSTATE_DISCON                    = 3, ///< discon: link broken
PLAYBACKSTATE_NOTSTART                  = 4, ///< notstart: not started
PLAYBACKSTATE_PAUSE                     = 5, ///< pause: play paused
PLAYBACKSTATE_SINGLE                    = 6, ///< single: single frame
PLAYBACKSTATE_START_FAILED              = 7, ///< allfail: all playbacks
failed
PLAYBACKSTATE_START_PARTIAL_FAILED      = 8, ///< partfail: some playbacks
failed
PLAYBACKSTATE_CHN_NO_STREAM             = 9, ///< chn_no_stream: no stream
in the current channel
};
```

Playback Controls

```
enum EPlaybackVcrCmdType
{
PLAYBACKVCRCMDTYPE_NORMAL      = 1,  ///< normal: play by the normal speed
PLAYBACKVCRCMDTYPE_PAUSE       = 2,  ///< pause: pause the play
PLAYBACKVCRCMDTYPE_RESUME      = 3,  ///< resume: resume the play
PLAYBACKVCRCMDTYPE_FAST2X      = 4,  ///< 2xfast: 2x fast
PLAYBACKVCRCMDTYPE_FAST4X      = 5,  ///< 4xfast: 4x fast
PLAYBACKVCRCMDTYPE_FAST8X      = 6,  ///< 8xfast: 8x fast
PLAYBACKVCRCMDTYPE_FAST16X     = 7,  ///< 16xfast: 16x fast
PLAYBACKVCRCMDTYPE_FAST32X     = 8,  ///< 32xfast: 16x fast
PLAYBACKVCRCMDTYPE_FAST64X     = 9,  ///< 64xfast: 16x fast
PLAYBACKVCRCMDTYPE_SLOW2X      = 10, ///< 2xslow: 1/2x slow
PLAYBACKVCRCMDTYPE_SLOW4X      = 11, ///< 4xslow: 1/4x slow
PLAYBACKVCRCMDTYPE_SLOW8X      = 12, ///< 8xslow: 1/8x slow
PLAYBACKVCRCMDTYPE_SLOW16X     = 13, ///< 16xslow: 16x slow
PLAYBACKVCRCMDTYPE_FRAME       = 14, ///< frame: single-frame play
PLAYBACKVCRCMDTYPE_DRAG        = 15, ///< drag: dragging on the timeline
PLAYBACKVCRCMDTYPE_SKIPF       = 16, ///< skipf: forward
PLAYBACKVCRCMDTYPE_SKIPB       = 17, ///< skipb: backward
};
```

Playback Frame Modes (Effective on Speed Adjustment Controls)

```
enum EPlaybackFrameMode
{
PLAYBACKFRAMEMODE_ALL          = 1, ///< Send all types of frames: I/P/B
frames
PLAYBACKFRAMEMODE_INTRA        = 2, ///< Send only key frames: I frames
PLAYBACKFRAMEMODE_PREDICTED    = 3, ///< Send only key frames and
prediction frames: I/P frames
};
```

Capture Type

```
enum ESnapPicEventType
{
SNAPPICEVENTTYPE_ALL                     = 0x01000000,   ///< All types;
all_event
SNAPPICEVENTTYPE_MANUAL                  = 0x02000000,   ///< Manual
capture; manual
SNAPPICEVENTTYPE_TIMER                   = 0x04000000,   ///< Manual
capture; timer
SNAPPICEVENTTYPE_MD                      = 0x08000000,   ///< Motion
detection; md
SNAPPICEVENTTYPE_PIN                     = 0x10000000,   ///< Alarm input;
pin
SNAPPICEVENTTYPE_INTEDETECT              = 0x20000000,   ///< Basic
intelligent feature
SNAPPICEVENTTYPE_INTEDETECT_ALL          = 0x20000001,   ///< Basic
intelligent feature; all
SNAPPICEVENTTYPE_INTEDETECT_TRIPLINE     = 0x20000002,   ///< Basic
intelligent feature, Guard Line; trip_line
SNAPPICEVENTTYPE_INTEDETECT_REGIONINVASION = 0x20000004,   ///< Basic
intelligent feature, Enter Guard Area; region_invasion
SNAPPICEVENTTYPE_INTEDETECT_REGINENTER   = 0x20000008,   ///< Basic
intelligent feature, Entry Guard Area; region_entering
SNAPPICEVENTTYPE_INTEDETECT_REGIONLEAVE  = 0x20000010,   ///< Basic
intelligent feature, Exit Guard Area; region_leaving
SNAPPICEVENTTYPE_INTEDETECT_OBJLEFT      = 0x20000020,   ///< Basic
intelligent feature, Object Left; object_left
SNAPPICEVENTTYPE_INTEDETECT_OBJTAKEN     = 0x20000040,   ///< Basic
intelligent feature, Object Removal; object_taken
SNAPPICEVENTTYPE_INTEDETECT_PEOPLEGATHER = 0x20000080,   ///< Basic
intelligent feature, Gathering; people_gather
SNAPPICEVENTTYPE_INTEDETECT_DETECTFACE   = 0x20000100,   ///< Basic
intelligent feature, Face Detection; detect_face
SNAPPICEVENTTYPE_INTEDETECT_SHADE        = 0x20000200,   ///< Basic
intelligent feature, Video Blocked; shade
SNAPPICEVENTTYPE_INTEDETECT_DEFOCUS      = 0x20000400,   ///< Basic
intelligent feature, Defocus; defocus
SNAPPICEVENTTYPE_INTEDETECT_SCENECHANGE  = 0x20000800,   ///< Basic
intelligent feature, Scene Change; scene_change
SNAPPICEVENTTYPE_INTEDETECT_AUDIOABNORMAL= 0x20001000,   ///< Basic
intelligent feature, Audio Surge; audio_abnormal
SNAPPICEVENTTYPE_INTEDETECT_IMPPERSON    = 0x20002000,   ///< Basic
intelligent feature, Audio Surge; imp_person
SNAPPICEVENTTYPE_INTEDETECT_STRANGEPERSON= 0x20004000,   ///< Basic
intelligent feature, Stranger; strange_person
};
```

Protocol Type

```
enum EProtoType
{
PROTOTYPE_UNKNOWN    = 0x00,    ///< Unknown
PROTOTYPE_ONVIF      = 0x01,    ///< ONVIF
PROTOTYPE_GB28181    = 0x02,    ///< GB28181
PROTOTYPE_RTSP       = 0x04,    ///< RTSP
PROTOTYPE_VSIP       = 0x08,    ///< VSIP
PROTOTYPE_IPCSEARCH = 0x10,    ///< ipcsearch
PROTOTYPE_LCAM       = 0x20,    ///< Local camera
PROTOTYPE_ALL        = 0xFF,    ///< All
};
```

Device Type

```
enum EDevType
{
DEVTYPE_NULL     = 0,    ///< Invalid
DEVTYPE_IPC      = 1,    ///< IPC
DEVTYPE_DVS      = 2,    ///< DVS
DEVTYPE_DVR      = 3,    ///< DVR
DEVTYPE_NVR      = 4,    ///< NVR
DEVTYPE_FISHEYE = 5,    ///< SINGLE_SRC_FISHEYE
};
```

Activation Status

```
enum EActiveState
{
ACTIVESTATE_ACTIVE      = 1, ///< Activated
ACTIVESTATE_NOTACTIVE   = 2, ///< Deactivated
ACTIVESTATE_ABNORMAL    = 3, ///< Abnormal
ACTIVESTATE_UNKNOWN     = 4, ///< Unknown
ACTIVESTATE_UNSUPPORT   = 5, ///< Not supported
};
```

Device Adding Modes

```
enum EDevAddMode
{
DEVADDMODE_AUTO          = 0x00,    ///< Auto
DEVADDMODE_SINGLESRC     = 0x01,    ///< One by one:single_source
DEVADDMODE_MULITSRC      = 0x02,    ///< In batches: multi_source
DEVADDMODE_KEDAFISHEYE   = 0x04,    ///< Fisheye devices: keda_fisheye
};
```

Transmission Protocol Types

```
enum ETransProto
{
ETRANSPROTO_AUTO              = 0x01,   ///< Auto
ETRANSPROTO_TCP               = 0x02,   ///< tcp
ETRANSPROTO_UDP               = 0x04,   ///< udp
ETRANSPROTO_RTP_OVER_TCP      = 0x08,   ///< rtp_over_tcp
};
```

Time Zone

```
enum ESummerTimeType
{
SUMMERTIMETYPE_DEC_12 = 0,
SUMMERTIMETYPE_DEC_11 = 1,
SUMMERTIMETYPE_DEC_10 = 2,
SUMMERTIMETYPE_DEC_9 = 3,
SUMMERTIMETYPE_DEC_8 = 4,
SUMMERTIMETYPE_DEC_7 = 5,
SUMMERTIMETYPE_DEC_6 = 6,
SUMMERTIMETYPE_DEC_5 = 7,
SUMMERTIMETYPE_DEC_4_30 = 8,
SUMMERTIMETYPE_DEC_4 = 9,
SUMMERTIMETYPE_DEC_3_30 = 10,
SUMMERTIMETYPE_DEC_3 = 11,
SUMMERTIMETYPE_DEC_2 = 12,
SUMMERTIMETYPE_DEC_1 = 13,
SUMMERTIMETYPE_0 = 14,
SUMMERTIMETYPE_ADD_1 = 15,
SUMMERTIMETYPE_ADD_2 = 16,
SUMMERTIMETYPE_ADD_3 = 17,
SUMMERTIMETYPE_ADD_3_30= 18,
SUMMERTIMETYPE_ADD_4 = 19,
SUMMERTIMETYPE_ADD_4_30 = 20,
SUMMERTIMETYPE_ADD_5 = 21,
SUMMERTIMETYPE_ADD_5_30 = 22,
SUMMERTIMETYPE_ADD_5_45 = 23,
SUMMERTIMETYPE_ADD_6 = 24,
SUMMERTIMETYPE_ADD_6_30 = 25,
SUMMERTIMETYPE_ADD_7 = 26,
SUMMERTIMETYPE_ADD_8 = 27,
SUMMERTIMETYPE_ADD_9 = 28,
SUMMERTIMETYPE_ADD_9_30 = 29,
SUMMERTIMETYPE_ADD_10 = 30,
SUMMERTIMETYPE_ADD_11 = 31,
SUMMERTIMETYPE_ADD_12 = 32,
SUMMERTIMETYPE_ADD_13 = 33,
};
```

Daylight Saving Time

```
enum ESummerTimeOffset
{
SUMMERTIMEOFFSET_30MIN  = 0x01,   ///< 30 minutes
SUMMERTIMEOFFSET_60MIN  = 0x02,   ///< 60 minutes
SUMMERTIMEOFFSET_90MIN  = 0x04,   ///< 90 minutes
SUMMERTIMEOFFSET_120MIN = 0x08,   ///< 120 minutes
};
```

Time Synchronization Server

```
enum ESysTimeSyncType
{
SYSTIMESYNCTYPE_NTP             = 0x0001,   ///< NTP
SYSTIMESYNCTYPE_VSIP            = 0x0002,   ///< VSIP
SYSTIMESYNCTYPE_ONVIF           = 0x0004,   ///< ONVIF
SYSTIMESYNCTYPE_GB1             = 0x0008,   ///< GB28181-1
SYSTIMESYNCTYPE_GB2             = 0x0010,   ///< GB28181-2
SYSTIMESYNCTYPE_ANDROID         = 0x0020,   ///< ANDROID
SYSTIMESYNCTYPE_GPS             = 0x0040,   ///< GPS
SYSTIMESYNCTYPE_WORKSTATION     = 0x0080,   ///< DDS
SYSTIMESYNCTYPE_AUTO            = 0x0100,   ///< Self-adaptive
};
```

Log Category

```
enum ELogMainType
{
LOGMAINTYPE_ALL                                 = 0x10000000, ///<
All
LOGMAINTYPE_ALARM                               = 0x20000000, ///<
Alarm
LOGMAINTYPE_SYSEXEPTION                         = 0x40000000, ///<
System exception
LOGMAINTYPE_USEROPERATE                         = 0x80000000, ///<
User operation
LOGMAINTYPE_SYSINFO                             = 0x01000000, ///<
System information
};
```

Log Type

```
enum ELogSubType
{
LOGSUBTYPE_ALL_ALL                              = 0x10000001, ///<
All-all

LOGSUBTYPE_ALARM_ALL                            = 0x20000001, ///<
Alarm-all
LOGSUBTYPE_ALARM_ALARMIN                        = 0x20000002, ///<
Alarm-alarm input
LOGSUBTYPE_ALARM_MOVING                         = 0x20000004, ///<
Alarm-Motion Detection
LOGSUBTYPE_ALARM_VIDEOLOST                      = 0x20000008, ///<
Alarm-Video Loss
LOGSUBTYPE_ALARM_WARNINGLINE                    = 0x20000010, ///<
Alarm-Guard Line
```

```
LOGSUBTYPE_ALARM_DETECT                                 = 0x20000020, ///<
Alarm-Enter Guard Area
LOGSUBTYPE_ALARM_DETECTIN                               = 0x20000040, ///<
Alarm-Entry Guard Area
LOGSUBTYPE_ALARM_DETECTOUT                              = 0x20000080, ///<
Alarm-Exit Guard Area
LOGSUBTYPE_ALARM_PROPERTYLOST                           = 0x20000100, ///<
Alarm-Object Left
LOGSUBTYPE_ALARM_PROPERTYTAKE                           = 0x20000200, ///<
Alarm-Object Removal
LOGSUBTYPE_ALARM_PEOPLEGATHERING                        = 0x20000400, ///<
Alarm-Gathering
LOGSUBTYPE_ALARM_FACEDETECTION                          = 0x20000800, ///<
Alarm-Face Detection
LOGSUBTYPE_ALARM_SHADE                                  = 0x20001000, ///<
Alarm-Video Blocked
LOGSUBTYPE_ALARM_OUTOFFOCUS                             = 0x20002000, ///<
Alarm-Defocus
LOGSUBTYPE_ALARM_SCENECHANGE                            = 0x20004000, ///<
Alarm-Scene Change
LOGSUBTYPE_ALARM_AUDIOEXCEPTION                         = 0x20008000, ///<
Alarm-Audio Surge
LOGSUBTYPE_ALARM_GPSOVERSPEED                           = 0x20010000, ///<
Alarm-Speeding

LOGSUBTYPE_SYSEXEPTION_ALL                              = 0x40000001, ///<
System exception-all
LOGSUBTYPE_SYSEXEPTION_NETFAULT                         = 0x40000002, ///<
System exception-Internet Disconnected
LOGSUBTYPE_SYSEXEPTION_IPCONFLIT                        = 0x40000004, ///<
System exception-IP Address Conflict
LOGSUBTYPE_SYSEXEPTION_MACCONFLIT                       = 0x40000008, ///<
System exception-MAC Address Conflict
LOGSUBTYPE_SYSEXEPTION_MONITORDROPPED                   = 0x40000010, ///<
System exception-Camera Disconnected
LOGSUBTYPE_SYSEXEPTION_NORECDISK                        = 0x40000020, ///<
System exception-No HDD
LOGSUBTYPE_SYSEXEPTION_DISKFAULT                        = 0x40000040, ///<
System exception-HDD Faulty
LOGSUBTYPE_SYSEXEPTION_RECSPACEFULL                     = 0x40000080, ///<
System exception-No Recording Space
LOGSUBTYPE_SYSEXEPTION_SNAPSPACEFULL                    = 0x40000100, ///<
System exception-No Snapshot Space
LOGSUBTYPE_SYSEXEPTION_ILLEGAACCESS                     = 0x40000200, ///<
System exception-Illegal Access
LOGSUBTYPE_SYSEXEPTION_HOTBACKUP                        = 0x40000400, ///<
System exception-Hot Backup Abnormal
LOGSUBTYPE_SYSEXEPTION_MEDIASTREAMLOSTALL               = 0x40001000, ///<
System exception-MSS Loss
LOGSUBTYPE_SYSEXEPTION_EXCEPTIONREBOOTALL               = 0x40002000, ///<
System exception-Power-Cut Shutdown
LOGSUBTYPE_SYSEXEPTION_RECALL                           = 0x40004000, ///<
System exception-Recording Error

LOGSUBTYPE_USEROPRATE_ALL                               = 0x80000001, ///<
User operation-all
LOGSUBTYPE_USEROPRATE_BOOTDEV                           = 0x80000002, ///<
User operation-rebooting/switching on/switching off the device
```

```
LOGSUBTYPE_USEROPRATE_LOGIN                                 = 0x80000004, ///<
User operation-Login and logout
LOGSUBTYPE_USEROPRATE_BROWSE                                = 0x80000008, ///<
User operation-starting/stopping viewing
LOGSUBTYPE_USEROPRATE_PTZ                                   = 0x80000010, ///<
User operation-PTZ controls
LOGSUBTYPE_USEROPRATE_RECIMAGEOPERATE                       = 0x80000020, ///<
User operation-capturing during playback
LOGSUBTYPE_USEROPRATE_RECIMAGEREPLAY                        = 0x80000040, ///<
User operation-capturing during playback
LOGSUBTYPE_USEROPRATE_LABEL                                 = 0x80000080, ///<
User operation-tagging
LOGSUBTYPE_USEROPRATE_BACKUPORDL                            = 0x80000100, ///<
User operation-backing up records
LOGSUBTYPE_USEROPRATE_CHN                                   = 0x80000200, ///<
User operation-channel operation
LOGSUBTYPE_USEROPRATE_CFGOPERATE                            = 0x80000400, ///<
User operation-importing/exporting configurations
LOGSUBTYPE_USEROPRATE_PARAMCONFIG                           = 0x80000800, ///<
User operation-configuring settings
LOGSUBTYPE_USEROPRATE_STORAGE                               = 0x80001000, ///<
User operation-HDD operations
LOGSUBTYPE_USEROPRATE_UPDATE                                = 0x80002000, ///<
User operation-upgrading
LOGSUBTYPE_USEROPRATE_RECOVERY                              = 0x80004000, ///<
User operation-restoring factory defaults
LOGSUBTYPE_USEROPRATE_GUIOUTPUTCUT                          = 0x80008000, ///<
User operation-switching between GUI outputs
LOGSUBTYPE_USEROPRATE_VOICECALLORBROADCAST                  = 0x80010000, ///<
User operation-broadcast and voice call
LOGSUBTYPE_USEROPRATE_EXTSTORAGE                            = 0x80020000, ///<
User operation-connecting to external storage units
LOGSUBTYPE_USEROPRATE_AIANLS                                = 0x80040000, ///<
User operation-intelligent analysis
LOGSUBTYPE_USEROPRATE_CTRLLIB                               = 0x80080000, ///<
User operation-archive operation

LOGSUBTYPE_SYSINFO_ALL                                      = 0x01000001, ///<
System information-all
LOGSUBTYPE_SYSINFO_LINE_CONNECT                             = 0x01000002, ///<
System information-network device access
LOGSUBTYPE_SYSINFO_USB_INF                                  = 0x01000004, ///<
System information-inserting/removing a USB flash drive
LOGSUBTYPE_SYSINFO_SATA_DISK                                = 0x01000008, ///<
System information-inserting/removing an HDD
LOGSUBTYPE_SYSINFO_SDCARD_INFO                              = 0x01000010, ///<
System information-inserting/removing a storage card
LOGSUBTYPE_SYSINFO_SIMCARD_INFO                             = 0x01000020, ///<
System information-inserting/removing a storage card
LOGSUBTYPE_SYSINFO_SRV_CENTER_REGIST                        = 0x01000040, ///<
System information-service registration
LOGSUBTYPE_SYSINFO_IPC_ONLINE                               = 0x01000080, ///<
System information-camera going online
LOGSUBTYPE_SYSINFO_REC_INFO                                 = 0x01000100, ///<
System information-starting/stopping a recording
};
```

Log Source

```
enum ELogSrcType
{
LOGSRCTYPE_ALL                                          = 0x01, ///< All
LOGSRCTYPE_NVR                                          = 0x02, ///< NVR
LOGSRCTYPE_CHN                                          = 0x04, ///<
Channel
LOGSRCTYPE_USER                                         = 0x08, ///< User
};
```

Structure Definition

NVR Capability

```
struct TNetNvrCap
{
s8 szDevModel[NET_MAXLEN_64];   ///< NVR model
BOOL32 bActive;                 ///< Whether the NVR is activated
s32 nMaxChnNum;                 ///< Maximum number of channels supported
by the NVR
s32 nMaxChnGroupNum;            ///< Maximum number of channel groups on
the NVR
s32 nNicNum;                    ///< Number of NICs
s32 nSerialNum;                 ///< Number of serial ports
s32 nMaxAlarmInNum;             ///< Maximum number of alarm inputs
s32 nGB28181InPlatNum;          ///< Number of upward SIP platforms (0: not
supports the GB28181)
BOOL32 bSupVsip;                ///< Whether supports the VSIP upward
protocol
BOOL32 bSupGB28181Ctl;          ///< Whether supports SIP downward controls
BOOL32 bSupPubSec;              ///< Whether supports the VIID upward
protocol
BOOL32 bSupPubSecCtl;           ///< Whether supports the VIID downward
protocol
BOOL32 bSupOnvif;               ///< Whether supports the ONVIF upward
protocol
BOOL32 bSupCloudServer;         ///< Whether supports the Cloud service
BOOL32 bSupZeroChnEnc;          ///< Whether supports the virtual channel
BOOL32 bSupAI;                  ///< Whether supports AI
BOOL32 bSupSnmp;                ///< Whether supports SNMP
BOOL32 bSupPcap;                ///< Whether supports package capturing
BOOL32 bSupBroadcast;           ///< Whether supports broadcast
BOOL32 bSupIpFilter;            ///< Whether supports IP address filtering
BOOL32 bSupSXTServer;           ///< Whether supports the VLine (currently
used by the vClient)
BOOL32 bDisableListenStateCtrl; ///< Whether disables listening status
control
BOOL32 bWirelessVeh;            ///< Whether being a vehicle device
BOOL32 bSupAisCtrlLib;          ///< Whether supports archives
BOOL32 bSupSnap;                ///< Whether supports capturing
BOOL32 bSupAiSearch;            ///< Whether supports intelligent search
BOOL32 bSupWebsocket;           ///< Whether supports websocket
BOOL32 bSupDDNS;                ///< Whether supports DDNS
BOOL32 bSupDragCusCanvas;       ///< Whether supports custom screen layouts
BOOL32 bSupSysHealth;           ///< Whether supports the system health
function
```

```
};
```

Login Information

```
struct TNetLoginInfo
{
u32 dwNvrIp;                                    ///< NVR IP, htonl
u16 wPort;                                      ///< NVR port number, htonl
s8  szUserName[NET_NVR_USER_NAME_MAX_LEN + 1]; ///< Username
s8  szPwd[NET_NVR_USER_PWD_MAX_LEN + 1];        ///< Password
s8  szEmail[NET_MAXLEN_32 + 1];                 ///< Mail address
EGuiLangType eGuiLangType;                      ///< GUI display language
(GUI only)
BOOL32 bEnableWebsocket;                        ///< Whether to enable
websocket
}NETPACKED;
```

Incorrect Login

```
struct TNetLoginErrInfo
{
s32 nLoginRetryTimes; ///< Number of remaining login attempts
s32 nLoginLockedTime; ///< Remaining login locking time
}NETPACKED;
```

NVR Status

```
struct TNetNvrState
{
u32 dwNvrStateMask;                 ///< NVR status, ENvrState value
u32 dwRealLogTimeStamp;             ///< Timestamp of real-time logs
u32 dwRealLogNum;                   ///< Number of real-time log entries
u32 dwRealStateNum;                 ///< Number of real-time status entries
s32 nNvrNtyEventNum;                ///< Number of NVR notifications
BOOL32 bFaceSnapChanged;            ///< Face capture change, effective
when NVRSTATE_AIUSTATUS
s8 szFaceSnapChnId[NET_MAXLEN_128]; ///< Channel ID of a face capture
change, effective when NVRSTATE_AIUSTATUS
BOOL32 bCarSnapChanged;             ///< Vehicle capture change, effective
when NVRSTATE_AIUSTATUS
s8 szCarSnapChnId[NET_MAXLEN_128];  ///< Channel ID of a vehicle capture
change, effective when NVRSTATE_AIUSTATUS
BOOL32 bAiuCfgChanged;              ///< Intelligence configuration change,
effective when NVRSTATE_AIUSTATUS
s8 szAiuCfgChnId[NET_MAXLEN_128];   ///< Channel ID of an intelligence
configuration change, effective when NVRSTATE_AIUSTATUS
BOOL32 bNvrAlarmChanged;            ///< NVR alarm change, effective when
NVRSTATE_ALARMSTATE
BOOL32 bChnAlarmChanged;            ///< Channel alarm change, effective
when NVRSTATE_ALARMSTATE
s8 szChnAlarmChnId[NET_MAXLEN_128]; ///< Channel ID of a channel alarm
change, effective when NVRSTATE_ALARMSTATE
s8 szChnListChnId[NET_MAXLEN_512];  ///< Channel ID of a channel list
change, effective when NVRSTATE_CHNLIST
};
```

Subscribe to Message Items

```
struct TNetSubscribeMsgItem
{
EMsgType eMsgType; ///< Message types
};
```

List of Subscribed Message Items

```
struct TNetSubscribeMsgList
{
s32 nNum;                                                    ///<
Number of subscribed message items
TNetSubscribeMsgItem atSubscribeMsgItem[NET_SUBSCRIBE_MSG_MAX_NUM]; ///<
Message contents
};
```

Message Item

```
struct TNetMsgItem
{
u32 dwHandle;      ///< Handle
EMsgType eMsgType; ///< Message type
void *pData;       ///< Data
s32 nDataLen;      ///< Data length
u32 dwData;        ///< MSGTYPE_FACE_DETECT_SNAP, MSGTYPE_CAR_DETECT_SNAP:
Number of faces or vehicles captured on the day
};
```

Request for Obtaining an RTSP URL

```
struct TNetGetRtspRealStreamUrlParam
{
s32 nChnId;                                  ///< Channel ID
ERtspTransType eRtspTransType;               ///< Transmission type of
RTSP streams
s32 nVideoEncNum;                            ///< Number of required
video encoding channels
s32 anVideoEncId[NET_NVR_VIDEO_ENC_MAX_NUM];   ///< Array of video encoding
channel IDs
s32 nAudioEncNum;                            ///< Number of required
audio encoding channels
s32 anAudioSrcId[NET_NVR_AUDIO_ENC_MAX_NUM];   ///< Array of audio encoding
channel IDs
}
```

rtsp url

```
struct TNetRtspRealStreamUrl
{
s8 szRtspUrl[NET_MAXLEN_256 + 1]; ///< Generated RTSP stream URL
s32 nRtspPort;                    ///< Port number in the RTSP stream URL
};
```

PTZ Controls

```
struct TNetPtzCtrl
{
s32 nChnId;            ///< Channel ID
EPTZCMD ePtzCmd;       ///< PTZ control
s32 nIspSpeed;         ///< ISP control speed (0~100); related commands:
PTZCMD_ZOOMTELE, PTZCMD_ZOOMWIDE, PTZCMD_FOCUSFAR, PTZCMD_FOCUSNEAR
s32 nPanSpeed;         ///< Panning speed (0~100); related commands:
PTZCMD_MOVELEFT, PTZCMD_MOVERIGHT, PTZCMD_MOVELEFTUP, PTZCMD_MOVELEFTDOWN,
PTZCMD_MOVERIGHTUP, PTZCMD_MOVERIGHTDOWN
s32 nTilSpeed;         ///< Tilting speed (0~100); related commands:
PTZCMD_MOVEUP, PTZCMD_MOVEDOWN, PTZCMD_MOVELEFTUP, PTZCMD_MOVELEFTDOWN,
PTZCMD_MOVERIGHTUP, PTZCMD_MOVERIGHTDOWN
s32 nNumber;           ///< Preset (0~255), tour path (1~32); related
commands: PTZCMD_PRESET_CALL, PTZCMD_PRESET_SAVE, PTZCMD_PRESET_DEL,
PTZCMD_PATHCRUISE_CALL, PTZCMD_PATHCRUISE_STOP
s32 nX;                ///< Horizontal coordinates during electronically
zooming (value range: 0~255; related commands: PTZCMD_GOTO_POINT,
PTZCMD_ZOOM_PART, PTZCMD_ZOOM_WHOLE)
s32 nY;                ///< Vertical coordinates during electronically
zooming (value range: 0~255; related commands: PTZCMD_GOTO_POINT,
PTZCMD_ZOOM_PART, PTZCMD_ZOOM_WHOLE)
s32 nWidth;            ///< Width during electronically zooming (value
range: 0~255; related commands: PTZCMD_GOTO_POINT, PTZCMD_ZOOM_PART,
PTZCMD_ZOOM_WHOLE)
s32 nHeight;           ///< Height during electronically zooming (value
range: 0~255; related commands: PTZCMD_GOTO_POINT, PTZCMD_ZOOM_PART,
PTZCMD_ZOOM_WHOLE)
s32 nWinWidth;         ///< Width during custom electronically zooming;
related commands: PTZCMD_GOTO_POINT, PTZCMD_ZOOM_PART, PTZCMD_ZOOM_WHOLE
s32 nWinHeight;        ///< Height during custom electronically zooming;
related commands: PTZCMD_GOTO_POINT, PTZCMD_ZOOM_PART, PTZCMD_ZOOM_WHOLE
};
```

Obtaining the Month View of Channel Records

```
struct TNetGetChnMonthRec
{
s32 nChnNum;                                        ///< Number of channels
s32 anChnId[NET_PER_CHNGETMONTHREC_MAX_NUM];        ///< Array of channel
IDs
s32 nYear;                                          ///< Year
s32 nMonth;                                         ///< Month
};
```

Month View Information

```
struct TNetChnMonthRecItem
{
s32 nChnId;                                         ///< Channel ID
u32 dwMonthRec;                                     ///< Month view
information (values of 1-31)
};
```

Month View Information List

```
struct TNetChnMonthRecList
{
s32 nChnNum;                                                    ///<
Number of channels
TNetChnMonthRecItem atChnMonthRecItem[NET_PER_CHNGETMONTHREC_MAX_NUM]; ///<
Month view information
};
```

Creating a Record Query Task

```
struct TNetCreatQueryRecTask
{
s32 nChnId;                    ///< Channel ID
BOOL32 bQueryRecord;           ///< Whether to log the search operation
s8 szStartTime[NET_MAXLEN_32]; ///< Start time (format: 2016-03-
15T12:48:01.330)
s8 szEndTime[NET_MAXLEN_32];   ///< End time (format: 2016-03-
15T12:48:01.330)
u32 dwRecEventTypeMask;        ///< Event type mask, the ERecEventType
value
BOOL32 bMergedRec;             ///< Whether to combine records
};
```

Searching Records

```
struct TNetGetRecTaskResult
{
s32 nTaskId;                           ///< Task ID
s32 nStartIndex;                       ///< Start record index
s32 nNeedNum;                          ///< Maximum number of desired
records
};
```

Array of Channel Records

```
struct TNetChnRecItem
{
u64 ullRecSize;                         ///< Record size, unit: byte
u64 ullStartTime;                       ///< Start time, unit: ms
s8 szStartTime[NET_MAXLEN_32];          ///< Start time, format: 2016-03-
15T12:48:01.330
u32 dwRecDur;                           ///< Record duration, unit: ms
ERecEventType eRecEventType;            ///< Record event type
BOOL32 bLocked;                         ///< Whether the record is locked
s8 szThumbnailUrl[NET_MAXLEN_512 + 1];  ///< Thumbnail URL of the record
};
```

Record Search Results

```
struct TNetChnRecList
{
BOOL32 bFinished;                                          ///< Whether
finished
s32 nTotalRecNum;                                          ///< Number of
total records
s32 nCurrRecNum;                                           ///< Number of
current records
s32 nRealRecNum;                                           ///< Number of
current records
TNetChnRecItem atChnRecItem[NET_PER_CHNGETRECLIST_MAX_NUM];  ///< Array of
channel records
};
```

Video Information

```
struct TNetDstVideoItem
{
u16 wDstRtpPort;    ///< Target RTP port number
u16 wDstRtcpPort;   ///< Tart RTCP port number
u8 byPayload;        ///< payload
};
```

Audio Information

```
struct TNetDstAudioItem
{
u16 wDstRtpPort;    ///< Target RTP port number
u16 wDstRtcpPort;   ///< Tart RTCP port number
u8 byPayload;        ///< payload
};
```

Playback Channel Information

```
struct TNetPlaybackDstChnItem
{
s32 nChnId;                                              ///< Channel ID
s8 szStartTime[NET_MAXLEN_32];                           ///< Start time
s8 szEndTime[NET_MAXLEN_32];                             ///< End time
s8 szResStartTime[NET_MAXLEN_32];                        ///< Resource
start time
s8 szResEndTime[NET_MAXLEN_32];                          ///< Resource
end time
u32 dwDstIp;                                             ///<
Destination address of stream transmission, htonl
s32 nDstVideoItemNum;                                    ///< Number of
videos
TNetDstVideoItem atDstVideoItem[NET_PLAYBACK_ITEM_MAX_NUM]; ///< Video
information
s32 nDstAudioItemNum;                                    ///< Number of
audio files
TNetDstAudioItem atDstAudioItem[NET_PLAYBACK_ITEM_MAX_NUM]; ///< Audio
information
};
```

Channel List in Playback

```
struct TNetChnRecList
{
s32 nNum;                                                    ///<
Quantity
TNetPlaybackDstChnItem atPlaybackDstChnItem[NET_PLAYBACK_MAX_NUM]; ///<
Channel information in playback
};
```

Source Video Information

```
struct TNetSrcVideoItem
{
u16 wSrcRtpPort;    ///< Source RTP port number
u16 wSrcRtcpPort;   ///< Source RTCP port number
};
```

Source Audio Information

```
struct TNetSrcAudioItem
{
u16 wSrcRtpPort;    ///< Source RTP port number
u16 wSrcRtcpPort;   ///< Source RTCP port number
};
```

Source Channel Information in Playback

```
struct TNetPlaybackSrcChnItem
{
s32 nChnId;                                                  ///< Channel ID
s32 nSrcVideoItemNum;                                        ///< Number of
videos
TNetSrcVideoItem atSrcVideoItem[NET_PLAYBACK_ITEM_MAX_NUM]; ///< Video
information
s32 nSrcAudioItemNum;                                        ///< Number of
audio files
TNetSrcAudioItem atSrcAudioItem[NET_PLAYBACK_ITEM_MAX_NUM]; ///< Audio
information
};
```

Source Channel List in Playback

```
struct TNetChnRecList
{
s32 nNum;                                                    ///<
Quantity
TNetPlaybackSrcChnItem atPlaybackSrcChnItem[NET_PLAYBACK_MAX_NUM]; ///<
Source channel information in playback
};
```

Obtaining RTSP URLs of Playback Streams

```
struct TNetGetRtspPlaybackUrlParam
{
s32 nTaskID;                                    ///< Playback task ID
s32 nChnID;                                     ///< Channel ID
s8 szStartTime[NET_MAXLEN_32];                  ///< Start time (format: 2016-
03-15T12:48:01.330)
s8 szEndTime[NET_MAXLEN_32];                    ///< End time (format: 016-03-
15T12:48:01.330)
s8 szResStartTime[NET_MAXLEN_32];               ///< Record start time
(format: 2016-03-15T12:48:01.330) (optional; if not specified, the record
start time may be earlier)
s8 szResEndTime[NET_MAXLEN_32];                 ///< Record end time (format:
2016-03-15T12:48:01.330) (optional; if not specified, the record end time
may be prolonged)
s32 nVideoEncNum;                               ///< Number of required video
encoding channels
s32 anVideoEncId[NET_NVR_VIDEO_ENC_MAX_NUM]; ///< Array of video encoding
channel IDs
s32 nAudioEncNum;                               ///< Number of required audio
encoding channels
s32 anAudioSrcId[NET_NVR_AUDIO_ENC_MAX_NUM]; ///< Array of audio encoding
channel IDs
};
```

RTSP URLs of Playback Streams

```
struct TNetRtspPlaybackUrl
{
s8 szRtspUrl[NET_MAXLEN_256 + 1];        ///< Generated RTSP stream URL
s32 nRtspPort;                           ///< Port number in the RTSP
stream URL
};
```

Playback Status

```
struct TNetPlaybackState
{
u64 ullPlaybackTime;                      ///< Playback time, unit: ms
EPlaybackVcrState ePlaybackState;         ///< Playback status
s8 szCurrPlayTime[NET_MAXLEN_32];         ///< Current playback time, unit:
ms (format: 2016-03-15T12:48:01.330)
s32 nCurrRtpTime;                         ///< RTP time
};
```

Playback Control

```
struct TNetPlaybackVcrCtrl
{
s32 nTaskId;                                ///< Task ID
s32 nChnId;                                 ///< Task ID
BOOL32 bBackwards;                          ///< Whether being a reverted
playback
EPlaybackVcrCmdType ePlaybackVcrCmdType;    ///< Playback control type
EPlaybackFrameMode ePlaybackFrameMode;      ///< Playback frame mode
(effective on speed adjustment controls)
s8 szSeekTime[NET_MAXLEN_32];               ///< Target time during dragging
on the timeline, unit: ms (format: 2016-03-15T12:48:01.330)
s32 nSkipTime;                              ///< Forward/backward interval
};
```

Capture Search Criteria

```
struct TNetQuerySnapPicInfo
{
s32 nChnId;                                 ///< Channel ID
s8 szStartTime[NET_MAXLEN_32];              ///< Start time (format: 2016-03-
15T12:48:01.330)
s8 szEndTime[NET_MAXLEN_32];                ///< End time (format: 2016-03-
15T12:48:01.330)
ESnapPicEventType eSnapPicEventType;        ///< Capture type
s32 nStartIndex;                            ///< Start index
s32 nNeedNum;                               ///< Number of captures to be
queried
};
```

Capture Information

```
struct TNetSnapPicInfo
{
s8 szUri[NET_MAXLEN_512 + 1];               ///< Unique identification of a
capture
s8 szSnapPicTime[NET_MAXLEN_32];            ///< Capturing time
u32 dwSnapPicEventMask;                     ///< Event type;
ESnapPicEventType group value
s32 nPicSize;                               ///< Capture size (unit: byte)
s32 nPicWidth;                              ///< Capture width
s32 nPicHeight;                             ///< Capture height
};
```

Capture List

```
struct TNetSnapPicList
{
BOOL32 bFinished;                                                ///< Whether
the search is completed
s32 nTotalNum;                                                   ///< Total
number of captures
s32 nCurrNum;                                                    ///< Number
of current captures
TNetSnapPicInfo atSnapPicInfo[NET_PER_QUERYSNAPPIC_MAX_NUM];  ///< Capture
array
};
```

Capture Parameters

```
struct TNetGetSnapPic
{
s32 nChnId;                                  ///< Channel ID
s8 szUri[NET_MAXLEN_512 + 1];                ///< Unique identification of a
capture
};
```

Capture Data

```
struct TNetPicData
{
s8 szPicPath[NET_MAXLEN_128];   ///< Capture save path
s8 *pszPicData;                 ///< Capture data
s32 nPicDataLen;                ///< Capture data length
};
```

Protocol List

```
struct TNetProtoList
{
u8 byProtoNum;                                    ///< Number of protocols
TNetProtoItem atProtoItem[NET_PROTO_MAX_NUM];  ///< Protocol contents
};
```

Protocol Information

```
struct TNetProtoItem
{
EProtoType eProtoType;   ///< Protocol type
BOOL32 bSupportSearch;   ///< Whether supports device search
u32 dwTransProtoMask;    ///< Media transmission protocol, ETransProto
value
};
```

Device Search Task Information

```
struct TNetSearchDevTask
{
u32 dwTaskId;        ///< Search task ID
u32 dwTaskTimes;     ///< Search duration, unit: s
};
```

Device Search Task Parameters

```
struct TNetSearchDevParam
{
u32 dwProtoMask;                    ///< Search devices which access the NVR
through a specific protocol, EProtoType value
BOOL32 bEnableIpFilter;             ///< Whether to enable IP filtering
u32 dwStartIp;                      ///< Whether to enable IP filtering
u32 dwEndIp;                        ///< End IP address, htonl
};
```

Searching the Device List

```
struct TNetSearchedDevList
{
BOOL32 bFinished;                                           ///< Whether
the search is finished
u16 wDevNum;                                                ///< Number
of devices
TNetDevInfo atDevInfo[NET_PER_GET_SEARCHED_DEV_MAX_NUM];    ///< Device
information
};
```

Device Extension Information

```
struct TNetDevExtInfo
{
    BOOL32 bSupActive;                      ///< Whether supports activation
    BOOL32 bActive;                         ///< Whether being activated
    BOOL32 bSupSetAddr;                     ///< Whether supports IP address
editing
    u32 dwSubnetMask;                       ///< Subnet mask, network byte
order
    u32 dwDefGateway;                       ///< Default gateway, network
byte order
    s8 szExtProtoVer[NET_MAXLEN_64 + 1];    ///< Protocol version used to
obtain the device extension information
    s8 szMac[NET_MAXLEN_32 + 1];            ///< MAC address
    s8 szSecureCode[NET_MAXLEN_64 + 1];     ///< Safety code
    EActiveState eActiveState;              ///< Activation status
};
```

Device Information

```
struct TNetDevInfo
{
u32 dwIp;                                   ///< Device IP, htonl
EProtoType eProtoType;                      ///< Protocol type
u16 wProtoPort;                             ///< Protocol port number
u16 wVideoEncChnNum;                        ///< Number of video encoding
channels
u16 wAudioEncNum;                           ///< Number of audio encoding
channels
u16 wAudioDecNum;                           ///< Number of audio decoding
channels
s8 szDevModel[NET_MAXLEN_32 + 1];           ///< Device model
```

```
s8 szManufacturer[NET_MAXLEN_32 + 1];          ///< Vendor
s8 szUuid[NET_MAXLEN_128 + 1];                 ///< Device UUID
EDevType eDevType;                             ///< Device type
s8 szSoftVer[NET_MAXLEN_64];                   ///< Software version
TNetDevExtInfo tDevExtInfo;                    ///< Extended device
information
};
```

Obtaining the NVR Channel List

```
struct TNetGetNvrChnList
{
u32 dwChnMask;                      ///< Obtain the specific information
about the channel list, EChnMask value
s32 nChnIdStart;                    ///< Start channel ID
s32 nChnIdEnd;                      ///< End channel ID
s8 szChnId[NET_MAXLEN_256];         ///< Channel ID; it is mutually exclusive
to nChnIdStart and nChnIdEnd; use a comma to separate multiple channels,
for example: 1,3,9
};
```

Channel List

```
struct TNetNvrChnList
{
u16 wChnNum;                ///< [in]-array length, [out]-number of
actually obtained channels
TNetChnItem *patChnItem;    ///< Point to the TNetChnItem array
};
```

Channel and Device Information

```
struct TNetChnItem
{
TNetChnInfo tChnInfo;            ///< Channel information
TNetDevItem tDevItem;            ///< Device information
};
```

Channel Information

```
struct TNetChnInfo
{
s32 nId;                           ///< Channel ID
s32 nDevId;                        ///< Device ID
s8  szAlias[NET_MAXLEN_64 + 1];    ///< Channel name, 0-32 characters
s32 nVideoSourceId;                ///< Video source ID
BOOL32 bIdle;                      ///< Whether the channel is idle
BOOL32 bNewAdd;                    ///< Whether the channel is newly added
TNetChnState tChnState;            ///< Channel status
};
```

Channel Status

```
struct TNetChnState
{
BOOL32 bDevOnline;          ///< The device is online.
BOOL32 bIsRecording;        ///< Whether a recording is ongoing
BOOL32 bMoveDetect;         ///< Whether a Motion Detection alarm is
triggered
BOOL32 bVideoLost;          ///< Whether a Video Source Loss alarm is
triggered
BOOL32 bAlmIn;              ///< Whether an alarm input alarm is triggered
BOOL32 bSmart;              ///< Whether a basic intelligent alarm is
triggered
EDevErrNo eDevErrNo;        ///< Error code for being offline
};
```

Device Information

```
struct TNetDevItem
{
TNetDevInfo tDevInfo;    ///< Basic information
};
```

Channel Configurations

```
struct TNetChnCfg
{
s32 nChnId;                                ///< Channel ID
TNetOnvifChnCfg tOnvifChnCfg;              ///< ONVIF device configurations
TNetRtspChnCfg tRtspChnCfg;                ///< RTSP device configurations
TNetGb28181DevCfg tGb28181DevCfg;          ///< SIP device configurations
};
```

ONVIF Device Configurations

```
struct TNetOnvifChnCfg
{
u32 dwIp;                                  ///< Device IP, htonl
u16 wProtoPort;                            ///< Protocol port number
EDevAddMode eDevAddMode;                   ///< Device adding mode
u16 wSrcId;                                ///< Video source ID
u16 wSrcNum;                               ///< Number of video sources
(effective only when eDevAddMode is set to DEVADDMODE_KEDAFISHEYE)
ETransProto eTransProto;                   ///< Transmission protocol type
s8 szAuthName[NET_MAXLEN_64 + 1];          ///< Username
s8 szAuthPwd[NET_MAXLEN_64 + 1];           ///< Password
};
```

RTSP Device Configurations

```
struct TNetRtspChnCfg
{
u32 dwIp;                                ///< Device IP, htonl
ETransProto eTransProto;                 ///< Transmission protocol type
s8 szMainStreamUrl[NET_MAXLEN_128 + 1]; ///< Address of the main stream
s8 szSecStreamUrl[NET_MAXLEN_128 + 1];  ///< Address of the secondary
stream
s8 szAuthName[NET_MAXLEN_64 + 1];        ///< Username
s8 szAuthPwd[NET_MAXLEN_64 + 1];         ///< Password
BOOL32 bSendRtspKplvForTcp;              ///< Whether to send RTSP heartbeat
messages in TCP mode
};
```

SIP Device Configurations

```
struct TNetGb28181DevCfg
{
s8 szDevId[NET_MAXLEN_32];                                          ///<
Device ID
s8 szDevPwd[NET_MAXLEN_64];                                         ///<
Password
ETransProto eTransProto;                                            ///<
Transmission protocol type
s32 nChnNum;                                                        ///<
Number of channels
TNetGb28181ChnCfg atGb28181ChnCfg[NET_GB28181_CHN_MAX_NUM];         ///<
Channel information
s32 nAlarmInNum;                                                    ///<
Number of alarm inputs
TNetGb28181AlarmInCfg atGb28181AlarmInCfg[NET_GB28181_CHN_MAX_NUM]; ///<
Alarm input information
};
```

SIP Channel Information

```
struct TNetGb28181ChnCfg
{
s32 nEncChnNum;
 ///< Number of encoding channels
TNetGb28181EncChnCfg atGb28181EncChnCfg[NET_GB28181_PRECHN_ENC_MAX_NUM];
///< Encoding channel IDs
};
```

Alarm Input Information

```
struct TNetGb28181AlarmInCfg
{
s8 szAlarmInId[NET_MAXLEN_32];         ///< Encoding channel ID
};
```

Device List

```
struct TNetAddDevList
{
u16 wDevNum;                                    ///< Number of devices
TNetAddDev atAddDev[NET_PER_ADD_DEV_MAX_NUM]; ///< Device information
};
```

Device Information

```
struct TNetAddDev
{
s32 nChnId;                    ///< Channel IDs starting from 1;
NET_CHNID_AUTO indicates that channel IDs are not specified and can be
automatically assigned
s8  szDevInfo[NET_MAXLEN_512]; ///< Device information(in xml format)
};
```

Deleting the Device List

```
struct TNetDelDevList
{
u16 wDevNum;                            ///< Number of devices
s32 anChnId[NET_PER_DEL_DEV_MAX_NUM];   ///< Array of channel IDs
};
```

Obtaining the System Time

```
struct TNetSystemTimeInfo
{
BOOL32 bTimeZone;       ///< Whether to obtain the time zone
BOOL32 bManualSync;     ///< Whether to obtain manually configured settings
BOOL32 bSummer;         ///< Whether to obtain DST settings
BOOL32 bAutoSync;       ///< Whether to obtain time synchronization
settings
};
```

System Time

```
struct TNetSystemTimeParam
{
ESummerTimeType eSummerTimeZone; ///< Time zone
BOOL32 bSyncEnable;              ///< Whether the current settings take
effect
s8 szTime[NET_MAXLEN_32];        ///< Time string
BOOL32 bSummerEnable;            ///< Whether to enable the DST
u32 dwSummerOffset;              ///< DST table, ESummerTimeOffset
ESummerTimeOffset eSummerOffset; ///< Current DST time
s32 nBeginMonth;                 ///< DST start month
s32 nBeginWeek;                  ///< DST start week
s32 nBeginDay;                   ///< DST start date
s32 nBeginHour;                  ///< DST start hour
s32 nEndMonth;                   ///< DST end month
s32 nEndWeek;                    ///< DST end week
s32 nEndDay;                     ///< DST end date
s32 nEndHour;                    ///< DST end hour
```

```
BOOL32 bAutoSyncEnable;          ///< Whether to enable automatic time
correction
s8 szType[NET_MAXLEN_16];        ///< Time correction type: ntp or proto
s8 szServerIP[NET_MAXLEN_32 + 1];///< IP or domain name
s32 nServerPort;                 ///< Port number
s32 nInterval;                   ///< Internal time interval
};
```

Automatic Time Correction

```
struct TNetSysTimeAutoSyncParam
{
BOOL32 bEnable;                                  ///< Whether to enable
automatic time correction
ESysTimeSyncType eAutoSyncType;                  ///< Time correction type
u32 dwAutoSyncTypeMask;                          ///< Supported time
correction type, EAutoSyncType group value
ESysTimeSyncType eSyncCurIndex;                  ///< Currently effective
type
s32 nAdaptLockTime;                              ///< Locking time for time
correction
s32 nAdaptLockTimeMin;                           ///< Minimum locking time
for time correction
s32 nAdaptLockTimeMax;                           ///< Maximum locking time
for time correction
s32 nAutoAdaptNum;                               ///< Number of self-
adaptive synchronization sources
TNetSysTimePriority atNetSysTimePriority[NET_SYSTIME_SYNCTYPE_MAX_NUM];
///< Priority list
};
```

Priority for Automatic Time Correction

```
struct TNetSysTimePriority
{
ESysTimeSyncType eAutoSyncType;     ///< Time correction type
BOOL32 bEnable;                     ///< Whether to enable it
};
```

System Time

```
struct TNetSystemTimeParamEx
{
ESummerTimeType eSummerTimeZone;                ///< Time zone
s8 szTime[NET_MAXLEN_32];                       ///< Time string (format:
2016-03-15T12:48:01.330)
TNetSysTimeAutoSyncParam tNetAutoSyncParam;     ///< Automatic time
correction settings
TNetSysTimeAutoSyncParam tNetDefSyncParam;      ///< Default automatic time
correction settings
BOOL32 bNtpEnable;                              ///< Whether to use the NTP
for the time correction purposes
s8 szNtpIP[NET_MAXLEN_32 + 1];                  ///< IP or domain name
s32 nNtpPort;                                   ///< Port number
s32 nNtpInterval;                               ///< Internal time interval
```

```
BOOL32 bSummerEnable;                          ///< Whether to enable the
DST
u32 dwSummerOffset;                            ///< DST table,
ESummerTimeOffset
ESummerTimeOffset eSummerOffset;               ///< Current DST time
s32 nBeginMonth;                               ///< DST start month
s32 nBeginWeek;                                ///< DST start week
s32 nBeginDay;                                 ///< DST start date
s32 nBeginHour;                                ///< DST start hour
s32 nEndMonth;                                 ///< DST end month
s32 nEndWeek;                                  ///< DST end week
s32 nEndDay;                                   ///< DST end date
s32 nEndHour;                                  ///< DST end hour
};
```

Log Type

```
struct TNetLogSubTypeItem
{
ELogSubType eLogSubType;                                        ///< Log
type
u32 dwLogSrcTypeMask;                                           ///< Log
source, ELogSrcType group value
};
```

Log Category

```
struct TNetLogMainTypeItem
{
ELogMainType eLogMainType;                                      ///< Log
category
u16 wSubTypeNums;                                               ///< Log
types
TNetLogSubTypeItem atLogSubTypeItem[NET_LOG_SUBTYPR_MAX_NUM];   ///< Log
type array
};
```

Parameters for Creating Log Search Tasks

```
struct TNetCreateSearchLogTaskParam
{
s8 szStartTime[NET_MAXLEN_32];                                  ///<
Start time
s8 szEndTime[NET_MAXLEN_32];                                    ///< End
time
ELogMainType eLogMainType;                                      ///< Log
category
ELogSubType eLogSubType;                                        ///< Log
type
ELogSrcType eLogSrcType;                                        ///< Log
source
s32 nChnId;                                                     ///< Log
source-channel ID
s8 szUserName[NET_NVR_USER_NAME_MAX_LEN + 1];                   ///< Log
source-username
EGuiLangType eGuiLangType;                                      ///<
Display language
}NETPACKED;
```

Information about a Log Searching Task

```
struct TNetSearchLogTaskInfo
{
u32 dwTaskId;                                                   ///< Task
ID
u32 dwLogTotalNums;                                             ///<
Number of logs
};
```

Log Information

```
struct TNetSearchLogItem
{
u32 dwTimeStamp;                                                ///< Time
stamp
s8 szCreateTime[NET_MAXLEN_32];                                 ///< Time
created
s8 szLogSrc[NET_MAXLEN_128];                                    ///< Log
source
s8 szLogType[NET_MAXLEN_64];                                    ///< Log
type
s8 szDetail[NET_MAXLEN_512 + 1];                                ///< Log
details
};
```

Interface Definition

Initialization

NET_Init

Initialize the kdnetsdk.

s32 NET_Init ();

Parameters
null

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
Initialize the netsdk before using it. Call NET_Cleanup before exiting the netsdk.

See Also
null

NET_Cleanup

Release kdnetsdk resources.

s32 NET_Cleanup ();

Parameters
null

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

Login and Logout

NET_GetNvrCap

Obtaining Device Capability

s32 NET_GetNvrCap(u32 dwNvrIp, u16 wPort, TNetNvrCap *ptNvrCap);

Parameters

```
dwNvrIp
[in] NVR IP, htonl
wPort
[in] NVR port number, htonl
ptNvrCap
[out] Device capability
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_Active

Activation

s32 NET_Active(TNetLoginInfo tLoginInfo, u32 *pdwHandle);

Parameters

```
tLoginInfo
[in] Login information
pdwHandle
[out] Handle
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
Obtain the device capability (TNetNvrCap) before login. Based on bActive, call this interface if the device is not activated. If the device is already activated, call the NET_LoginEx interface.
Determine whether to enable websocket during login and check whether the device supports websocket based on bSupWebsocket.
The NVR can receive subscribed messages only when websocket is enabled.

See Also
NET_SubscribeMsg

NET_LoginEx

Login

s32 NET_LoginEx(TNetLoginInfo tLoginInfo, u32 *pdwHandle, TNetLoginErrInfo *ptLoginErrInfo);

Parameters

```
tLoginInfo
[in] Login information
pdwHandle
[out] Handle
ptLoginErrInfo
[out] Extended returned message
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
Obtain the device capability (TNetNvrCap) before login. Based on bActive, call this interface if the device is already activated. If the device is not activated, call the NET_Active interface.
Determine whether to enable websocket during login and check whether the device supports websocket based on bSupWebsocket.
The NVR can receive subscribed messages only when websocket is enabled.

See Also
NET_SubscribeMsg

NET_Logout

Logout/Exit

s32 NET_Logout(u32 dwHandle);

Parameters

```
dwHandle
[in] Handle
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_RegNvrStateNty

Registering with NVR Status Notifications

s32 NET_RegNvrStateNty(u32 dwHandle, pfNvrStateCallBack pfFun);

Parameters

```
dwHandle
[in] Handle
pfFun
[in] Callback function for NVR status notifications
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
After login, this interface can be called. After registration, the NVR can receive link breaking down, channel list change, intelligence status change, alarm status change, and other types of notifications.
typedef void (*pfNvrStateCallBack)(u32 dwHandle, TNetNvrState tNvrState);

See Also
null

NET_UnregNvrStateNty

Cancel the Registration of NVR Status Notifications

s32 NET_UnregNvrStateNty(u32 dwHandle);

Parameters

```
dwHandle
[in] Handle
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_SubscribeMsg

Message Subscription

s32 NET_SubscribeMsg(u32 dwHandle, TNetSubscribeMsgList tSubscribeMsgList);

Parameters

```
dwHandle
[in] Handle
tSubscribeMsgList
[in] Message Subscription List
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
After calling this interface, you must call the NET_RegMsgNty interface. Otherwise, the NVR cannot receive notifications.

See Also
NET_RegMsgNty

NET_RegMsgNty

Registering with Message Notifications

s32 NET_RegMsgNty(pfMsgCallBack pfFun);

Parameters

```
pfFun
[in] Callback function for NVR status notifications
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
After login, this interface can be called. After this, the NVR can receive face/vehicle detection and matching alarm notifications.
Before calling this interface, you must call the NET_SubscribeMsg interface. Otherwise, the NVR cannot receive notifications.
typedef void (*pfMsgCallBack)(TNetMsgItem tMsgItem);

See Also
NET_SubscribeMsg

Reboot/Shutdown

NET_Shutdown

Shutdown

s32 NET_Shutdown(u32 dwHandle);

Parameters

```
dwHandle
[in] Handle
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_Reboot

Reboot

s32 NET_Reboot(u32 dwHandle);

Parameters

```
dwHandle
[in] Handle
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

Viewing

NET_GetRtspRealStreamUrl

Obtaining RTSP URLs of Live Streams

s32 NET_GetRtspRealStreamUrl(u32 dwHandle, TNetGetRtspRealStreamUrlParam tGetRtspRealStreamUrlParam, TNetRtspRealStreamUrl *ptRtspRealStreamUrl);

Parameters

```
dwHandle
[in] Handle
tGetRtspRealStreamUrlParam
[in] Request parameters
ptRtspRealStreamUrl
[out] rtsp url
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_RealStreamForceKeyFrame

Force Key Frames for Live Streams

s32 NET_RealStreamForceKeyFrame(u32 dwHandle, s32 nChnId, s32 nVidEncId);

Parameters

```
dwHandle
[in] Handle
nChnId
[in] Channel ID
nVidEncId
[in] Video encoding channel ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_SendPtzCmd

Send PTZ Commands

s32 NET_SendPtzCmd(u32 dwHandle, TNetPtzCtrl tPtzCtrl);
Parameters

```
dwHandle
[in] Handle
nChnId
[in] Channel ID
nVidEncId
[in] Video encoding channel ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

DEC_StartPlayRtspTcp

Start Playing-rtsp tcp

s32 DEC_StartPlayRtspTcp(TDecPlayRtspTcpParam tPlayRtspTcpParam, u32 *pdwHandle);

Parameters

```
tPlayRtspTcpParam
[in] Play parameters
pdwHandle
[out] Player handle starting from 1
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

DEC_StartPlayRtspUdp

Start Playing-rtsp udp

s32 DEC_StartPlayRtspUdp(TDecPlayRtspUdpParam tPlayRtspUdpParam, u32 *pdwHandle);

Parameters

```
tPlayRtspUdpParam
[in] Play parameters
pdwHandle
[out] Player handle starting from 1
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

DEC_StartPlayRtp

Start Playing-rtp udp

s32 DEC_StartPlayRtp(TDecPlayRtpParam tPlayRtpParam, u32 *pdwHandle);

Parameters

```
tPlayRtpParam
[in] Play parameters
pdwHandle
[out] Player handle starting from 1
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

DEC_StopPlay

Stop Playing

s32 DEC_StopPlay(u32 dwHandle);

Parameters

```
dwHandle
[in] Player handle
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

DEC_PlayAudio

Stop Playing

s32 DEC_PlayAudio(u32 dwHandle, EPlayAudioType ePlayAudioType);

Parameters

```
ePlayAudioType
[in] Audio playing type
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

DEC_SetAudioVolume

Stop Playing

s32 DEC_SetAudioVolume(u32 dwHandle, u32 dwVolume);

Parameters

```
dwVolume
[in] Audio volume; range [0,0xFFFF]
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

DEC_GetAudioVolume

Stop Playing

s32 DEC_GetAudioVolume(u32 dwHandle, u32 *pdwVolume);

Parameters

```
pdwVolume
[out] Audio volume; range [0,0xFFFF]
```

Return Values
Succeeded: DEC_OK; failed: an error code

Remarks
null

Requirements
Header: decsdk.h
Library: kddecsdk.dll

See Also
null

Playback

NET_GetChnMonthRec

Obtaining the Month View of Channel Records

s32 NET_GetChnMonthRec(u32 dwHandle, TNetGetChnMonthRec tGetChnMonthRec, TNetChnMonthRecList *ptChnMonthRecList);

Parameters

```
dwHandle
[in] Handle
tChnMonthRec
[in] Obtaining parameters
ptChnMonthRecList
[out] Receiving results
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
Obtain dates on which records of channels are available.

See Also
null

NET_CreatQueryRecTask

Creating a Record Search Task

s32 NET_CreatQueryRecTask(u32 dwHandle, TNetCreatQueryRecTask tCreateQueryRecTask, s32 *pnTaskId);

Parameters

```
dwHandle
[in] Handle
tCreateQueryRecTask
[in] Obtaining parameters
pnTaskId
[out] Receiving results
```

Return Values
Succeeded: NET_OK; failed: an error code

After the task is completed, you must call NET_DestroyQueryRecTask to destroy the task.

See Also
null

NET_GetRecTaskResult

Obtaining Record Search Results

s32 NET_GetRecTaskResult(u32 dwHandle, TNetGetRecTaskResult tGetRecTaskResult, TNetChnRecList *ptChnRecList);

Parameters

```
dwHandle
[in] Handle
tGetRecTaskResult
[in] Obtaining parameters
ptChnRecList
[out] Receiving results
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
A maximum of X (NET_PER_CHNGETRECLIST_MAX_NUM) results can be obtained at a time.

See Also
null

NET_DestroyQueryRecTask

Destroying a Record Search Task

s32 NET_DestroyQueryRecTask(u32 dwHandle, s32 nTaskId);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_CreatePlaybackTaskEx

Creating a Playback Task

s32 NET_CreatePlaybackTaskEx(u32 dwHandle, EPlaybackType ePlaybackType, BOOL32 bCreateRtspUrl, TNetPlaybackDstChnList tPlaybackDstChnList, s32 *pnTaskId, TNetPlaybackSrcChnList *ptPlaybackSrcChnList);

Parameters

```
dwHandle
[in] Handle
ePlaybackType
[in] Playback type
bCreateRtspUrl
[in] Whether to generate an RTSP URL
tPlaybackDstChnList
[in] Playback channel information
pnTaskId
[out] Task ID
ptPlaybackSrcChnList
[out] Playback channel information
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
After the task is completed, you must call NET_DestroyPlaybackTaskEx to destroy the task.

See Also
null

NET_DestroyPlaybackTaskEx

Destroying a Playback Task

s32 NET_DestroyPlaybackTaskEx(u32 dwHandle, s32 nTaskId);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_GetRtspPlaybackUrl

Obtaining RTSP URLs of Playback Streams

s32 NET_GetRtspPlaybackUrl(u32 dwHandle, TNetGetRtspPlaybackUrlParam tNetGetRtspPlaybackUrlParam, TNetRtspPlaybackUrl *ptNetRtspPlaybackUrl);

Parameters

```
dwHandle
[in] Handle
tNetGetRtspPlaybackUrlParam
[in] Request parameters
ptNetRtspPlaybackUrl
[out] Returned parameters
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_StartPlaybackEx

Start Playback

s32 NET_StartPlaybackEx(u32 dwHandle, s32 nTaskId);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
NET_StopPlaybackEx

NET_StopPlaybackEx

Stop Playback

s32 NET_StopPlaybackEx(u32 dwHandle, s32 nTaskId, s32 nChnId);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
nChnId
[in] Channel ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
NET_StartPlaybackEx

NET_AddPlaybackChnEx

Adding Channels to a Playback Task

s32 NET_AddPlaybackChnEx(u32 dwHandle, s32 nTaskId, TNetPlaybackDstChnList tPlaybackDstChnList, TNetPlaybackSrcChnList *ptPlaybackSrcChnList);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
tPlaybackDstChnList
[in] Playback channel information
ptPlaybackSrcChnList
[out] Playback channel information
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
NET_DelPlaybackChnEx

NET_DelPlaybackChnEx

Deleting Channels from a Playback Task

s32 NET_DelPlaybackChnEx(u32 dwHandle, s32 nTaskId, const s32 *panChnId, s32 nNum);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
panChnId
[in] Channel array
nNum
[in] Number of arrays
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
NET_AddPlaybackChnEx

NET_QueryPlaybackProgress

Querying the Playback Progress

s32 NET_QueryPlaybackProgress(u32 dwHandle, s32 nTaskId, s32 nChnId, TNetPlaybackState *ptPlaybackState);

Parameters

```
dwHandle
[in] Handle
nTaskId
[in] Task ID
nChnId
[in] Channel ID
ptPlaybackState
[out] Playback status
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_PlaybackVcrCtrl

Playback Control

s32 NET_PlaybackVcrCtrl(u32 dwHandle, TNetPlaybackVcrCtrl tPlaybackVcrCtrl);

Parameters

```
dwHandle
[in] Handle
tPlaybackVcrCtrl
[in] Control information
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_QuerySnapPic

Querying Captures

s32 NET_QuerySnapPic(u32 dwHandle, TNetQuerySnapPicInfo tQuerySnapPicInfo, TNetSnapPicList *ptSnapPicList);

Parameters

```
dwHandle
[in] Handle
tQuerySnapPicInfo
[in] Information about the target channel
ptSnapPicList
[out] Returned capture list
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_GetSnapPic

Obtaining Captures

s32 NET_GetSnapPic(u32 dwHandle, TNetGetSnapPic tGetSnapPic, TNetPicData *ptGetPicData);

Parameters

```
dwHandle
[in] Handle
tGetSnapPic
[in] Capture parameters
ptGetPicData
[out] Capture data
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
ptGetPicData cannot be NULL; the memory of ptGetPicData->pszPicData can be requested by the caller and cannot be NULL; ptGetPicData->nPicDataLen is the requested length.

See Also
null

DEC_StartRecDownloadTcp

Starting a Record Download

s32 DEC_StartRecDownloadTcp(TDecRecDownload tRecDownload, u32 *pdwHandle);

Parameters

```
tRecDownload
[in] Download parameters
pdwHandle
[out] Download handle starting from 1
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

DEC_StopRecDownload

Stopping a Record Download

s32 DEC_StopRecDownload(u32 dwHandle);

Parameters

```
dwHandle
[in] Download handle
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

Channel Management

NET_GetProtoList

Obtaining the Protocol List

s32 NET_GetProtoList(u32 dwHandle, TNetProtoList *ptProtoList);

Parameters

```
dwHandle
[in] Handle
ptProtoList
[out] Protocol list
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_CreateSearchDevTask

Creating a Device Search Task

s32 NET_CreateSearchDevTask(u32 dwHandle, u32 dwProtoMask, TNetSearchDevTask *ptSearchDevTask);

Parameters

```
dwHandle
[in] Handle
dwProtoMask
[in] Searching devices that access the NVR through a specific protocol;
EProtoType value
ptSearchDevTask
[out] Search task information
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_CreateSearchDevTaskEx

Creating a Device Search Task with Search Criteria

s32 NET_CreateSearchDevTaskEx(u32 dwHandle, TNetSearchDevParam tSearchDevParam, TNetSearchDevTask *ptSearchDevTask);

Parameters

```
dwHandle
[in] Handle

tSearchDevParam
[in] Search criteria

ptSearchDevTask
[out] Search task information
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_DestroySearchDevTask

Destroying a Device Search Task

s32 NET_DestroySearchDevTask(u32 dwHandle, u32 dwTaskId);

Parameters

```
dwHandle
[in] Handle

dwTaskId
[in] Search task ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_GetSearchedDevList

Obtaining Searched Devices

s32 NET_GetSearchedDevList(u32 dwHandle, u32 dwTaskId, TNetSearchedDevList *ptSearchedDevList);

Parameters

```
dwHandle
[in] Handle

dwTaskId
[in] Search task ID

ptSearchedDevList
[out] List of searched devices
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_GetNvrChnList

Succeeded: NET_OK; failed: an error code

s32 NET_GetNvrChnList(u32 dwHandle, TNetGetNvrChnList tGetNvrChnList, TNetNvrChnList *ptNvrChnList);

Parameters

```
dwHandle
[in] Handle

tGetNvrChnList
[in] Parameters for obtaining the NVR channel list

ptSearchedDevList
[out] List of searched devices
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
A maximum of X (NET_PER_GET_CHNLIST_MAX_NUM) results can be obtained at a time.

See Also
null

NET_GetChnCfg

Obtaining Channel Configurations

s32 NET_GetChnCfg(u32 dwHandle, TNetChnCfg *ptNvrChnCfg);

Parameters

```
dwHandle
[in] Handle

ptNvrChnCfg
[out] Receiving results
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_SetChnCfg

Configuring Channels

s32 NET_SetChnCfg(u32 dwHandle, s32 nChnId, const s8 *pszChnCfgInfo);

Parameters

```
dwHandle
[in] Handle

nChnId
[in] Channel ID

pszChnCfgInfo
[in] Channel configurations
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_AddDev

Configuring Channels

s32 NET_AddDev(u32 dwHandle, TNetAddDevList tAddDevList, TNetChnCfg *ptChnCfg);

Parameters

```
dwHandle
[in] Handle

tAddDevList
[in] Device list

ptChnCfg
[in] Device information; effective for SIP devices; if the value is NULL,
no attention is required
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_AddGB28181Dev

Adding a SIP Device

s32 NET_AddGB28181Dev(u32 dwHandle, const s8 *pszChnCfgInfo, TNetChnCfg *ptChnCfg);

Parameters

```
dwHandle
[in] Handle

pszChnCfgInfo
[in] SIP device information

ptChnCfg
[in] Device information; effective for SIP devices; if the value is NULL,
no attention is required
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_DelDev

Succeeded: NET_OK; failed: an error code

s32 NET_DelDev(u32 dwHandle, s32 nChnId);

Parameters

```
dwHandle
[in] Handle

dwChnId
[in] Channel ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_DelDevEx

Deleting a Device

s32 NET_DelDevEx(u32 dwHandle, TNetDelDevList tDelDevList);

Parameters

```
dwHandle
[in] Handle

tDelDevList
[in] Deleting the device list
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

System Settings

NET_GetSystemTimeParam

Obtain Parameters of the System Time

s32 NET_GetSystemTimeParam(u32 dwHandle, TNetSystemTimeInfo tSystemTimeInfo, TNetSystemTimeParam* ptSystemTimeParam);

Parameters

```
dwHandle
[in] Handle
tSystemTimeInfo
[in] Obtain parameters of the system time
ptSystemTimeParam
[out] Obtain parameters of the system time
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
This interface applies to the NVR R2B2 and earlier versions.

NET_GetSystemTimeParamEx

Obtain Parameters of the System Time

s32 NET_GetSystemTimeParamEx(u32 dwHandle, TNetSystemTimeParamEx* ptSystemTimeParam);

Parameters

```
dwHandle
[in] Handle
ptSystemTimeParam
[out] System time
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
This interface applies to the NVR R2B3 and later versions.

System Maintenance

NET_GetLogCap

Obtain Logging Capability

s32 NET_GetLogCap(u32 dwHandle, u32 *pdwLangCap, TNetLogMainTypeItem *patLogMainTypeItem, s32 *pLen);

Parameters

```
dwHandle
[in] Handle
pdwLangCap
[out] Language capability of logging. When the pointer is empty, the system will not obtain the capability.
ptLogMainTypeItem
[out] Supported log types
pLen
[in/out] You are advised to enter the greatest value to ensure that all search results can be displayed. Search results are subject to actual situations.
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_CreateSearchLogTask

Create a Log Search Task

s32 NET_CreateSearchLogTask(u32 dwHandle, TNetCreateSearchLogTaskParam tCreateSearchLogTaskParam, TNetSearchLogTaskInfo* ptSearchLogTaskInfo);

Parameters

```
dwHandle
[in] Handle
tCreateSearchLogTaskParam
[in] Parameters of the task
ptSearchLogTaskInfo
[out] Task information
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_GetSearchLogResult

Obtain Log Search Results

s32 NET_GetSearchLogResult(u32 dwHandle, u32 dwTaskId, u32 dwStart, TNetSearchLogItem *patSearchLogItem, u32 *pNums);

Parameters

```
dwHandle
[in] Handle
dwTaskId
[in] Task ID
ptSearchLogItem
[out] Log information
pNums
[in/out] You are advised to enter the greatest value to ensure that all search results can be displayed. Search results are subject to actual situations.
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

NET_DestroySearchLogTask

Destroy a Log Search Task

s32 NET_DestroySearchLogTask(u32 dwHandle, u32 dwTaskId);

Parameters

```
dwHandle
[in] Handle
dwTaskId
[in] Task ID
```

Return Values
Succeeded: NET_OK; failed: an error code

Remarks
null

See Also
null

Error Code

Error Code	Value	Description
NET_OK	0	Operation succeeded
Definition of cgiapp error code	Error code range: 1~500	
ERR_NET_UPGRADE_PREPARE_OK	1	Preparation for Device Upgrade Completed
ERR_NET_UPGRADE_SUCCESS	2	Device Upgraded
ERR_NET_SYSTEM_REBOOT	3	Device Rebooted
ERR_NET_FACTORY_DEF	4	Reset to Factory Defaults
ERR_NET_URL_CHANGE	5	CGI Service Address Changed
ERR_NET_PTZ_UPGRADE	6	PT Upgraded
ERR_NET_ISUPGRADE	7	Device Being Upgraded
ERR_NET_NO_SEC_MAIL	8	Safety Mail Not Configured
ERR_NET_ALREADY_ACT	9	Device Activated
ERR_NET_ALREADY_LOGIN	10	User Already Logged In
ERR_NET_SYSTEM_SHUTDOWN	11	Device Shut Down
ERR_NET_UNKNOW	201	Protocol Error
ERR_NET_USERNAME_ERR	202	User Not Exist
ERR_NET_NOT_AUTH	203	Authentication Failed
ERR_NET_PASS_ERR	204	Incorrect Password
ERR_NET_AUTHID_ERR	205	Incorrect Authentication ID
ERR_NET_NO_POWER	206	No Authorization
ERR_NET_IP_DENY	207	IP Locked
ERR_NET_OLDPASS_ERR	208	Incorrect Old Password
ERR_NET_USER_EXISTED	209	User Already Exists
ERR_NET_URL_NO_SUPPORT	210	Function Not Supported or Illegal Signaling Message
ERR_NET_NEED_BODY	211	Illegal Signaling Message
ERR_NET_XML_ERR	212	Illegal Signaling Message
ERR_NET_XML_ROOT_ERR	213	Illegal Signaling Message
ERR_NET_PARAM_LOST	214	Illegal Signaling Message

Error Code	Value	Description
ERR_NET_PARMM_TOOLONG	215	Illegal Signaling Message
ERR_NET_CHECK_ERR	216	Verification of Upgrade Information Failed
ERR_NET_FILE_ERR	217	Verification of Upgrade Package Failed
ERR_NET_UPGRADE_FAILURE	218	Upgrade Failed
ERR_NET_PARAM_ERR	219	Illegal Signaling Message
ERR_NET_IP_ERR	220	Illegal IP
ERR_NET_MASK_ERR	221	Incorrect Subnet Mask
ERR_NET_GATEWAY_ERR	222	Incorrect Default Gateway
ERR_NET_DNS_ERR	223	Incorrect DNS Address
ERR_NET_MULTICAST_ERR	224	Incorrect Multicast Address
ERR_NET_MTU_ERR	225	MTU Error
ERR_NET_SEQ_ERR	226	Signaling Message Sequence Error
ERR_NET_URL_ERR	227	Incorrect URL Format
ERR_NET_INVALID_ARG	228	Illegal Parameter
ERR_NET_DEVICE_BUSY	229	Device Busy
ERR_NET_RECOVER	230	Operation Failed and Parameter Values Recovered
ERR_NET_USER_BANNED	231	User Locked
ERR_NET_USER_ACTIVE	232	Device Deactivated
ERR_NET_NO_MEM	233	Insufficient Memory
ERR_NET_BUF_TOO_SMALL	234	Insufficient Data Buffer
ERR_NET_NOT_FOUND	235	Illegal Signaling Message
ERR_NET_NO_IMPLEMENTED	236	Function Not Supported
ERR_NET_ALREADY_EXIST	237	Exists
ERR_NET_NEED_LOGIN	238	Not Logged In
ERR_NET_USER_DISABLED	239	User Disabled
ERR_NET_USER_PORT_OCCUPIED	240	Port Occupied
ERR_NET_USER_USERS_TOPLIMIT	241	Upper Limit for Number of User Logins Reached

Error Code	Value	Description
Definition of service error code	Error code range: 1000~3000	
ERR_NET_ERROR	1001	Operation failed. Please try again later.
ERR_NET_ASSERT	1002	Illegal Parameter
ERR_NET_SEM_TAKE_FAILED	1003	Operation failed. Please try again later.
ERR_NET_SEM_GIVEE_FAILED	1004	Operation failed. Please try again later.
ERR_NET_PARAM_INVALID	1005	Illegal Parameter
ERR_NET_WAIT_TIMEOUT	1006	Operation Timed Out
ERR_NET_MALLOC_FAILED	1007	Failed to Allocate Memory
ERR_NET_STRING_ILLEGAL	1008	Illegal String
ERR_NET_STRING_TOO_SHORT	1009	String Too Short
ERR_NET_STRING_TOO_LONG	1010	String Too Long
ERR_NET_FILE_INEXIST	1020	File Not Exist
ERR_NET_WRITE_FILE	1021	Failed to Write Data into File
ERR_NET_EXPORTING_CFG	1022	Importing or Exporting Configuration File
ERR_NET_REGISTER_FULL	1023	Upper Registration Limit Reached
ERR_NET_ALREADY_REGISTERED	1024	Registered
ERR_NET_NO_REGISTERED	1025	Not Registered
ERR_NET_IO_ERROR	1026	IO error. The HDD does not exist or becomes full.
ERR_NET_CAP_NO_SUPPORT	1100	Capability Limited
ERR_NET_USER_EXIST	1200	User Already Exists
ERR_NET_USER_INEXIST	1201	User Not Exist
ERR_NET_USER_FILE_BROKEN	1202	File Damaged
ERR_NET_USER_NUM_MAX	1203	Upper Limit for Number of Users Reached
ERR_NET_USER_DEL_DISALLOWED	1204	Deleting User Prohibited
ERR_NET_USER_NAME_ILLEGAL	1205	Illegal Parameters Found in Username

Error Code	Value	Description
ERR_NET_USER_PWD_ILLEGAL	1206	Illegal Parameters Found in Password
ERR_NET_USER_NAME_LEN_TOO_LONG	1207	Username Too Long
ERR_NET_USER_NAME_LEN_TOO_SHORT	1208	Username Too Short
ERR_NET_USER_PWD_LEN_TOO_LONG	1209	Password Too Long
ERR_NET_USER_PWD_LEN_TOO_SHORT	1210	Password Too Short
ERR_NET_USER_PWD_STRENGTH_WEAK	1211	Password Too Weak
ERR_NET_USER_MGR_EMAIL_ILLEGAL	1212	Incorrect Mail Format
ERR_NET_USER_REMOTE_IP_INFO_ILLEGAL	1213	Illegal Remote IP
ERR_NET_USER_NAME_MDY_DISALLOWED	1214	Editing Username Prohibited
ERR_NET_USER_ADMIN_PERM_MDY_DISALLOWED	1215	No Authorization
ERR_NET_USER_DEV_SYS_NOACTIVE	1216	Device Deactivated
ERR_NET_USER_PASS_SAME_TO_BEFORE	1217	Same as Old Password
ERR_NET_USER_NAME_NULL	1218	Username Not Specified
ERR_NET_USER_PASS_NULL	1219	Password Not Specified
ERR_NET_LOG	1300	Log Related
ERR_NET_USER_LOG_TASK_BUSY	1301	The log task is busy. Please try again later.
ERR_NET_USER_LOG_TASK_ID_UNAVAILABLE	1302	Invalid Task ID
ERR_NET_DEV	1400	Peripherals
ERR_NET_NET	1500	Network
ERR_NET_NET_REGISTER_FULL	1501	Upper Registration Limit Reached
ERR_NET_NET_ALREADY_REGISTERED	1502	Registered
ERR_NET_NET_NO_REGISTERED	1503	Not Registered
ERR_NET_NET_PING_NUM_MAX	1504	Upper Ping Limit Reached
ERR_NET_NET_DOMAIN_ANALY_FAILD	1505	Failed to Parse Domain
ERR_NET_NET_PORT_IS_USING	1506	Network Port Occupied
ERR_NET_NET_OTHER_IP_IN_SAME_NET	1507	NIC IP addresses must be located on different segments.
ERR_NET_NET_IP_GW_NOTIN_SAME_NET	1508	The subnet mask and default gateway must be located on the same segment.

Error Code	Value	Description
ERR_NET_NET_OPERATE_TOO_FREQUENCY	1509	Too Frequent Requests. Please try again later.
ERR_NET_NET_PING_CHN_NO_IP	1510	The channel has no IP address.
ERR_NET_MPU	1600	mpu-related Error
ERR_NET_MPU_DEC_CHN_NUM_OVER_MAX	1601	Upper Limit for Number of Decoding Channels Reached
ERR_NET_MPU_DEC_ABILITY_OVER_MAX	1602	Decoding Capability Exceeded
ERR_NET_MPU_CHN_ID_IS_DECODING	1603	Channel Being Decoded
ERR_NET_MPU_MC_SET_LAYOUT_FAILED	1604	Failed to Configure Screen Layout
ERR_NET_MPU_MC_SET_DEC_PARAM_FAILED	1605	Failed to Create Decoder
ERR_NET_MPU_MC_SET_OPT_FAILED	1606	Failed to Configure Parameters
ERR_NET_MPU_MC_GET_OPT_FAILED	1607	Failed to Obtain Parameter Values
ERR_NET_MPU_OVER_MC_DEV_ZOOM_CAP	1608	Scaling Capability Exceeded
ERR_NET_MPU_BIND_FAILED	1609	Binding Failed
ERR_NET_MPU_UNBIND_FAILED	1610	Unbinding Failed
ERR_NET_MEDIA	1700	media-related Error
ERR_NET_DM	1800	HDD-related Error
ERR_NET_DM_DISK_ID_INVALID	1801	Invalid HDD ID
ERR_NET_DM_DISK_IN_USE	1802	HDD in Use
ERR_NET_DM_NET_DISK_NAME_TOO_LONG	1803	HDD Name Too Long
ERR_NET_DM_FUNCTION_NOT_SUPPORT	1804	Function Not Supported
ERR_NET_DM_DISK_USED_BY_RP	1805	HDD in Use
ERR_NET_DM_DISK_USED_BY_RP_PLY	1806	Playing Back
ERR_NET_DM_DISK_USED_BY_RP_DLD	1807	Downloading
ERR_NET_DM_DISK_UMOUNT_PART_ERR	1808	Uninstalling Failed
ERR_NET_DM_DISK_FORMAT_PART_ERR	1809	Formatting Failed
ERR_NET_DM_DISK_CHG_CALLBACK_FULL	1810	Full HDD
ERR_NET_DM_DISK_EXPCPTION_FORBID_OPERATION	1811	Operation Forbidden Due to HDD Exception

Error Code	Value	Description
ERR_NET_DM_DISK_EXTERNAL_DISK_LIMIT	1812	Upper Limit for Number of External Storage Units Reached
ERR_NET_DM_DISK_BAD_SECTOR_CHECK_NO_TASKID	1813	Upper Limit for Number of Bad Sector Detection Tasks Reached
ERR_NET_DM_DISK_DISK_FS_TYPE_FAILD	1814	Incorrect HDD Format
ERR_NET_DM_DISK_RAID_HOTBACKUP_DISK_SIZE_ERROR	1815	Small Capacity of Hot Backup HDD
ERR_NET_DM_DISK_SET_QUTOA_SIZE_OVER_ALL_DISK_SIZE	1816	Quota Size Greater than Total HDD Capacity
ERR_NET_DM_DISK_RAID_DELING	1817	Deleting RAID
ERR_NET_DM_DISK_RAID_CREATING	1818	Creating RAID
ERR_NET_DM_DISK_JUST_SUP_ONE_SMART_DISK	1820	Only one smart HDD is supported.
ERR_NET_REC	1900	Recording-related Error
ERR_NET_REC_CFG_DATA_NOT_EXIST	1901	Recording Configurations Not Exist
ERR_NET_REC_NO_IDLE_PLY_TASK	1902	Upper Limit for Number of Recordings Reached
ERR_NET_REC_MSIN_STOP_FAILED	1903	Failed to Stop Receiving Recording Data
ERR_NET_REC_MSIN_RELEASE_FAILED	1904	Failed to Stop Receiving Recording Data
ERR_NET_REC_REPEAT_TO_ADD_CHN	1905	Channel Already Added
ERR_NET_REC_START_PLY_FAILED	1906	Failed to Start the Playback
ERR_NET_REC_BAKUP_TASK_FULL	1907	Upper Limit for Number Recording Backup Tasks Reached
ERR_NET_REC_IMG_BAK_TASK_FULL	1908	Upper Limit for Number Snapshot Backup Tasks Reached
ERR_NET_REC_CHN_NOT_START	1909	Recording for Channel Not Enabled
ERR_NET_REC_CMD_DEAL_THREAD_BUSY	1910	Processing Thread Busy
ERR_NET_REC_PART_BUSY	1911	Partition Busy
ERR_NET_REC_COMPONENT_LIB_ERR	1912	Recording Error
ERR_NET_REC_DISK_STATUS_SLEEP	1913	Waking HDD Up

Error Code	Value	Description
ERR_NET_REC_PLAYER_FULL	1914	No more playback tasks can be created. Some playbacks already failed.
ERR_NET_CFG	2000	Configuration-related Error
ERR_NET_CFG_OPEN_DATABASE_FALID	2001	Failed to Open Database
ERR_NET_CFG_CLOSE_DATABASE_FALID	2002	Failed to Close Database
ERR_NET_CFG_CREATE_TABLE_FALID	2003	Failed to Create Table
ERR_NET_CFG_GET_PARAM_FALID	2004	Failed to Obtain Parameter Values
ERR_NET_CFG_SET_PARAM_FALID	2005	Failed to Configure Parameters
ERR_NET_CFG_NO_THIS_DATA	2006	Illegal Parameter
ERR_NET_CFG_NO_TABLE	2007	Illegal Parameter
ERR_NET_CFG_INPORT_CFG_DEV_ERR	2008	Configurations and Model Do Not Match
ERR_NET_CFG_INPORT_CFG_CRC_FAILED	2009	Verification of Configurations to Be Imported Failed
ERR_NET_PUI	2100	pui-related Error
ERR_NET_PUI_CHNID_ADDED	2101	Channel Occupied
ERR_NET_PUI_DEV_REPEAT_ADD	2102	Device Already Added
ERR_NET_PUI_DEV_ADD_FAILED	2103	Failed to Add Device
ERR_NET_PUI_CHNID_ADDED_FULL	2104	No Channel Available
ERR_NET_PUI_APPCLT_ERR	2105	Operation Failed
ERR_NET_PUI_DEV_DEL_FAILED	2106	Failed to Delete Device
ERR_NET_PUI_OVER_MAX_USRNUM	2107	Upper Limit for Number of Searching Tasks Reached
ERR_NET_PUI_OVER_MAX_GROUP_NUM	2108	Upper Limit for Number of Groups Reached
ERR_NET_PUI_OVER_MAX_CHN_NUM	2109	Upper Limit for Number of Channels Reached
ERR_NET_PUI_LEN_NOT_ENOUGH	2110	Illegal Parameter
ERR_NET_PUI_OVER_MAX_ACPT_BANDWIDTH	2111	Upper Limit for Access Bandwidth Reached

Error Code	Value	Description
ERR_NET_PUI_PTZ_TASK_RUNING	2112	PTZ Task Occupied
ERR_NET_PUI_VALID_DEV_UPGRADE_TASK	2113	No Upgrade Task Available
ERR_NET_PUI_NO_DETECT_AREA	2114	Number of Motion Detection Areas Cannot Be Zero
ERR_NET_PUI_DEV_FORBIDDEN	2115	Device Disabled
ERR_NET_PUI_AUTH_ID_ERR_FORBIDDEN	2116	Authentication Error
ERR_NET_VTDUCTRL	2200	Nvrvtductrl-related Error
ERR_NET_VTDU_APPCLT_STREAM_PREPARE_FAILED	2201	Failed to Obtain Front-End Stream
ERR_NET_VTDU_APPCLT_STREAM_START_FAILED	2202	Failed to Obtain Front-End Stream
ERR_NET_VTDU_SND_IS_FULL	2203	Sending Capability Exceeded
ERR_NET_VTDU_SEN_RATE_OVER	2204	Sending Capability Exceeded
ERR_NET_VTDU_DEV_OFFLINE	2205	Device Offline
ERR_NET_VTDU_MSIN_NO_STREAM	2206	Failed to Receive Stream
ERR_NET_VTDU_MSIN_CREATE_FAILED	2207	Failed to Create Stream Receiving Object
ERR_NET_VTDU_MSIN_SET_OPT_FAILED	2208	Failed to Configure Stream Receiving Settings
ERR_NET_VTDU_MSIN_SET_TRANSPARAM_FAILED	2209	Failed to Configure Stream Receiving Settings
ERR_NET_VTDU_MSIN_INPUT_DATA_FAILED	2210	Failed to Receive Stream
ERR_NET_VTDU_MSIN_START_FAILED	2211	Failed to Start Receiving Stream
ERR_NET_VTDU_MSIN_STOP_FAILED	2212	Failed to Stop Receiving Stream
ERR_NET_VTDU_MSIN_RELEASE_FAILED	2213	Failed to Stop Receiving Stream
ERR_NET_VTDU_ADD_PIPELINE_FAILED	2215	Failed to Create Stream Output Object
ERR_NET_VTDU_REMOVE_PIPELINE_FAILED	2216	ailed to Stop Sending
ERR_NET_VTDU_MSOUT_CREATE_FAILED	2217	Failed to Create Stream Output Object

Error Code	Value	Description
ERR_NET_VTDU_MSOUT_SET_OPT_FAILED	2218	Failed to Configure the Output Attribute of Stream
ERR_NET_VTDU_MSOUT_SET_TRANSPARAM_FAILED	2219	Failed to Configure Stream Transmission Settings
ERR_NET_VTDU_MSOUT_SET_DATA_CB_FAILED	2220	Failed to Configure the Output Attribute of Stream
ERR_NET_VTDU_MSOUT_GET_DATA_POS_FAILED	2221	Failed to Obtain Position of Stream Data
ERR_NET_VTDU_MSOUT_GET_DATA_FAILED	2222	Failed to Obtain Stream Data
ERR_NET_VTDU_MSOUT_RELEASE_DATA_FAILED	2223	Failed to Release Stream Data
ERR_NET_VTDU_MSOUT_STRAT_FAILED	2224	Failed to Start Sending
ERR_NET_VTDU_MSOUT_STOP_FAILED	2225	Failed to Stop Sending
ERR_NET_VTDU_MSOUT_RELEASE_FAILED	2226	Failed to Stop Sending
ERR_NET_VTDU_INPUT_VID_PARAM_INVALID	2227	Illegal Value for Video Input
ERR_NET_VTDU_INPUT_AUD_PARAM_INVALID	2228	Illegal Value for Audio Input
ERR_NET_VTDU_IS_AUDCALLING	2230	Calling
ERR_NET_VTDU_OVER_MAX_SND_BANDWIDTH	2231	Sending Bandwidth Capability Exceeded
ERR_NET_VTDU_BROADCASTING_NO_SUPPORT_CHN	2235	No channels supporting voice calls are found.
ERR_NET_SMTP_ERR	2301	Failed to Send Mail
ERR_NET_SMTP_FILE_LEN_ERR	2302	Mail Attachment Too Large
ERR_NET_SMTP_PARAM_INVALID	2303	Mail Parameter Error
ERR_NET_SMTP_CONNECT_SERVER_ERR	2304	Failed to Connect to Server
ERR_NET_SMTP_LOGIN_ERR	2305	User Authentication Failed
ERR_NET_SMTP_SEND_ERR	2306	Failed to Send Data
ERR_NET_SMTP_RECV_ERR	2307	Failed to Receive Data
ERR_NET_SMTP_CONNECT_TIME_OUT	2308	Connecting to SMTP Server Timed Out
ERR_NET_SMTP_RESPONSE_ERR	2309	Response Error

Error Code	Value	Description
ERR_NET_SMTP_CONNECT_SSL_ERR	2300	SSL Disconnected
ERR_NET_SMTP_STARTTLS_ERR	2311	Failed to Encrypt Mail
ERR_NET_SMTP_ASSERT_ERR	2312	Illegal Parameter
ERR_NET_SMTP_DOMAIN_ANALY_ERR	2313	Failed to Parse Mail Domain
ERR_NET_CTRLLIB_OVER_MAX_NUM	2701	Archive: no more archives can be added.
ERR_NET_CTRLLIB_OTHER_OPT_IS_DOING	2702	Archive: the archive is being occupied by another application.
ERR_NET_CTRLLIB_SAME_NAME	2703	Archive: the name already exists.
ERR_NET_CTRLLIB_WRITE_DB_FAIL	2705	Archive: writing the archive data into the database failed.
ERR_NET_CTRLLIB_OTHER_USER_OPT_ALG	2706	Archive: another user is configuring the algorithm engine.
ERR_NET_CTRLLIB_CREATE_FILE_FAIL	2707	Archive: failed to create a table
ERR_NET_CTRLLIB_OPEN_DB_FAIL	2708	Archive: failed to open the database
ERR_NET_CTRLLIB_CREATE_TABLE_FAIL	2709	Archive: failed to create a table
ERR_NET_CTRLLIB_EXE_SQL_FAIL	2710	Archive: failed to execute SQL statements
ERR_NET_CTRLLIB_PIC_OVER_RAM_SIZE	2711	Archive: the picture is too large
ERR_NET_CTRLLIB_SYS_CMD_FAIL	2712	Archive: failed to execute system commands.
ERR_NET_CTRLLIB_OTHER_USER_IMPORT	2713	Archive: the picture import function is occupied by another user.
ERR_NET_CTRLLIB_CHECK_IMPORT_FAIL	2714	Archive: verification of imported files failed
ERR_NET_CTRLLIB_CREATE_THREAD_FAIL	2715	Archive: failed to create threads
ERR_NET_CTRLLIB_EGI_FAIL	2716	Archive: failed to extract characteristics

Error Code	Value	Description
ERR_NET_CTRLLIB_COMPARE_USED	2717	Archive: the archive is being occupied by a compare rule.
ERR_NET_CTRLLIB_OVER_SUP_MAX	2718	Archive: the upper limit of an archive is reached.
ERR_NET_AIS_PIC_QUERY_RESULT_OVER_NUM	2719	Archive: excessive search results may be produced. You are advised to narrow down the time range.
ERR_NET_AIS_PIC_NO_FACE_PIC_FEATURE	2720	Archive: no faces are detected.
ERR_NET_AIS_ADD_NO_EGI	2721	Archive: personnel information is added to the archive, but the data model is not created.
ERR_NET_AIS_OVER_SUP_DETECT_NUM	2722	Archive: the detection capability is exceeded.
ERR_NET_AIS_ALG_INVALID	2723	Archive: this algorithm cannot be found.
ERR_NET_AIS_CREATE_DETECT_FAILED	2724	Archive: failed to create detection handles.
ERR_NET_AIS_OVER_CMP_CHN_NUM	2725	Archive: no more channels can be added.
ERR_NET_AIS_RULE_NAME_EXIST	2726	Archive: the name already exists.
ERR_NET_AIS_CMP_CTRLLIB_EXIST	2727	Archive: the archive already exists.
ERR_NET_AIS_OVER_SUP_RULE_NUM	2728	Archive: no more rules can be created.
ERR_NET_AIS_CREATE_CMP_HANDLE_FAILED	2729	Archive: failed to create compare handles.
ERR_NET_CTRLLIB_MEM_ALREADY_DEL	2730	Archive: operation failed because the target had been deleted
Definition of internal netsdk	Error code range: 6000~6499	
ERR_NET_INIT_FAILED	6000	Initialization Failed
ERR_NET_INVALID_HANDLE	6001	Invalid Handle
ERR_NET_INVALID_PARAM	6002	Illegal Parameter

Error Code	Value	Description
ERR_NET_PARSE_FAILED	6003	Parsing Failed
ERR_NET_CREATE_HANDLE_FAILED	6004	Failed to Create a Handle
ERR_NET_ALLOC_MEM_FAILED	6005	Failed to Apply for Memory
ERR_NET_SOCKET_OPT_FAILED	6006	Failed to Configure Network Attributes
ERR_NET_CONNECT_FAILED	6007	Connection Failed
ERR_NET_CONNECT_CLOSED	6008	Disconnected
ERR_NET_SEND_FAILED	6009	Sending Failed
ERR_NET_RECV_FAILED	6010	Receiving Failed
ERR_NET_SEND_TIMEOUT	6011	Sending Timed Out
ERR_NET_RECV_TIMEOUT	6012	Receiving Timed Out
ERR_NET_SEND_OUT_MEM	6013	Insufficient Sending Buffer
ERR_NET_RECV_OUT_MEM	6014	Insufficient Receiving Buffer

Commissioning

Command	Description
netsdkhelp	Display commissioning commands.
netsdkver	Version
netsdksetsl	Configure the screen printing level; 0: close printing; 1: print to a file; 2: print error information; 3: print import information; 4: print commissioning information; 5: print temporary information; 6: print all
netsdksetfl	Configure the file printing level; 0: close printing; 1: print to a file; 2: print error information; 3: print import information; 4: print commissioning information; 5: print temporary information; 6: print all

How to Use
telnet 127.0.0.1 2400